Concept Drift Handling in Information Systems: Preserving the Validity of Deployed Machine Learning Models

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften
(Dr.-Ing.)

von der KIT-Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Lucas Baier

Tag der mündlichen Prüfung: 07.07.2021

Referent: Prof. Dr. Gerhard Satzger

Korreferent: Prof. Dr. Ali Sunyaev

Abstract

Predictions computed by supervised machine learning models play a crucial role in a variety of innovative applications in business and industry. Typically, value is generated as soon as these models are deployed and continuously used in information systems of an organization. However, machine learning endeavors predominantly focus on conceiving applications for static situations. In this context, the management of the models' lifecycle to preserve their effectiveness over time in dynamic environments is still in its infancy.

Therefore, this thesis starts with systematically analyzing the full lifecycle of machine learning applications from an information systems (IS) perspective—and understanding and documenting all choices that have to be made throughout this cycle. On that basis, we then perform a qualitative study via practitioner interviews to map particular challenges in the deployment phase. In this context, we identify *concept drift* as a particularly important challenge to overcome: Concept drift refers to changes in the environment over time which affect the behavior of a machine learning model. This can have an impact on the model's prediction quality and its overall utility.

We analyze and categorize concept drift handling approaches covering both the detection of concept drift as well as the appropriate adaptation of the model. We identify two particular research gaps: the handling of concept drift for regression tasks and the handling of concept drift for tasks where additional labels for retraining a model are hard or costly to obtain. For both areas, we develop new methods and demonstrate their effectiveness in technical experiments and real-world use cases.

This thesis contributes new methods to handle concept drift, for particular difficult contexts: First, this thesis should raise researchers' and practitioners' awareness for the topic of changing input data over time as well as for its impact on deployed machine learning. Second, the developed and tested methods can either be implemented directly or serve as inspiration to conceive appropriate drift handling strategies within information systems. Finally, we expect that advanced concept drift handling not only technically ensures a reliable prediction quality over time, but that it will also increase trust and acceptance of machine learning-based information systems—and, thus, help to boost the impact of machine learning.

Acknowledgement

Completing this dissertation would not have been possible without the support of many inspiring people surrounding me in the last four years. First, I am very grateful to my supervisor Prof. Dr. Gerhard Satzger for allowing me to pursue a PhD at his chair and his constant supervision and guidance during this entire time. Furthermore, I want to thank Prof. Dr. Ali Sunyaev for reviewing my dissertation. Additionally, I want to thank Prof. Dr. Stefan Nickel and Prof. Dr. Benjamin Scheibehenne for serving on the dissertation committee and for their inspiring questions and ideas for future research.

My PhD time would not have been the same experience without the fantastic team at the chair for Digital Service Innovation. In this context, I want to especially thank my post-doctoral supervisor Dr. Niklas Kühl who constantly supported my academic work. It was always a great pleasure to discuss and develop ideas with him and his very positive attitude helped me to overcome the setbacks associated with every PhD thesis. Furthermore, the Applied AI in Services lab is a great place for any researcher because my colleagues always took the time for interesting and inspiring discussions regarding all my research projects. However, I also want to thank all team members specifically for the time which we spent apart from research because this was always a pleasant distraction. Regarding this final document, I especially want to thank Jannis, Jakob, Philipp, and Lucas for reviewing and proofreading this dissertation.

Pursuing a PhD at the KIT provides the unique opportunity to work together with very motivated and creative students. Some of these students have greatly supported my work in recent years and often surprised me with their inspiring ideas regarding concept drift handling. In this context, I want to especially highlight Josua, Fabian, Marcel, Vincent and Tim for their contributions regarding joint research projects.

Finally, I want to thank my friends and family: Philipp and Michael whose experiences initially sparked my interest in pursuing a PhD. Tsun-Tao, Henning, and Jana for always being there for spending time apart from work. Furthermore, I am very grateful to my parents and my sister for supporting me in every possible situation. Above all, I want to thank my wife Maraike and our little son Jakob who always stood by my side during this entire time. Their presence reminded me of the

important things in life, provided the necessary distractions, and lastly, they have always succeeded in cheering me up.

Contents

I Fundamentals **1**

1 Introduction **3**
1.1 Motivation 4
1.2 Research Design 7
1.3 Structure of Work 12

2 Related Work **17**
2.1 AI and Machine Learning in IS 17
2.2 Learning in Dynamic Environments 21
2.2.1 Machine Learning in Data Streams 21
2.2.2 Online Learning 24
2.3 Concept Drift 26
2.3.1 Definition 26
2.3.2 Concept Drift Handling 29
2.3.3 Applications and Current Research 38

II Choices and Challenges for Machine Learning Applications **41**

3 The Supervised Machine Learning Reportcard **43**
3.1 Introduction 43
3.2 Fundamentals and Positioning 44
3.3 Towards Rigorous Supervised Machine Learning Documentation . . 48
3.3.1 Problem Characteristics and Key Choices of Supervised Machine Learning 48
3.3.2 The Supervised Machine Learning Reportcard (SMLR) . . . 55
3.4 Empirical Study 59
3.4.1 Methodology and Data Set 59
3.4.2 Model Initiation 61
3.4.3 Performance Estimation 63
3.4.4 Model Deployment 64
3.5 Conclusion 65

4 Challenges in the Deployment of Machine Learning 67
4.1 Introduction . . . 67
4.2 Related Work . . . 68
4.3 Research Methodology . . . 74
4.3.1 Sampling . . . 74
4.3.2 Data Collection and Analysis . . . 74
4.4 Results . . . 75
4.4.1 Pre-Deployment . . . 77
4.4.2 Deployment . . . 78
4.4.3 Non-Technical Challenges . . . 80
4.5 Discussion . . . 81
4.6 Conclusion and Outlook . . . 86

III Challenges for the Application of Concept Drift Handling 89

5 Handling Concept Drift for Predictions in Business Process Mining 91
5.1 Introduction . . . 91
5.2 Related Work . . . 93
5.2.1 Concept Drift . . . 93
5.2.2 Process Mining . . . 94
5.3 Data Selection for Retraining . . . 96
5.3.1 Learning Mode . . . 96
5.3.2 Data Selection for Retraining of the Machine Learning Model 97
5.4 Use Case in Process Mining . . . 98
5.4.1 Data Analysis . . . 99
5.4.2 Evaluation of Prediction . . . 102
5.5 Conclusion . . . 105

6 Preserving Validity of Predictive Services over Time 107
6.1 Introduction . . . 107
6.2 Foundations . . . 109
6.2.1 Machine Learning for Services . . . 109
6.2.2 Concept Drift . . . 110
6.2.3 Predictive Services . . . 111
6.3 Conceptual Framework . . . 112
6.3.1 Methodology . . . 112
6.3.2 Setup Decisions for Predictive Service . . . 113
6.3.3 Algorithmic Decisions . . . 114
6.3.4 Operation of Predictive Service . . . 116

6.3.5 Heatmap of Research Papers . . . 118
6.4 A Research Agenda for Preserving Validity of Predictive Services Over Time . . . 120
6.5 Conclusion . . . 122

IV Concept Drift Handling for Regression Problems 125

7 Handling by Switching Models - the Error Intersection Approach 127
7.1 Introduction and Related Work . . . 127
7.2 Use Case . . . 131
7.2.1 New York City Taxi Dataset . . . 132
7.2.2 Exemplary Drifts . . . 132
7.3 Design of the Error Intersection Approach . . . 135
7.4 First Evaluation . . . 136
7.5 Discussion . . . 138
7.6 Conclusion . . . 140

8 Handling by Switching Adaptation Mode - the Switching Scheme 143
8.1 Introduction . . . 143
8.2 Related Work . . . 145
8.2.1 Concept Drift . . . 145
8.2.2 Demand Forecast . . . 146
8.2.3 Research Gap and Contribution . . . 147
8.3 Use Case . . . 147
8.4 Methodology for Handling Incremental Drift . . . 149
8.4.1 Adaptation Strategies . . . 149
8.4.2 Drift Detectors . . . 151
8.5 Evaluation . . . 152
8.5.1 Evaluation of Pre-Test . . . 152
8.5.2 Evaluation of Adaptation Strategies . . . 155
8.5.3 Robustness Check . . . 158
8.6 Conclusion . . . 159

V Concept Drift Handling with Limited Label Availability 161

9 Handling by Model Uncertainty - Uncertainty Drift Detection 163
9.1 Introduction . . . 163
9.2 Background and Related Work . . . 164
9.2.1 Dataset Shift and Concept Drift . . . 164

9.2.2 Handling Concept Drift . . . 165
9.2.3 Uncertainty in Neural Networks . . . 166
9.3 Methodology . . . 167
9.4 Experiments . . . 170
9.4.1 Experimental Setup . . . 170
9.4.2 Data Sets . . . 172
9.4.3 Performance Metrics . . . 174
9.4.4 Analysis on Synthetic Data Sets . . . 174
9.4.5 Experimental Results . . . 176
9.5 Conclusion . . . 178

10 Handling by Outlier Detection - the Two-Step Prediction Method 181
10.1 Introduction . . . 181
10.2 Foundations . . . 183
10.2.1 Machine Learning . . . 183
10.2.2 Concept Drift . . . 184
10.2.3 Outlier Detection . . . 184
10.3 Problem Definition and Requirements . . . 185
10.4 Design Options . . . 187
10.4.1 Step 1: Data Validity . . . 187
10.4.2 Step 2: Model Robustness . . . 188
10.5 Evaluation . . . 188
10.5.1 Evaluation of Data Validity (Step 1) . . . 190
10.5.2 Evaluation of Model Robustness (Step 2) . . . 194
10.5.3 Evaluation of Overall Prediction Method . . . 197
10.6 Conclusion . . . 200

VI Finale 203

11 Conclusion 205
11.1 Summary and Contributions . . . 205
11.2 Practical Implications . . . 214
11.3 Limitations and Future Research . . . 216

Bibliography 221

A Appendix 265

Declarations 273

List of Abbreviations

ADWIN	Adaptive Windowing
AI	Artificial Intelligence
AUC	Area Under the Curve
CC	Correlation Coefficient
CO	Complex Operations
COD	Coefficient Of Determination
CRISP-DM	Cross-Industry Standard Process for Data Mining
CS	Computer Science
D&M	DeLone & McLean
DDM	Drift Detection Method
EIA	Error Intersection Approach
ERP	Enterprise Resource Planning
EWMA	Exponential Weighted Moving Average
FHVs	For-Hire-Vehicles
FPN	Floating Point Numbers
GD	Global Deviation
HDDDM	Hellinger Distance Drift Detection Method
HLFR	Hierarchical Linear Four Rate
IS	Information Systems
ITA	Information Theoretic Approach
JIT	Just-In-Time
KDD	Knowledge Discovery in Databases
KNN	K-Nearest-Neighbors
KSWIN	Kolmogorov-Smirnov Window
LSTM	Long Short-Term Memory
MAPE	Mean Absolute Percentage Error
MD	Mahalanobis Distance
MK	Mann-Kendall
ML	Machine Learning
MLP	MultiLayer Perceptron
MSE	Mean Squared Error
NLP	Natural Language Processing

NN	Neural Network
NRMSE	Normalized Root Mean Squared Error
NYC	New York City
PCA-CD	Prinicipal Component Analysis-based Change Detection
PH	Page-Hinkley
RMSE	Root Mean Squared Error
RQ	Research Question
SMAPE	Symmetric Mean Absolute Percentage Error
SML	Supervised Machine Learning
SMLR	Supervised Machine Learning Reportcard
SNR	Signal-Noise Ratio
SO	Simple Operations
STEPD	Statistical Test of Equal Proportions
SVR	Support Vector Regression
TLC	Taxi and Limousine Commission
TSMSD	Two-Stage Multivariate Shift-Detection
UDD	Uncertainty Drift Detection

Part I

Fundamentals

1 Introduction

In recent years, many organizations have explored sophisticated techniques from the field of machine learning to gain competitive advantages. Scientific breakthroughs have resulted in a series of exciting applications. Examples range from the impressive performance in playing Jeopardy (Ferrucci et al., 2013) to outperforming humans in object recognition on the ImageNet dataset with 1000 different object classes (Russakovsky et al., 2015) or mastering the complex board game Go even without training on human generated matches (Silver et al., 2017). More recently, surprising results have been achieved with machine learning systems performing in video games where they master the games of Dota2 (OpenAI et al., 2019) or Starcraft II (Vinyals et al., 2019) better than professional video gamers. Furthermore, machine learning is boosting innovation in autonomous driving applications (Janai et al., 2020). Tremendous progress has also been achieved in developing new models for natural language processing which are applicable to various tasks such as translation and question answering (Brown et al., 2020). In industrial settings, machine learning-based predictive maintenance solutions are applied to reduce production downtimes (Carvalho et al., 2019).

Progress in the field is due to multiple reasons: One major breakthrough in recent years is the access to improved learning algorithms such as deep neural networks (Brynjolfsson & Mitchell, 2017; Davenport, 2018; Jöhnk et al., 2020). Furthermore, the sheer volume of data available makes it possible to appropriately train these models (Zhou et al., 2017). This allows to capture highly valuable regularities that enable models to outperform the best humans at a task (Brynjolfsson & Mitchell, 2017). Increased computing power enabled by the design of better microchips (Monroe, 2018; Sunyaev, 2020b) as well as more storage capacity allow organizations to make use of the available data and train large enough models (Ågerfalk, 2020; Thrall et al., 2018). Another important reason for explaining machine learning dissemination is the broad availability of open source libraries (Ambati, 2019; Prado et al., 2020). This allows to accelerate machine learning projects without the necessity to develop frameworks from scratch (Rock, 2020).

The aforementioned reasons have also spurred the widespread adoption of machine learning in many industries (Bughin et al., 2017) with rapidly increasing investments

(Pumplun et al., 2019). Currently, companies are moving from R&D solutions, which were previously the main focus of budget spending (Bughin et al., 2017), to deployed solutions (Holstein et al., 2019; Schelter et al., 2018). This allows organizations to enhance existing products or services (Schüritz et al., 2017). Due to the increasing share of deployed models, all steps of the machine learning lifecycle—from data cleansing to model building and deployment as well as proper monitoring of the model—must be taken into account (Wang, Ram, et al., 2020). In this context, appropriate monitoring of models is especially critical since deployed machine learning models are continuously fed with new data instances which are changing over time as the related real-world phenomenon evolves.

The most widely used machine learning technique today is supervised machine learning, where the objective is to learn a function that maps the input data X to a corresponding label y (Jordan & Mitchell, 2015). A model is learned by considering a set of training data samples where the information regarding both the input data X and labels y is available. Once a model is trained, it can be deployed into production to generate predictions for new incoming data instances. From this moment, the model can generate added value for an organization in many applications. However, the underlying distribution of the input data X or the relationship between the input data X and the target y may change over time (Gama et al., 2014). This phenomenon is also called *concept drift* (Widmer & Kubat, 1996). Concept drifts can have a large impact on the prediction quality of the underlying prediction model (Lindstrom et al., 2013; Lu et al., 2019; Žliobaitė et al., 2016). Due to the substantial influence of machine learning models in many core components of organizations, it is crucial that appropriate strategies are applied to tackle this challenge. Otherwise, today's complex systems optimized via machine learning (e.g., multi-stage supply chain networks) will fail to deliver their value proposition—with far-reaching implications.

1.1 Motivation

Information systems are increasingly improved by the application of machine learning. While the adoption of this technology is still in its infancy, it eventually has the potential to enhance every single component of information systems (Bawack et al., 2019). This technological change also requires an adaptation and reinvention of knowledge regarding efficient information systems management (Berente et al., 2019). To keep a competitive advantage, companies need to adapt on various levels since machine learning triggers change not only at the execution of tasks and

processes but also influences entire business models in different application areas (Jöhnk et al., 2020).

Well-known examples of machine learning-based information systems are digital assistants, such as chatbots, which aim to support humans for specific tasks in contexts such as smart homes or customer services (Maedche et al., 2019). In other scenarios, the analytical capabilities of machine learning are directly applied to improve internal processes, e.g., by identifying customers prone to churn (Cai et al., 2018). Furthermore, machine learning can be utilized to create entire new service offerings (Schüritz & Satzger, 2016) such as predictive maintenance solutions (Lüttenberg et al., 2018). Due to this widespread use of machine learning technologies, it is important to guarantee the robustness and validity of those solutions. However, the application of machine learning is associated with various difficult choices and challenges. One of the identified challenges in the context of deployed models is changing input data over time. Therefore, any information system needs to be prepared to handle changing data as well as the corresponding decay in prediction quality over time.

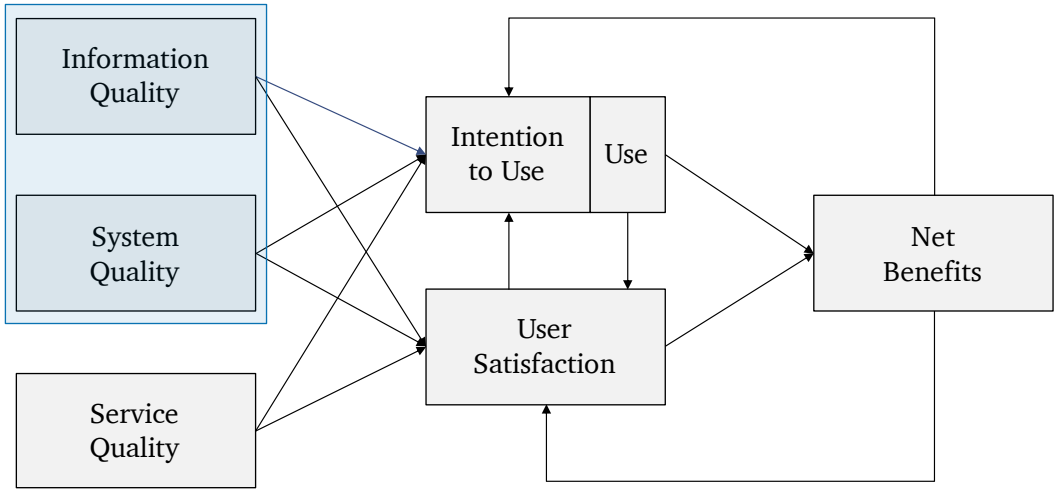


Fig. 1.1.: D&M information systems success model, adapted from Delone and McLean (2003, p. 24).

The DeLone & McLean (D&M) information systems success model (Delone & McLean, 2003) depicted in Figure 1.1 identifies different influencing factors on the net benefits of an information system. Based on this model, the success of an information system is mainly determined by various quality aspects which can be divided into three different components: information quality, system quality and service quality (Delone & McLean, 2003). Information quality can be measured in terms of accuracy, timeliness, completeness, relevance, and consistency. In this context, machine learning models with decreasing prediction quality surely affect the *accuracy* as well

as the *relevance* of information quality with corresponding impact on the benefits of the deployed information system. System quality can be measured in terms of *functionality* and *reliability*, where reliability can be understood as the probability that the system produces correct outputs (Sunyaev, 2020a). These two attributes are also negatively influenced by predictions with decreasing quality.[1]

Whereas the previous paragraph highlights the need for the management of changing data from a success model perspective, this requirement can also be motivated by a more process-oriented view on information systems. Similar to many other products or services, information systems use follows a lifecycle (Duarte & Costa, 2012) which is depicted in Figure 1.2. Initially, this lifecycle was mainly derived from observations during the introduction and usage of Enterprise Resource Planning (ERP) systems (Stefanou, 2001), but the phases are also applicable to other forms of information systems. During the application and use, organizations need to be continuously proactive and adapt their information system to constant changes. This especially applies to machine learning-enhanced components that are required to process changing input data. Furthermore, the decline of such systems can be delayed with correct management decisions regarding robustness and updates.

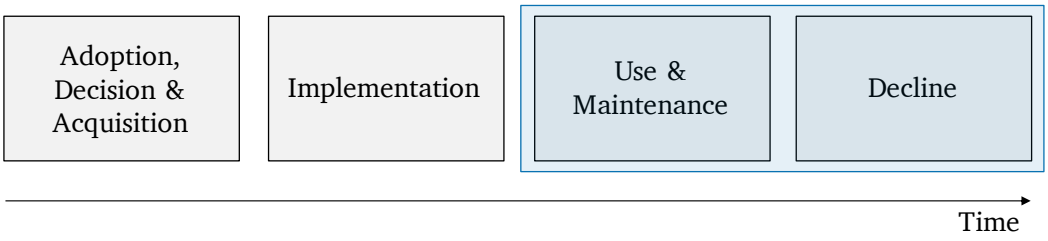


Fig. 1.2.: Lifecycle of information systems, adapted from Duarte and Costa (2012, p. 27).

Besides the motivation from an information systems perspective on proper management of changing data, the need for adaptations for deployed machine learning models is also given on an economic and ethical level (Schelter et al., 2018). In the real world, data distributions are not stable but change over time and thereby influence prediction quality. In recent years, the economic impact of machine learning models has grown significantly with potentially huge consequences in case of failures (Bughin et al., 2017). Furthermore, individuals are often largely affected by machine learning-based decision making, e.g., in lending (Barocas et al., 2017). Therefore, it is critical that machine learning models are continuously adapted to generate correct predictions. The phenomenon of changing data distributions over

[1] Depending on the definition, service quality can also be affected by decreasing prediction quality. However, in the context of the D&M success model, service quality mainly refers to the responsiveness and assurance of the customer support associated with an information system and is therefore not influenced by a change in prediction performance.

time is described with the term concept drift the in computer science literature (Gama et al., 2014).

Examples of data changes influencing machine learning predictions can be observed in isolated settings such as replacing the input material in a production process in a predictive maintenance setting. However, also on a global level, exceptional events such as 9/11, the financial crisis in 2008/09, or COVID-19 have significant macroeconomic consequences impacting the accuracy of demand prediction models in different areas.

Due to the more widespread use of machine learning models in recent years, the changes during the COVID-19 pandemic offer various examples where machine learning models have experienced problems. For instance, many retailers needed to adapt their deployed demand forecasting models because they provided inaccurate predictions with the start of the COVID-19 crisis (Deeplearning.AI, 2020). Furthermore, streaming services were surprised by changing consumer behavior leading to less relevant content recommendations (Heaven, 2020), and food inventory management systems based on sales forecasting struggled due to large bulk orders (Heaven, 2020). The financial markets also provide various examples where machine learning-based funds performed poorly during the March 2020 crisis (Knight, 2020).

These examples clearly demonstrate the necessity to monitor and frequently adapt deployed machine learning models to cope with changing environments. However, adaptation strategies also need to be carefully implemented: In 2016, Microsoft released a pretrained chatbot named “Tay” on Twitter which was equipped with self-learning and adaptation capabilities. Within a few hours, the bot published a variety of racist, antisemitic and offensive tweets by adapting its behavior based on interactions with other Twitter users (Lee, 2016). In another example, Facebook engineers released various chatbots which were allowed to adapt the underlying machine learning models based on their experiences after deployment. The chatbots quickly developed their own language that was completely incomprehensible to humans (Wilson, 2017).

1.2 Research Design

Information systems are increasingly enhanced by machine learning techniques (Buxmann et al., 2021). Therefore, it is important to understand the problem of changing data over time in this context. Since finding unique solutions requires

an in-depth understanding of the problem space (Sturm & Sunyaev, 2019), we need to consider relevant knowledge from Computer Science (CS) literature for the technical details as well as Information Systems (IS) literature for the requirements from an information system's perspective. Since most systems based on machine learning in practical use apply supervised machine learning (Lipton, 2018), we seek to better comprehend the application of this technique in IS. To this end, we aim to explore and elaborate on the necessary steps and choices for the application of supervised machine learning in IS. Furthermore, we analyze to which extent existing IS literature fulfills these documentation requirements.

Research Question 1 (RQ1)
Which choices are relevant in the application of supervised machine learning in IS research?

We answer Research Question (RQ) 1 by developing a reportcard for the application of supervised machine learning. The reportcard describes which choices need to be documented to ensure reproducibility of research results (Hutson, 2018). It divides the application process into three phases: model initiation, performance estimation and model deployment. Subsequently, we use the reportcard to analyze published IS research papers applying machine learning. The results indicate that these papers only scarcely report information about the model deployment phase, which might be connected to a lack of knowledge regarding this topic. However, this phenomenon might also be explained by the fact that researchers may only want to prove feasibility of a machine learning approach. In this case, a detailed discussion of deployment is not required. Nevertheless, difficulties with deployment are also reported in other research domains such as network traffic analysis (Pacheco et al., 2019). In industrial applications in general, machine learning deployment is a difficult endeavor with several challenges such as access to different data sources, removal of data silos, and hardware requirements (Jöhnk et al., 2020; Schelter et al., 2018). Therefore, we aim to better understand the challenges of machine learning operation and deployment in practical settings.

Research Question 2 (RQ2)
Which challenges arise regarding the deployment of supervised machine learning models?

We explore RQ2 by performing an interview study with machine learning practitioners. The interviews reveal numerous challenges including the standardization of

machine learning infrastructure but also non-technical challenges such as appropriate communication and expectation management. Furthermore, another challenge refers to concept drift (changing data distributions) and its impact on deployed machine learning solutions. Models need to be intensively monitored over time, and adaptations are required to ensure high-quality predictions. Otherwise, machine learning models will achieve poor learning results and provide predictions with low accuracy in a changing environment (Lu et al., 2019). Therefore, the phenomenon of concept drift and the challenges associated with it are investigated in more detail by considering the next research question.

Research Question 3 (RQ3)
What are typical challenges for the application of concept drift handling algorithms?

RQ3 is investigated by pursuing two different research projects. First, we instantiate concept drift handling in a technical experiment based on a real-world use case. This enables us to better understand the effects of different handling strategies. Additionally, we also elaborate on different data selection strategies in case of retraining—a typical challenge in concept drift problems—and evaluate their impact on prediction performance. Second, we perform a literature review regarding concept drift handling projects in real application use cases. Based on the identified literature, we derive a framework which describes different algorithmic options and characteristics for deployed machine learning models confronted with concept drift.

This framework allows us to identify various areas where concept drift handling solutions are still limited. Current concept drift algorithms are mainly applicable to classification problems (Jaworski, 2018), which is why concept drift handling and its effects have been widely studied in this setting (Cavalcante et al., 2016). The focus on classification tasks can be explained by the fact that many algorithms make specific assumptions regarding the distribution of the prediction error (Gama et al., 2014). Nevertheless, predicting numerical outputs (regression) is another key activity within supervised machine learning relevant for many real-world tasks (Harrington, 2012). However, regression and especially time series forecasting problems have different characteristics that need to be considered for concept drift handling (Cavalcante et al., 2016). This is reflected in the fourth research question.

Research Question 4 (RQ4)
How can concept drift in regression problems be handled?

We explore RQ4 by performing two technical experiments. Those experiments are used to evaluate the feasibility and performance of two methods that we propose to handle concept drift in regression problems. Both methods are based on a switching mechanism. While the first method (*Error Intersection Approach*) switches between two prediction models for concept drift handling, the second method (*Switching Scheme*) switches between two different modes of adaptation for prediction models.

In detail, the *Error Intersection Approach* applies a simple as well as a complex prediction model for handling concept drift in time series problems. The complex model is applied during periods with regular patterns because it accurately captures the general components of a time series. However, in times with sudden concept drift, the simple model is utilized since it is capable of quickly adapting to the current situation. In contrast, the *Switching Scheme* changes between performing incremental updates and retraining of prediction models.

Besides the need for specific methods in regression contexts, true label availability in operation is another critical aspect for the application of concept drift handling strategies. Most concept drift detection algorithms are based on the error rate of the underlying machine learning model (Lu et al., 2019). Computing the error rate of a model requires the acquisition of true labels. However, obtaining true labels can be expensive in real-world applications (Lindstrom et al., 2013), e.g., because domain experts need to label new data instances. In other applications, true labels will only be available with significant delay (Krawczyk et al., 2017). For instance, the information whether a customer will default her credit card payments within one year is only available after the entire year has been elapsed. Furthermore, the speed and sheer volume of data streams in many real-world applications can simply rule out the possibility to acquire all true labels (Hu et al., 2020; Lu et al., 2019). Therefore, concept drift handling strategies for dealing with limited label availability are investigated in research question 5.

Research Question 5 (RQ5)
How can concept drift be handled in machine learning settings with limited availability of true labels?

RQ5 is answered by performing technical experiments evaluating two methods for concept drift handling in settings with limited availability of true labels. While the first method (*Uncertainty Drift Detection (UDD)*) detects drift by monitoring uncertainty metrics and triggers corresponding adaptation, the second method (*Two-Step Prediction Method*) prevents negative effects of concept drift by improving model robustness with an outlier detection.

To be specific, *UDD* detects concept drift without requiring true labels by considering uncertainty values associated with predictions. In this context, uncertainty refers to the information how certain a model is about its computed predictions. For *UDD*, we consider neural networks as prediction models and derive uncertainty based on Monte-Carlo Dropout (Gal & Ghahramani, 2016). Nevertheless, this method requires a limited set of true labels for retraining in case of drift. In contrast, the *Two-Step Prediction Method* follows a different approach. Instead of adapting to concept drift, it focuses on providing robust machine learning services in concept drifting environments. This is achieved by combining an outlier prediction with a robust machine learning model.

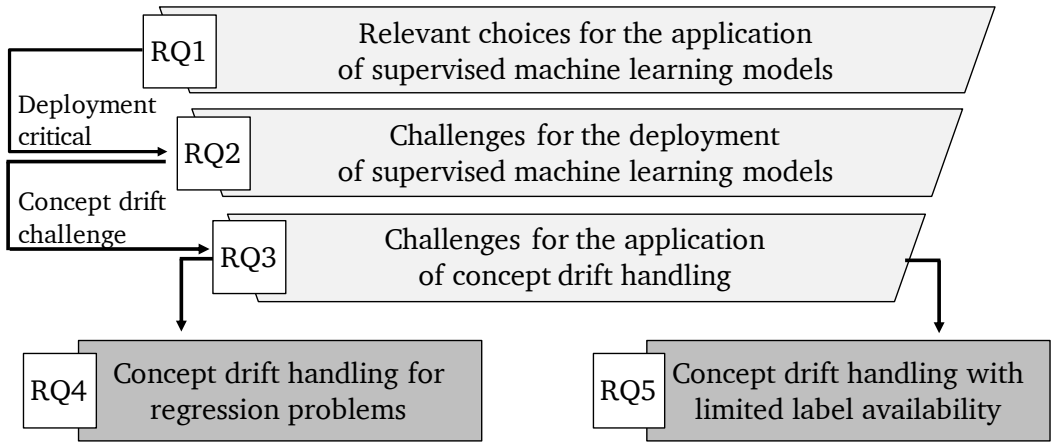


Fig. 1.3.: Overview of thesis content.

The overall content of this thesis is summarized in Figure 1.3. First, this thesis describes relevant choices and documentation requirements for the application of supervised machine learning models in IS (RQ1). However, the adoption of machine learning technologies is still confronted with a large number of challenges. Those are investigated in detail by performing an interview study (RQ2). One of the major identified challenges refers to concept drift or changing data distributions over time as well as its impact on prediction performance. For an improved understanding of this problem, we perform two different activities: First, a concept drift algorithm is instantiated on a real-world use case. Second, a framework describing typical characteristics and choices for concept drift handling strategies is developed. Those

activities allow to derive typical challenges for concept drift handling (RQ3). Based on this acquired knowledge, two aspects for improvement of existing solutions are investigated: concept drift handling for regression problems (RQ4) and concept drift handling with limited label availability (RQ5).

For answering the research questions, we apply a set of different suitable research methods. While we apply mainly qualitative research methods regarding RQ1-3, quantitative methods are in focus for working with RQ4 and RQ5. RQ1 is answered by performing a structured literature review and developing a reportcard. For dealing with RQ2, we perform an expert interview study with a qualitative content analysis. RQ3 is tackled from two different perspectives: While the first part is based on a technical experiment, the second part builds on a structured literature review. The final RQ4 and RQ5 are both answered by performing two technical experiments each per research question.

1.3 Structure of Work

This thesis consists of six main parts with several chapters each. Part I covers general foundations and Part II describes necessary steps as well as various challenges during the application of supervised machine learning. Part III sheds light on the problem of changing data distributions over time and its impact on deployed models. Part IV and Part V offer solutions for specific aspects of concept drift handling, namely in regression problems and in situations with limited availability of true labels. In Part VI, the findings of the previous parts are summarized and directions for future research are outlined. Figure 1.4 provides an overview of the thesis's structure that also shows which peer-reviewed publications are included in the individual chapters. Note that this figure adopts the general structure introduced in Figure 1.3.

In Part I, we lay the foundations for this thesis. Chapter 1 introduces the motivation as well as the research design. Chapter 2 presents relevant related work. First, we review the role and applications of machine learning in IS. This is followed by an overview of research considering machine learning in the context of data streams as well as concept drift and strategies for handling concept drift.

Part II describes fundamental steps and challenges in the application of supervised machine learning. Chapter 3 describes the development of a reportcard for documenting necessary steps and choices in any kind of supervised machine learning endeavor in IS. A subsequent analysis of 121 research papers with the help of this

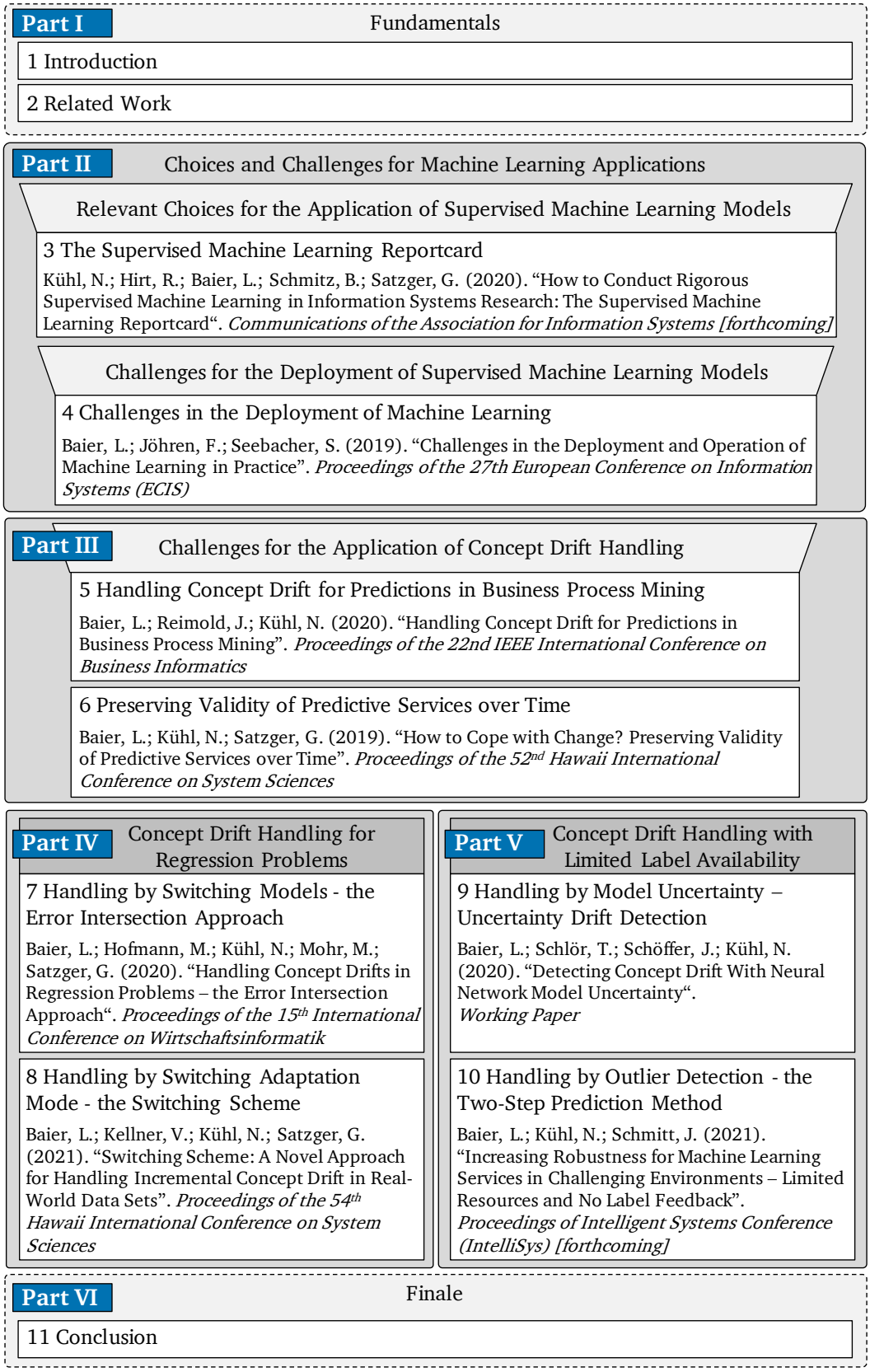


Fig. 1.4.: Structure of this thesis.

reportcard reveals various weaknesses regarding the documentation and implementation of machine learning projects. Therefore, Chapter 4 investigates and discusses corresponding challenges in the operation and deployment of machine learning in practice. To this end, we conduct an interview study with 11 machine learning experts across various industries such as manufacturing, health care or finance and perform a qualitative content analysis. This approach allows us to identify key challenges along three categories and six clusters.

Part III introduces the general problem of changing data over time. Chapter 5 contains a technical experiment where concept drift handling strategies are instantiated in a use case requiring predictions for business process mining. We can show that the application of those strategies significantly improves the performance for predicting the delivery time of ordered goods. This chapter also illustrates the challenges for setting up those systems with real-world data. The second chapter in this part (Chapter 6) introduces a framework which describes possible alternatives for setting up machine learning services for dealing with concept drift and is based on a literature review of 34 research papers. Different design options for services in this context can be divided into setup decisions, algorithmic decisions and decisions regarding operation.

In Part IV, we investigate two methods for handling concept drift in regression problems and also evaluate those with technical experiments. Chapter 7 describes the Error Intersection Approach—a concept drift handling method—which is based on two prediction models of different complexity for handling sudden concept drift in demand forecasting. We instantiate this method on a data set describing the taxi ride demand in New York City and show its superior overall prediction performance. Furthermore, we also illustrate the operating principle of the method by investigating specific examples of concept drift in detail. Whereas the previous method focuses on sudden concept drift, Chapter 8 introduces the Switching Scheme, a newly developed method for handling incremental concept drift. It describes a mechanism for combining the advantages of different adaptation strategies for machine learning models. This method is also evaluated on the New York City taxi ride data set as well as on a data set containing information about flight delays in the US.

Part V contains two methods for handling concept drift in deployment situations where the acquisition of true labels is difficult. The evaluation of both chapters in this part is based on technical experiments. Chapter 9 introduces the Uncertainty Drift Detection (UDD) approach which applies the uncertainty of a neural network for detecting concept drift. The approach is evaluated on two synthetic data sets as well as on a set of eight classification and two regression real-world data sets

which are often applied for testing novel concept drift strategies. Furthermore, Chapter 10 introduces the Two-Step Prediction Method for increasing the robustness of machine learning services. In contrast to the previous contributions, this method does not focus on adapting the prediction model on novel concepts but rather guarantees robustness regarding the issued predictions. This behavior is required since it is impossible to acquire any true labels and, therefore, adapt the model during operation at all. The method is validated with a data set from a large German OEM with the objective to optimize engine control.

The final Part VI concludes this thesis by providing a summary of the insights gained from working on the different research questions. It also describes practical implications of this thesis and outlines limitations as well as provides potential ideas for future research (Chapter 11).

A different view on the content of this thesis is provided in Figure 1.5 on page 16. It illustrates the relationship between research questions, the corresponding chapters of this thesis as well as the employed research methods.

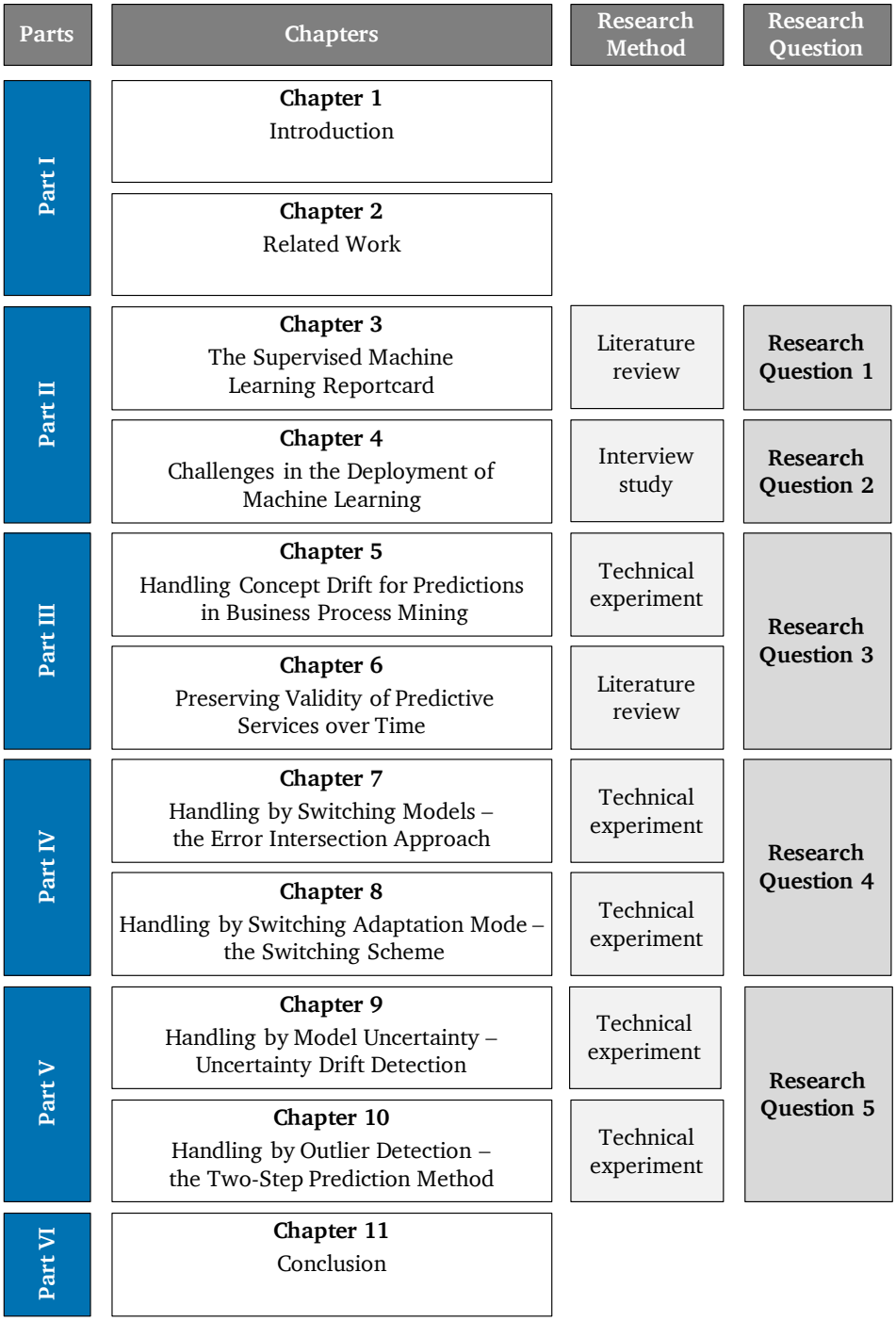


Fig. 1.5.: Overview of research methods and research questions.

2 Related Work

In order to provide a common understanding, we outline the theoretical foundations and related literature that are relevant to our work. First, Section 2.1 discusses the role of Artificial Intelligence (AI) and machine learning from the perspective of the IS literature and also introduces a wide range of application examples. Second, we discuss the consequences and challenges of dynamic environments on the application of machine learning based on CS literature. Therefore, the special characteristics of machine learning algorithms in data stream settings are explained (Section 2.2). Eventually, Section 2.3 introduces the definition of concept drift, illustrates various concept drift handling algorithms as well as common data sets for evaluation and discusses novel and open research directions.

2.1 AI and Machine Learning in IS

This section provides a short overview on the development of AI in general as well as in IS specifically. Subsequently, we highlight the main research interests of IS regarding this topic and explain the difference between AI and machine learning. This is followed by an overview of different application cases. Lastly, we elaborate on current research areas.

The term *artificial intelligence* was first popularized at a conference held in 1956 at Dartmouth College in the US, which connected researchers on various topics, including language understanding and self-improvement of machines (McCarthy et al., 2006). In the field of IS, AI has been discussed from the 1980s onwards, and the overall increase of AI applications recently has also spurred considerable research efforts (Berente et al., 2019). However, sometimes it is difficult to follow the progress because the terminology for AI applications has been fragmented, since AI can be divided into many facets and subfields (Kühl et al., 2019). For instance, some popular terms for describing different fields of AI are expert systems, decision support systems, big data analytics, data mining, and machine learning, with IS contributing to each of those subfields (Nascimento et al., 2018).

Besides the technical implementation details, researchers are increasingly emphasizing AI as an object of broader research, as suggested by Shmueli and Koppius (2011). In line with this general trend, the maturation process of AI solutions is investigated apart from singular proofs-of-concept: Popovič et al. (2018) elaborate on the maturity of AI systems and Pappas et al. (2018) stress the importance of broader business analytics ecosystems as a pathway for successful digital transformation. In order to enable such transformations, data is a prerequisite as it is an absolute necessity for any kind of AI application. The origin of data, e.g., open data (Enders et al., 2020) and its value (Günther et al., 2017) are the subject of frequent discussions in the IS community. The new perspective of data as a central resource has also led to the broader discussion of data-driven vs. theory-driven research in IS (Maass et al., 2018).

Due to the fundamental changes driven by AI, several researchers also stress the need to adapt current management guidelines for an efficient handling of corresponding challenges and opportunities (Berente et al., 2019). Since the introduction of AI into organizations is a difficult task, researchers are increasingly evaluating different factors such as technological, environmental and organizational aspects for AI adoption (Alsheibani et al., 2018; Pumplun et al., 2019). Jöhnk et al. (2020) develop a set of AI readiness factors within five categories: resources, knowledge, culture, data, strategic alignment. These allow to assess whether organizations are ready for AI adoption.

Based on data from different contexts, it has been shown that AI can provide significant benefits for decision-making (Meyer et al., 2014). In line with this finding, researchers agree that AI contributes to value creation (Wodecki, 2019). The role of the human in this process, however, is a controversial issue. There are three different levels of collaboration: 1) AI assisting the human (Terveen, 1995), 2) vice versa (Holzinger, 2016), or 3) both entities collaborating on an equal basis in a hybrid model (Seeber et al., 2020). The associated role changes are of major interest for the research community, with first studies showing empirical results (Demetis & Lee, 2018).

In this context, AI-based digital assistants such as chatbots or voice-based assistants play an important role. Those assistants can support humans by relieving them from routine tasks, e.g., by taking over redundant tasks in customer service. This frees up time and resources of the employees for other, more demanding tasks (Maedche et al., 2019). Besides technological barriers, IS researchers have investigated how the design of digital assistants, e.g., by imitating human-like features, behaviors and characteristics, affects the interaction quality between the digital assistant and the

user (Benlian et al., 2020). Furthermore, such systems have been tested in different application areas with specific adaptations, e.g., for supporting financial investment decisions with robo-advisory (Adam et al., 2019). Since those AI-based systems increasingly influence decisions with far-reaching consequences, transparency into their decision-making procedures is an important evaluation criterion (Rzepka & Berger, 2018). In this context, establishing trust in AI systems is crucial. However, the creation of trustworthy AI systems is associated with various challenges along the different stages (e.g., input, model, or output stage) of such systems (Thiebes et al., 2020).

For a better understanding of the subject, it is crucial to elaborate on the difference between AI and machine learning. AI applies a broad range of different techniques to mimic intelligent behavior in machines, whereas machine learning is just one of the techniques applied (Kühl et al., 2019). Machine learning in turn can be divided into three different paradigms (Jordan & Mitchell, 2015): 1) *Supervised learning*, where a system learns a mapping $f(x)$ based on a set of input features x and corresponding labels y, 2) *unsupervised learning*, which usually involves the analysis of unlabeled data to identify underlying structural properties of the data, e.g., by identifying groups in the data, and 3) *reinforcement learning*, where the system learns to improve its actions by receiving different types of rewards. In general, supervised machine learning is the most popular paradigm (Jordan & Mitchell, 2015). This finding is also confirmed for IS research by Nascimento et al. (2018) who reveal that IS researchers mainly apply different classification or regression techniques—both subgroups of supervised machine learning.

In fact, an extant body of research in the discipline of IS deals with the application of machine learning, e.g., case studies showing its feasibility and efficiency. Typical examples for applying machine learning to novel application contexts include, among others, optimizing business operations by forecasting customer churn (Baumann et al., 2015; Coussement et al., 2017; De Caigny et al., 2018) or detecting fraud (Abbasi et al., 2012; Dong et al., 2018). Other studies describe its application for predicting business processes without applying an explicit process model (Evermann et al., 2016), or for optimizing human resources planning (Stein et al., 2018). In e-commerce applications, machine learning techniques have been applied for predicting conversion from users to buyers in e-commerce (Ding et al., 2015; Koehn et al., 2020) or for delivering personalized recommendations (Guo et al., 2018) as well as optimizing the sales promotion strategy (Wang, Li, et al., 2020). In the context of social media, researchers use machine learning for detecting clickbait on social media (Kadian et al., 2018) or identifying cyberbullies (Ptaszynski et al., 2019).

In industrial settings, predictive maintenance scenarios are also investigated, e.g., by predicting road defects (Chatterjee, Saeedfar, et al., 2018) or by predicting component failures for agricultural machines (Lüttenberg et al., 2018). Other applications are instantiated within the energy sector, for improving oil price forecasting (Tripathi & Kaur, 2018) or optimizing redispatch in energy transmission grids (Staudt et al., 2018).

There are also numerous examples of IS researchers applying machine learning for improving healthcare services, e.g., by monitoring diabetes patients (Chatterjee, Byun, et al., 2018), enhancing the analysis of chest X-rays (Rädsch et al., 2021), or forecasting the course of cardiovascular diseases (Brahma et al., 2020), as well as predicting hospital readmission risk for patients with different diseases (Xie & Zhang, 2018). Furthermore, Öksüz et al. (2018) predict therapy success for obese children, and Bahja (2018) extract patient experience and satisfaction with sentiment analysis on online reviews.

However, recent progress in terms of machine learning performance is mainly driven by the advances in deep learning. Unfortunately, the high prediction performance of deep learning methods is accompanied by high complexity and low interpretability (Lipton, 2018). Therefore, recent research focuses on different explainability methods available to make these systems more interpretable (Heinrich et al., 2019). This also allows to investigate the effect of different explainability methods on users, e.g., by measuring the impact of different explanations techniques on users' confidence and overall effectiveness of the decision process (Wanner, Herm, et al., 2020). Furthermore, Wanner, Heinrich, et al. (2020) examine the relationship and prioritization between explanation method, prediction accuracy and implementation effort.

The previous paragraphs highlight the broad range of machine learning systems within IS research. Due to their maturity, an increasing share of these systems is also adopted for support in real-world organizations. Despite the initial setup and training, this also requires the necessary infrastructure for deployment and integration into the respective information systems. Furthermore, constant monitoring after deployment is crucial for ensuring a reliable prediction quality over time. Therefore, it is necessary to consider those systems in the context of data streams where new data instances continuously need to be processed.

2.2 Learning in Dynamic Environments

This section introduces related work regarding learning and predicting in dynamic environments. Dynamic refers to non-static environments such as data streams where new information arrives over time. Section 2.2.1 introduces research which considers the problem of machine learning in data streams, whereas Section 2.2.2 introduces the research stream of online learning, which focuses on game-theoretic aspects in dynamic settings.

2.2.1 Machine Learning in Data Streams

In traditional applications, machine learning algorithms assume the existence of static batch data which can be used for training and evaluation. However, many problems in real-world applications require the handling of dynamic information. Companies need to be able to perform real-time analytics: They do not only want answers to their queries instantly—the processing of the query must be fast—but also with the most recent data (Bifet et al., 2018). Therefore, there has been a shift in research from batch learning to data stream applications where models and patterns need to be identified in a continuous stream of information (Gaber et al., 2005). Methods for static data sets often cannot be transferred to data streams due to their specific characteristics (Rutkowski et al., 2020), as algorithms should mine continuous, high-volume, open-ended data streams as they arrive (Domingos & Hulten, 2001).

A data stream S can be defined as $S = (S_1, S_2, ..., S_n)$, where $S_1, S_2, ..., S_n$ correspond to the individual data instances. Thus, a data stream is an ordered sequence of data instances with the cardinality $n(S)$ possibly being infinity since the stream is unlimited over time (Ramírez-Gallego et al., 2017). There are various characteristics where streams differ from traditional setups. Instances are not given beforehand but arrive sequentially one by one as the stream moves forward (Ramírez-Gallego et al., 2017). Furthermore, new data instances might arrive at very high speed and also with different time intervals between each other (Gama, 2012). Data volume is another processing challenge as streams can be potentially infinite in size which makes it impossible to store all incoming data (Rutkowski et al., 2020). Often, the processing of a data instance is only possible once at arrival because subsequently it is aggregated for storage reasons (Krawczyk et al., 2017). Additionally, the acquisition of true class labels can be limited due to high cost (Ramírez-Gallego et al., 2017).

Due to the characteristics described above, data stream algorithms need to fulfill certain criteria: Methods need to perform with a certain accuracy despite the simplifications made for the stream setting (Gaber et al., 2005). A low processing time of new instances is crucial since new data instances arrive with high velocity (Bifet et al., 2018). Additionally, memory requirements are an important metric for evaluation of any method since streams can be possibly infinite in size. Therefore, the objective is to achieve "maximum accuracy with minimum time and low total memory" (Bifet et al., 2018, p. 5). Other researchers also refer to specific metrics for distributed data streams such as the required communication bandwidth (Gama, 2012).

For handling the complexity of data streams, either *data-based* or *task-based solutions* can be applied (Gaber et al., 2005). Data-based solutions work by reducing the size of the data that needs to be analyzed. Task-based solutions, in contrast, refer to approximations of traditional algorithms for batch data. By allowing broader error bounds and, therefore, less accurate models, fast and efficient solutions for data streams can be achieved. Figure 2.1 depicts an overview of different solutions for machine learning in data streams.

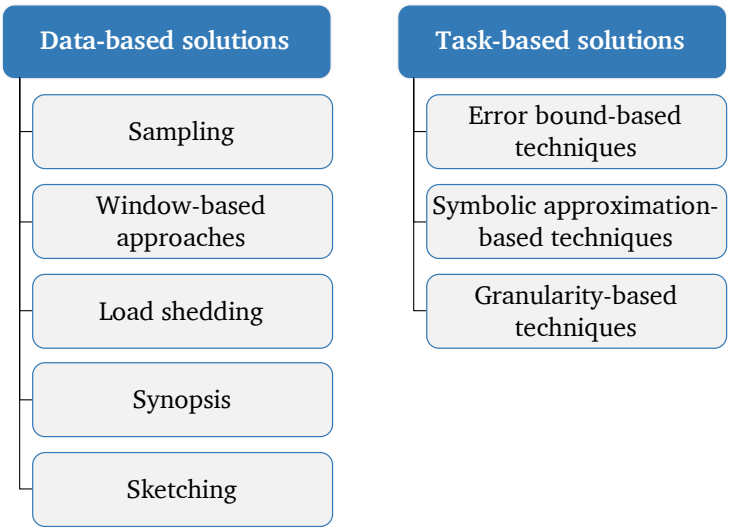


Fig. 2.1.: Overview of solutions for machine learning in data streams.

Sampling is one data-based technique where only a subset of the data—based on certain criteria, e.g., a random selection—is processed. The other data instances are discarded, which naturally reduces the computational load and costs (Gama, 2012). Samples can either be drawn by simple random sampling or by applying reservoir sampling (Aggarwal, 2006). In reservoir sampling, a sample of size k—the reservoir—is kept. Every new element in the stream has the probability of k/n to replace an old element in the reservoir, which means that the probability of

inclusion in the reservoir decreases with increasing data stream size n. In this context, *window-based approaches* play an important role. Babcock et al. (2001) propose two different methods: sequence-based and timestamp-based windows. In a sequence-based window, the size of the window is defined by the number of observations considered. In contrast, timestamp-based windows are described based on the duration of a window. All data instances with a timestamp within the time interval of the window are selected. *Load shedding* is similar to sampling but drops whole sequences of incoming data instances. This might be necessary when the input data speed exceeds the capacity of the system. Load shedding can be applied to prevent system overload and large latency (Tatbul et al., 2003). *Synopsis* works by summarizing the incoming data stream, e.g., with wavelet transformations (Gaber et al., 2005) or aggregating information by computing means and variances (Aggarwal et al., 2003). *Sketching*, in contrast, works with analyzing a subset of features only (Muthukrishnan, 2005). For instance, Ramírez-Gallego et al. (2017) suggest applying feature extraction algorithms such as Principal Component Analysis in this context.

Task-based solutions refer to approximations of static methods for the online setting of data streams. One strategy for computing approximated solutions is represented by *error bound-based techniques* which have the objective of increasing the training speed of models. This strategy is usually based on the concept of a Hoeffding bound (Hoeffding, 1963), which allows to estimate the probability that a model deviates by a certain amount from its optimal parameter configuration (Hulten et al., 2001). For time series data, *symbolic approximation-based techniques* have been introduced (Lin et al., 2003). Symbolic methods reduce computational complexity of determining an appropriate model by applying piecewise aggregate approximation functions in combination with symbolic values. Lastly, *granularity-based techniques* refer to a set of machine learning methods that can adapt their resource consumption patterns over time (Gaber, 2012). For instance, algorithm processing granularity refers to the process of adapting the parameters of an algorithm to reduce the required processing power.

A wide range of machine learning tasks such as clustering, classification, frequent pattern mining as well as time series analysis have been considered in the context of data streams (Gaber et al., 2005; Gama, 2012). In literature, many different methods are proposed for handling these tasks: instance-based classifiers (k-nearest neighbors), Bayesian classifiers (Naive Bayes), ANNs (multilayer perceptron), decision trees (very fast decision tree) (Domingos & Hulten, 2000) and ensemble techniques (Rutkowski et al., 2020).

The previous paragraphs highlight the complexity of data stream mining. Various challenges need to be considered such as the cost-performance management regarding incremental learning and forgetting (Kifer et al., 2004). Intuitively, there is a trade-off between high prediction performance and the corresponding cost: Frequent model updates lead to better prediction performance but also result in higher cost due to the increased computing power required. Furthermore, data streams are challenged with changing probability distributions over time, which might endanger the validity of the applied methods (Rutkowski et al., 2020). In this context, especially evolving features, e.g., new words over time in a text mining task, are also challenging (Masud et al., 2013).

2.2.2 Online Learning

Closely related to machine learning in data streams is the research stream of *online learning*. However, online learning specifically follows the supervised machine learning paradigm (predict a label for a new data instance and then receive the corresponding true label for this prediction), whereas machine learning in data streams (Section 2.2.1) covers a broader range of techniques (e.g., also unsupervised machine learning). Since this thesis puts a strong focus on supervised machine learning tasks, we briefly introduce the fundamentals of online learning in the following.

Online learning can be explained as making a series of predictions with knowledge regarding the correctness of previous predictions (Shalev-Shwartz, 2011). In this sense, online learning is very similar to machine learning in data streams due to the dynamic nature of information processing. However, research around this topic is mainly driven by a separate research community in computer science. In contrast to other research streams, online learning investigates the dynamics of data streams from a game-theoretic perspective. In this context, online learning considers the problem of computing predictions over time as a game, and any kind of entity capable of issuing predictions (e.g., a machine learning model or a human expert) is referred to as player. In its simplest form, online learning considers one player only, but there are also applications with several players.

Online learning is played in rounds. A player receives a question (a new data instance x_t) and is required to answer this question (perform a prediction p_t). Subsequently, the true answer is revealed (receive true label y_t). Based on this information, the loss $l(p_t, y_t)$ that a player is suffering is computed (Beyazit et al., 2019). This loss also triggers the corresponding learning process. Subsequently, the next round

Algorithm 1 Principle of online learning (Shalev-Shwartz, 2011)

1: **for** $t = 1, 2, ...$ **do**
2: receive question $x_t \in X$
3: predict $p_t \in D$
4: receive true answers $y_t \in Y$
5: suffer loss $l(p_t, y_t)$
6: **end for**

of the game is started. The game-theoretic aspect is induced by considering the iterative nature of the problem setup. A player receives feedback about her previous predictions and can adapt her behavior based on strategic decisions to minimize her loss. The fundamental principle of online learning is depicted in algorithm 1.

Applications in online learning can be differentiated based on certain characteristics. In some cases, additional knowledge in the form of other input features can also be included for predictions. In other cases, a player receives only partial information about the correctness of previous answers. Online classification is one of the main problems in this research stream, where predictions and answers can only be either yes or no. In this case, the two spaces for possible predictions D and true labels Y are equivalent: $D = Y = \{0, 1\}$ (Shalev-Shwartz, 2011). An example for this setting is predicting whether it will rain tomorrow. Today, the learning algorithm receives meteorological information in a vector x_t (e.g., containing temperature and humidity). Based on this vector, the learning algorithm predicts whether it is going to rain tomorrow. The next day, the algorithm receives feedback and knows the true answer. In the end, the objective of the learner is to minimize the total cumulative loss over all time steps t.

Apart from online classification, examples for online learning are online regression, where the objective is to predict a real-valued target (Moroshko et al., 2015). Other approaches include prediction with expert advice where a player needs to select an expert which she trusts, as well as online ranking where an item set needs to be ordered according to its relevance for a player. Furthermore, a well-known problem in this domain is the multi-armed bandit problem which is inspired by a gambler that needs to decide which arm of various slot machines she needs to pull to maximize her reward in several trials (Vermorel & Mohri, 2005). Each round, the player receives a loss according to the chosen arm, but she does not receive any information about the potential loss that she would have suffered by playing a different arm (Agrawal & Goyal, 2012). Due to its game-theoretic approach, online learning also considers adversarial learning cases, e.g., where an adversary intentionally delays feedback for a prediction (Quanrud & Khashabi, 2015).

Besides the introduced different research streams, various emerging problems are currently investigated. For instance, Beyazit et al. (2019) examine the effect of varying features within online learning settings, e.g., cases where some features disappear and other features might emerge over time. Other researchers study the incentives of experts in such an online scenario, e.g., regarding the behavior of pollsters in elections (Roughgarden & Schrijvers, 2017). Further variants of online learning algorithms can be found for the domain of deep learning representation in graph analytics (Perozzi et al., 2014) or for topic modeling in text mining based on latent Dirichlet allocation (Hoffman et al., 2010).

Both research streams—either online learning or machine learning in data streams—require models to generate ongoing predictions over time. In this context, machine learning models need to be able to react to changing data distributions while keeping a high prediction performance. Therefore, we introduce research regarding concept drift which describes solutions for this problem domain in the following Section 2.3.

2.3 Concept Drift

This section contains research related to concept drift. Section 2.3.1 introduces relevant definitions of concept drift and also explains different concept drift types. The following Section 2.3.2 describes various algorithms for concept drift detection and also explains how machine learning models can be adapted in case of concept drift. Furthermore, it gives an overview on relevant evaluation strategies and popular benchmarking data sets. Finally, Section 2.3.3 introduces several application examples of concept drift handling and also elaborates on novel research areas.

2.3.1 Definition

Concept drift refers to changing data distributions over time and is, therefore, a challenge when dealing with data streams because prediction models learned on old data are not able to perform well on new data (Tsymbal, 2004). A concept is defined as the common probability distribution $P(X, y)$ of a set of input features X and a target y (Webb et al., 2016). This definition reveals that the problem is mainly considered in supervised machine learning settings. In general, concepts depend on the context of the respective data stream which is often hidden from the machine learning model (e.g., important factors that are not included in the input features of

the model). Therefore, changes in the context cannot be observed by the machine learning model, leading to challenges for computing correct predictions (Widmer & Kubat, 1996). A good example for concept drift are changing customer preferences over time. In this context, a system for predicting the buying behavior of customers is applied. Suddenly, one of the customers gets a significant pay raise which is not observable for the algorithm (changing hidden context). Due to her changed income, the customer adapts her shopping behavior, e.g., by buying more organic products, which makes high-quality recommendations for the machine learning model difficult.

Formally, *concept drift* is defined in the following way (Gama et al., 2014; Widmer & Kubat, 1996):

$$P_{t_0}(X, y) \neq P_{t_1}(X, y).$$

Here, t_0 and t_1 are two different points in time with $t_1 > t_0$. Besides concept drift, the machine learning community also uses other terms to describe similar phenomena referring to changing data distributions (Moreno-Torres et al., 2012). For instance, *dataset shift* (Quionero-Candela et al., 2009) is described as a change in the common probability distribution of input data X and corresponding labels y between training (tr) and test time (tst):

$$P_{tr}(X, y) \neq P_{tst}(X, y).$$

For the differentiation between the two definitions, the indices are crucial: Dataset shift focuses on the difference between training and testing environment, whereas concept drift refers to the temporal aspect of data and is, therefore, closely linked to the problem of machine learning in data stream settings.

There are further subcategories of concept drift, where *covariate shift* refers to changes in the distribution of the input data X only, without affecting the distribution of labels: $P_{t_0}(X) \neq P_{t_1}(X)$ and $P_{t_0}(y|X) = P_{t_1}(y|X)$ (Moreno-Torres et al., 2012). This change is also referred to as *virtual drift* in the concept drift literature (Gama et al., 2014). Furthermore, *real concept drift* describes any changes in $P(y|X)$, independent of whether this change is triggered by changes in $P(X)$ or not. A subtype of real concept drift is called *label shift*, *concept shift* or *conditional change* (Gao et al., 2007; Moreno-Torres et al., 2012): $P_{t_0}(X) = P_{t_1}(X)$ and $P_{t_0}(y|X) \neq P_{t_1}(y|X)$. In this case, it is only the distribution of the labels given X that changes over time, whereas the distribution of the input features alone remains constant.

For purpose of illustration, an information system recommending sports products is considered. The task of the system is to classify sports products as *relevant* or not *not*

relevant to a given customer. In the beginning, the customer is interested in running shoes. Therefore, any products related to running shoes are *relevant* and bicycle equipment is *not relevant*. The following year, new running shoes with different characteristics are available for sale. Despite changed characteristics, those shoes are still relevant to the customer. This is an example of *virtual drift*. However, if the customer has bought a pair of running shoes and additionally wants to exercise on a bicycle, running shoes become *irrelevant* and bicycle equipment becomes *relevant*. This represents a scenario of *real concept drift*.

The relationship between virtual and real concept drift is also depicted in Figure 2.2. A dot in the figure represents a data instance and different colors refer to different class affiliations. In case of virtual drift, the decision boundary compared to the original data remains the same whereas the input data X changes. Therefore, no adaptation of the machine learning model is required. In contrast, in case of real concept drift, the decision boundary of the machine learning model needs to be adapted to represent the new class affiliations. In this specific case, the concept drift depicted represents a label shift since the distribution of the input data X remains the same.

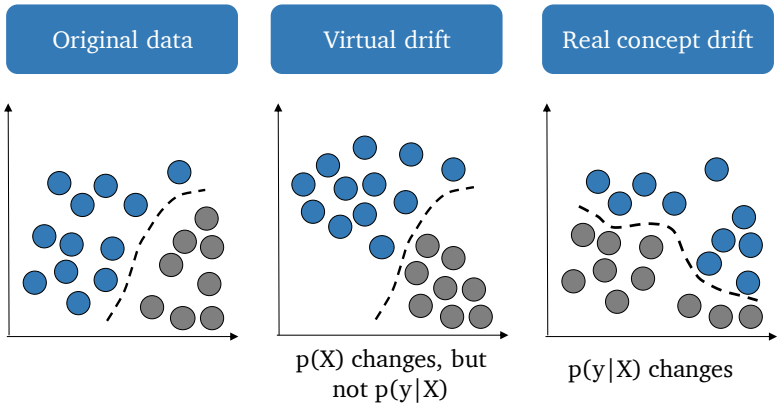


Fig. 2.2.: Virtual and real concept drift, adapted from Gama et al. (2014).

In statistics, the problem of changing data is also investigated under the terms *structural breaks in time series* (Aue & Horváth, 2013) or *detection of change points* (Antoch et al., 2019). In this context, it is common to estimate several structural breaks simultaneously from hindsight (Bai & Perron, 2003). Often, those statistical models are applied in economic contexts, e.g., for detecting structural breaks in financial time series due to economic and political events such as the introduction of a single European currency (Antoch et al., 2019).

Besides different definitions of concept drift, researchers also differentiate various structural types of concept drift, depicted in Figure 2.3. In this simplified case, it

is assumed that the data is just one-dimensional. In total, there are four different types (Žliobaitė, 2010): Sudden or abrupt concept drift, incremental and gradual concept drift as well as reoccurring concepts.

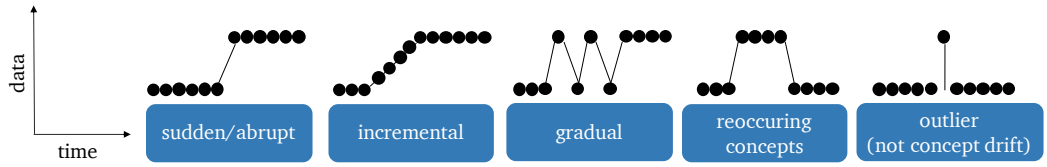


Fig. 2.3.: Overview of structural types of concept drift, adapted from Žliobaitė (2010).

Sudden concept drift refers to sudden changes in the data distribution. This can be triggered for instance by the replacement of a sensor with a sensor with a slightly different calibration in a manufacturing environment (Gama et al., 2014). Another example is the sudden popularity of masks in e-commerce shops during the beginning of the COVID-19 pandemic. Incremental concept drift is characterized by slower changes, often referring to changes in a whole population (e.g., people increasingly buying organic products) (Žliobaitė et al., 2016). An example for gradual drift is a user changing her hobbies over time. At first, she mainly reads articles about finance and then she also gets interested in sports articles. She keeps coming back and reading finance articles but in the end, she stays with sports (Žliobaitė, 2010). Reoccurring concepts are best characterized by seasonal changes, e.g., changing sales of ice cream or air conditioning in summer and winter (Ramamurthy & Bhatnagar, 2007). Note that reoccurring concepts go beyond simple periodic seasonality (Žliobaitė, 2010): In the ice cream example, the peak of sales will be slightly different each year depending on the precise weather profile in this specific year as well as other factors. Outliers, in contrast, should not be interpreted as concept drift (Tsymbal, 2004). Adjusting to outliers will lead to poor prediction performance of the underlying machine learning model.

Besides these main structural types of concept drift, Webb et al. (2016) provide a more detailed taxonomy of concept drift characteristics. Among others, several features such as drift duration, concept duration as well as drift transition and severity are introduced to further distinguish concept drifts.

2.3.2 Concept Drift Handling

The handling of concept drifts can be divided into two steps: First, the detection of concept drifts and second, the required adaptation of the underlying prediction models. Therefore, we first describe different algorithms for concept drift detection

both from the machine learning as well as the statistics community (paragraph *Detection*). Subsequently, the paragraph *Adaptation* explains how machine learning models can be adapted in case of concept drift and illustrates a set of adaptive learning strategies. Lastly, the paragraph *Performance Evaluation* gives an overview on relevant evaluation metrics as well as popular benchmarking data sets.

Detection

An important feature of any machine learning system deployed in data stream settings is the ability to detect concept drift. Therefore, literature provides a wide variety of concept drift detection methods. Lu et al. (2019) define three different categories of algorithms: error rate-based drift detection, data distribution-based drift detection and multiple hypothesis test drift detection. Popular representatives for each category are displayed in Table 2.1 and are explained briefly in the following paragraphs.

Tab. 2.1.: Different categories of drift detection algorithms and popular representatives.

Error rate-based drift detection	Data distribution-based drift detection	Multiple hypothesis test drift detection
• DDM (Gama et al., 2004)	• PCA-CD (Qahtan et al., 2015)	• JIT (Alippi & Roveri, 2008)
• PH (Page, 1954)	• KSWIN (Raab et al., 2020)	• HLFR (Yu & Abraham, 2017)
• ADWIN (Bifet & Gavaldà, 2007)	• ITA (Dasu et al., 2006)	• TMSD (Raza et al., 2015)

The first category, *error rate-based drift detection* forms the largest category of detection algorithms (Lu et al., 2019). They monitor the error rate of a prediction model and if a significant increase or decrease in the error rate is detected, a drift alarm is triggered. Methods include DDM, PH and ADWIN.

The Drift Detection Method (DDM) (Gama et al., 2004) assumes that the error of a classifier can be modeled as a random variable from Bernoulli trials where the classifier either makes a correct prediction (no error) or a false prediction (error). The probability distribution of several prediction errors in turn can be modeled via a binomial distribution. If the error rate deviates from the expected error rate to a certain extent, a drift is likely to have happened. There are various other concept drift detection algorithms which are based on the same principle (Lu et al., 2019).

The Page-Hinkley (PH) algorithm (Page, 1954) implements the idea of a cumulative sum, a sequential analysis technique. The algorithm computes a test statistic which

monitors the cumulative difference between the mean and the observed value of a variable over time. Usually, this variable is the error rate of the prediction model. Concept drift is detected if the difference is larger than a predefined threshold set by the user.

Adaptive Windowing (ADWIN) (Bifet & Gavaldà, 2007) is another popular approach which works by analyzing a window of recent observations. As in the previously presented methods, the variable that is being monitored is the prediction error. ADWIN cuts this window of recent observations into two windows by computing all possible subwindow combinations. If ADWIN detects a large difference between the mean of the two windows, a concept drift alarm is triggered. The detection threshold is derived by applying the Hoeffding bound.

The second category, *data distribution-based drift detection*, works by monitoring historic as well as current data of the input features X and does not require the prediction error rate. Usually, algorithms in this category apply some kind of distance function to quantify the difference between the distribution of old as well as recent data instances. Methods include PCA-CD, KSWIN and ITA.

As data distribution-based methods are often confronted with high dimensionality, their computation can be difficult. Therefore, Prinicipal Component Analysis-based Change Detection (PCA-CD) (Qahtan et al., 2015) applies PCA to project the data stream to a lower-dimensional space. This enables the computation of density estimations of the distribution between different time windows as well as the computation of change scores. Drifts are detected by feeding change scores into a PH algorithm.

Kolmogorov-Smirnov Window (KSWIN) (Raab et al., 2020), in contrast, is based on computing the Kolmogorov-Smirnov test over windows of past input data. For each input feature in the data stream, KSWIN computes an absolute difference between the distributions of recent instances as well as a representative sample. If there is a significant difference (based on the Kolmogorov-Smirnov test) for one of the features included, KSWIN indicates a concept drift.

The Information Theoretic Approach (ITA) (Dasu et al., 2006) again relies on two different windows of observations. Data instances are sorted into bins and differences between the two window distributions are derived by computing the Kullback-Leibler distance (Kullback & Leibler, 1951). The threshold for change detection is estimated by applying bootstrapping on the empirical distribution of the input data.

Regarding the differences between error rate-based drift detection and data distribution-based drift detection, Hu et al. (2020) note the limitations of data distribution-based methods to detect label shift. Error-rate based drift detection performs better for this type of drift but requires the acquisition of true labels. Therefore, there is a trade-off between performance (higher drift detection accuracy) and cost (acquisition cost for true labels) among drift detection algorithms. The authors describe this as an example of the "no free lunch" theorem, according to which it is impossible to have a single best performing approach with respect to multiple evaluation criteria.

The third category, *multiple hypothesis test drift detection*, represents a group of methods that uses similar techniques as in the two other categories, but it performs several hypothesis tests or methods to improve detection accuracy. For instance, Just-In-Time (JIT) detection (Alippi & Roveri, 2008) combines cumulative sum and PCA techniques in parallel to improve drift detection. In contrast, Hierarchical Linear Four Rate (HLFR) (Yu & Abraham, 2017) performs concept drift detection in a sequential manner. First, it detects drift by monitoring all four rates of the confusion matrix (e.g., true positive rate). Second, when a drift is detected in the first layer, this information is sent to the second layer for confirmation in order to prevent false alarms. Two-Stage Multivariate Shift-Detection (TSMSD) (Raza et al., 2015) works in a similar way. First, an exponentially weighted moving average is applied for drift detection. In case of drift, this information is validated by applying a Kolmogorov-Smirnov test.

As described above, the problem of concept drift is related to the detection of structural changes in time series data in statistics (Verbesselt et al., 2010). The methods applied from the statistics community typically differ in various aspects. Instead of analyzing the prediction error or the distribution of input features, it is common to fit various models on the time series and test whether the parameters of those models differ (Hansen, 2001). The traditional test for identifying structural changes is the Chow-test, which works by splitting the data set to be analyzed in two samples (Chow, 1960). A linear model is fitted for each sample. Subsequently, the two sets of parameters—one set per model—are tested for equality. A structural change has occurred if the parameters differ significantly. However, this method requires an assumed breakpoint date (e.g., based on external information) in advance as input (Zeileis et al., 2003) and then tests whether the structural change at this date is significant (Nielsen & Whitby, 2015). Furthermore, this method is not able to deal with more than one structural break (Dufour, 1982).

Andrews (1993) provides a test that requires the time series as input only and determines the timing of the break date itself. Furthermore, it is possible to estimate confidence intervals for the possible breakpoint date which gives an indication for the estimation accuracy of the method (Bai, 1994). Nevertheless, those methods can also handle one structural break only. Therefore, Bai and Perron (1998) introduce an iterative method to test for multiple breakpoints. First, the approach tests for one structural break. Second, the time series is split according to the identified break date and the approach continues to test on the two resulting smaller time series.

In general, the application of methods for break detection requires the full input data, i.e., the complete time series (Perron, 1989). Furthermore, since those methods basically rely on estimating parameters for two different models, their application is difficult if a break is near the end or the beginning of a time series since there may not be sufficient information to estimate the parameter values of the linear model (Lai, 1995).

Besides fitting two different models, the development of monotonic trends can also be analyzed for detecting changes in time series data. One statistical method in this category is the non-parametric Mann-Kendall (MK) test (Kendall, 1948). This test is often applied in long term data sets such as ecological or meteorological contexts (Sonali & Kumar, 2013). The idea of the MK test is to investigate whether time series observations follow a monotone trend over time or whether they are ordered randomly. Furthermore, the analysis of stochastic trends can be performed by testing for unit roots (White & Granger, 2011). This allows to analyze the characteristics of the time series by investigating whether the time series is stationary, trend stationary or has a unit root (Haldrup et al., 2013). Unit root time series follow a random walk model. This implies that random shocks have a lasting effect on the trend of a time series. In contrast, in a trend stationary model, the time series reverts back to the previous trend some time after the shock (Hansen, 2001). For instance, those methods can be used for analyzing the effect of external shocks on stock markets (Kremser et al., 2019). However, unit root tests need to consider structural breaks because otherwise inference for time series predictions can be misleading (Perron, 1989; Zeileis et al., 2003).

Adaptation

The previous paragraph *Detection* introduces a broad range of algorithms for detecting changes in data streams. However, one important aspect of concept drift handling is not only the detection of drifts but also how to adapt the underlying

machine learning model in case of change. In general, the model might be adjusted whenever new labeled instances are acquired. According to Gama et al. (2014), there are two different learning modes available: retraining and incremental updates. Retraining describes training a new model from scratch by accessing buffered data with labels and discarding the current model. In contrast, incremental approaches update the current model by considering the most recent data instances (Žliobaitė, 2010). Relevant parameters of the underlying model are continuously adapted, e.g., the weights of a neural network or a logistic regression.

Figure 2.4 shows an overview of four different classes of adaptive learning strategies (Žliobaitė et al., 2016). One relevant dimension is the model form: This describes whether a single machine learning model or an ensemble of multiple models is applied. The other dimension refers to the initiation of model adaptations. Adaptations can either be based on triggers such as explicit concept drift detection algorithms or are carried out regularly on an evolving level. Triggered adaptation strategies are also described as informed or active strategies, whereas evolving strategies are considered as blind or passive (Ditzler et al., 2015; Khamassi et al., 2018).

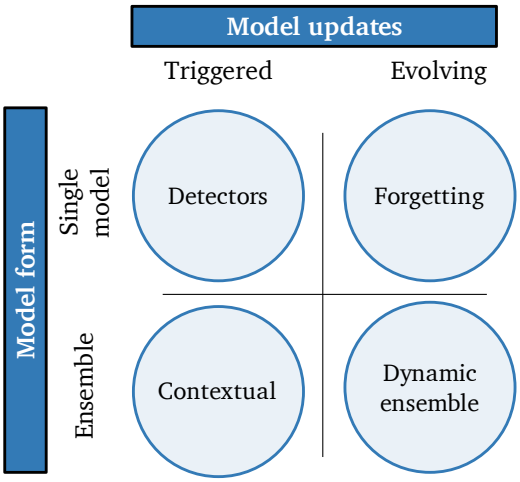


Fig. 2.4.: Adaptive learning strategies, adapted from Žliobaitė et al. (2016).

The *detectors* strategy uses a single prediction model and is based on a drift detection algorithm (e.g., ADWIN). In case of a drift alarm, the machine learning model is adapted. It is often a suitable strategy for handling sudden concept drift (Žliobaitė, 2010). The *forgetting* strategy, in contrast, does not apply a specific drift detection algorithm. Instead, it uses a sliding window of fixed size over past data instances and is periodically retrained with this data (Žliobaitė et al., 2016). Regarding ensemble model strategies, the *contextual* strategy relies on triggered adaptation. It uses available information for assigning the most relevant model contained in the ensemble for decision making (Žliobaitė et al., 2016). A good example is food

sales prediction, where products are differentiated into different sales groups based on certain characteristics with distinct prediction models each (Žliobaitė et al., 2012). The last strategy, *dynamic ensemble*, refers to ensembles that evolve over time. Usually, the decision rule determining how the different models are combined (e.g., different models receive different weights) is dynamically updated over time based on the performance of the individual machine learning models (Minku et al., 2009). Due to their characteristics, ensemble strategies are especially suitable for handling reoccurring concept drift (Lu et al., 2019).

Different forms of ensemble methods for concept drift handling can also be distinguished based on other dimensions (Krawczyk et al., 2017), for instance by considering structural change. In this context, structural changes lead to the complete replacement of individual machine learning models in an ensemble over time (Kuncheva, 2004).

Within the supervised machine learning domain, concept drift handling for classification problems represents the predominant share of approaches (Hu et al., 2018; Iwashita & Papa, 2019; Jaworski, 2018). Ditzler et al. (2015) even provide an entire literature review focused on classification task only. This focus can also be explained by the fact that many drift detection algorithms exploit the characteristics of the binomial error distribution in classification tasks (Gama et al., 2014). However, many tasks need to be modeled as regression problems (Harrington, 2012), but only very few approaches tackle the concept drift problem in this setting (Song et al., 2019). Compared to classification, regression and especially time series forecasting problems have certain characteristics that are relevant for concept drift analysis (Cavalcante et al., 2016). For instance, Cavalcante et al. (2016) compute a set of eight time-series specific metrics, such as autocorrelation and partial autocorrelation, to identify concept drift in a time series context. Another approach by Song et al. (2019) identifies historical patterns which are likely to reappear based on clustering techniques for improved concept drift handling.

Performance Evaluation

For evaluating different concept drift handling strategies, various metrics can be applied. One set of metrics refers to the concept drift detection itself, where the mean time to detection, mean time between false alarms and missed detection rate are popular examples (Bifet, 2017). The mean time to detection, or delay of detection, describes the average duration until a drift is detected (Gama et al., 2014). The missed detection rate refers to the probability of not receiving an alarm

even though concept drift has occurred (Bifet, 2017). The mean time between false alarms is an indicator of the number of false alarms where the presence of concept drift is falsely assumed (Sethi & Kantardzic, 2017). However, all those metrics require the knowledge of the true drift point (Gama et al., 2014). Real-world data sets usually do not contain information about the precise start and end of drifts since drifts in reality are often influenced by hidden factors that cannot be measured (Gonçalves et al., 2014; Lu et al., 2019).

Therefore, it is a popular approach to compare the overall predictive accuracy of different drift handling strategies (drift detection algorithm + prediction model) on real-world data sets (Elwell & Polikar, 2011; Gonçalves et al., 2014; Souza et al., 2020). Metrics such as accuracy, Area Under the Curve (AUC) or F1-score are applied for classification tasks, and the mean absolute error for regression tasks (Gonçalves et al., 2014; Song et al., 2019). Besides the detection accuracy, the required computational resources such as RAM-hours (Bifet et al., 2010), the necessary overall computational time (Gonçalves et al., 2014), or the decision time for a single instance (Krawczyk et al., 2017) can be also considered as evaluation metrics.

An additional level of evaluation for real-world data sets can be implemented by considering relevant baseline strategies. The *static baseline* (Lindstrom et al., 2013; Sethi & Kantardzic, 2017) model refers to a model that is held constant over time. Therefore, no change is considered, and the model is never updated over the course of the data stream. This is a lower-bound baseline which all drift handling strategies should outperform. For classification problems, two other baselines can be implemented: First, a naive predictor which always predicts the majority class based on a moving window over time (Bifet, Read, Žliobaitė, et al., 2013). Second, a no-change predictor that predicts the next class label in the data stream to be equal to the class label of the previous data instance (Bifet, Read, Žliobaitė, et al., 2013; Souza et al., 2020). A similar baseline can be implemented for time series problems, where a random walk model predicts the next data instance to have the same value as the last data instance (Gama et al., 2014).

For the evaluation of concept drift handling strategies, there are several simulated or synthetic data sets available with different types of concept drift included. One way to induce drift consists in changing the classification function or by manipulating the input features over time. A good overview on different simulated data sets is given by Bifet et al. (2009). Table 2.2 gives an overview on popular synthetic as well as real-world data sets.

Real-world data sets for concept drift evaluation range from the prediction of electricity prices (Electricity) to network intrusion detection (KDDCUP99) as well as airline delay prediction (Airlines) and prediction of winning chances in games (Pokerhand). Those data sets include different types of drift, among others daily, seasonal, yearly or geographical concept drift (Lu et al., 2019). A detailed overview and corresponding explanations on the data sets introduced in Table 2.2 as well as further examples are given by Souza et al. (2020).

Tab. 2.2.: Popular synthetic and real-world data sets for concept drift evaluation.

Synthetic	**Real-world**
• SEA (Street & Kim, 2001) • STAGGER (Schlimmer & Granger, 1986) • Rotating Hyperplane (Hulten et al., 2001) • Random RBF (Bifet et al., 2009)	• Electricity (Harries, 1999) • KDDCUP99 (Tavallaee et al., 2009) • Airlines (Ikonomovska et al., 2011) • Pokerhand (Cattral et al., 2002)

Despite the existence of the data sets described above, real-world evaluation of concept drift handling strategies is still confronted with a number of challenges: First, there is usually no information given regarding the exact drift time as well as drift type (Gonçalves et al., 2014; Lu et al., 2019). Second, various researchers have commented on a shortage of appropriate real-world data sets (Bifet et al., 2009; Krawczyk et al., 2017; Nguyen et al., 2015; Souza et al., 2020). Third, class distributions in real cases are often highly imbalanced (Hoens et al., 2012). Fourth, in many real-world applications, the acquisition of true labels can either be very expensive or simply impossible (Hu et al., 2020; Lindstrom et al., 2013). Therefore, a large share of concept drift algorithms cannot be applied in this setting. Fifth, some of the real-world data sets, e.g., the Electricity data set, have been questioned regarding their applicability for concept drift analysis altogether (Žliobaitė, 2013).

To mitigate some of those problems, researchers have taken real-world data sets and artificially induced concept drifts (Sethi & Kantardzic, 2017; Sobolewski & Wozniak, 2013). Some common approaches to simulate changes in real data with static distributions are manipulating input features (Ramamurthy & Bhatnagar, 2007) or joining classes (Vreeken et al., 2007).

2.3.3 Applications and Current Research

There are examples of concept drift handling strategies for improved predictions in different industry domains. In the energy sector, concept drift handling methods are applied for the detection of wind power peaks (Tomin et al., 2015) or for improved prediction of photovoltaic power production (Ceci et al., 2019). In a production context, researchers utilize those methods to optimize a predictive maintenance application for industrial radial fans (Zenisek et al., 2019), to set up a visual quality control system on weld defects (Mera et al., 2019), and to detect intruders in industrial control systems (Zizzo et al., 2019). Furthermore, Saadallah et al. (2020) develop a system capable of handling different concept drift types for providing more reliable mobility demand predictions. In the IT domain, the identification of spam e-mails over time (Ruano-Ordas et al., 2018; Sheu et al., 2017), credit card fraud detection (Dal Pozzolo et al., 2017; Somasundaram & Reddy, 2019) or churn prediction (Machado & Ruiz, 2017) are all subject to concept drift investigation. The same applies to the optimized analysis of event logs (Seeliger et al., 2017), also in combination with human-in-the-loop systems (Barbon Junior et al., 2018). In security research, ageing has been investigated under the notion of concept drift for facial recognition tasks (Akhtar et al., 2015). Generally, Žliobaitė et al. (2016) differentiate use cases with concept drift handling in three categories, namely monitoring and control, information management, and analytics and diagnostics. They also provide a more detailed overview on characteristics of each category.

Most research papers in the concept drift literature focus on developing methods for concept drift detection (Webb et al., 2018). However, recently, more work has also been dedicated towards understanding and explaining concept drift. Lu et al. (2019) differentiate into three different categories of drift understanding: the timing of concept drift, the severity of concept drift, and the drift regions of concept drift. The *timing of concept drift* is a piece of information which is basically provided by all concept drift detection algorithms and some algorithms even provide a warning window (Gama et al., 2004). In this context, Webb et al. (2016) propose to measure the drift and concept duration. The *severity of concept drift* can be measured by considering the drift magnitude which can be determined by measuring the distance between drifting distributions (Webb et al., 2016).

Most of the recent research advances have focused on providing novel solutions for identifying the *drift regions* or the input features responsible for concept drift. For this reason, Demšar et al. (2014) develop a method for visualizing model explanations based on Shapley values in a data stream setting. This method is applied for detecting concept drift which also allows to provide interpretable visualizations

of concept drift (Demšar & Bosnić, 2018). Other researchers use methods from counterfactual explanations to better understand which features are responsible for concept drift (Hinder & Hammer, 2020). Concept drift mapping (Webb et al., 2018) analyzes concept drifts by considering the marginal distributions over different combinations of input features. This allows for a detailed description of the type and form of concept drift. Furthermore, visualization techniques have been combined with established drift detectors for improved understanding and examination of concept drift (Wang, Chen, et al., 2020), e.g., with the introduction of streaming scatterplots (Yang et al., 2020).

To summarize, concept drift algorithms are already utilized in a broad range of applications. However, most of the research papers either test their algorithms on simulated or proprietary data sets. Therefore, publicly available real-world data sets for evaluation of novel methods are scarce. Furthermore, regression problems only play a minor role in concept drift research even though they represent a significant share of machine learning modeling tasks. Lastly, the unrealistic assumption of immediate true label feedback makes it impossible to utilize most concept drift algorithms in many real-world problems. Therefore, this thesis introduces novel methods and ideas to improve concept drift handling in those specific application settings.

Part II

Choices and Challenges for Machine Learning Applications

3 The Supervised Machine Learning Reportcard[1]

3.1 Introduction

Replication of published research is an important endeavor in the academic world. Replication studies repeat previously conducted studies with the goal to investigate whether the findings are reliable—and to what extent they can be generalized. Over the last decade, a lack of these methodologically important supplements have constituted the so-called "replication crisis"—reflecting that many scientific studies and their results are in fact difficult or even impossible to replicate. So far, this replication crisis has particularly been proclaimed in the fields of medicine and psychology (Schooler, 2014; Tackett et al., 2019).

While IS research has started to actively incentivizing replication studies (Olbrich et al., 2017; Weinhardt et al., 2019), the rise of methods from Machine Learning in IS entail new challenges in replication (Coiera et al., 2018; Hutson, 2018). Especially Supervised Machine Learning (SML) is gaining increasing popularity in the field: Between 2010 and 2018, 35 contributions published in *Management Information Systems Quarterly (MISQ)*, *Information Systems Research (ISR)* and *Journal of Management Information Systems (JMIS)* apply SML in their research. In addition, the number of publications in typical IS conferences (*European Conference on Information Systems (ECIS)*, *International Conference on Information Systems (ICIS)*) that rely on SML as a key method is also steadily growing over time.

While SML is enjoying widespread popularity and promises considerable potential in IS research, there is room for improvement when it comes to rigorously applying these technologies: Many IS research articles lack a thorough documentation of the SML process and the results obtained, which makes it challenging or virtually

[1]This chapter comprises an article that was published as: Kühl, N., Hirt, R., Baier, L., Schmitz, B. & Satzger, G. (2020). How to Conduct Rigorous Supervised Machine Learning in Information Systems Research: The Supervised Machine Learning Reportcard. *Communications of the Association for Information Systems [forthcoming]*. Note: The abstract has been removed. Tables and figures were reformatted, and newly referenced to fit the structure of the thesis. Chapter and section numbering and respective cross-references were modified. Formatting and reference style was adapted and references were integrated into the overall references section of this thesis.

impossible to reproduce or replicate their results. Naturally, researchers may prefer discussing the implications of SML results instead of stringently documenting the SML process itself. This, however, will contribute to spread the replication crisis described above also in the IS research community, as it is neither possible to follow or replicate the precise choices of the research nor to judge whether its results are indeed meaningful.

We set out to address this problem, and develop and test a documentation standard ultimately enabling frequent replication of SML studies in IS. To this end, we first review the literature to identify the typical problem characteristics and choices to be made in SML endeavors. On this basis, we develop a "Supervised Machine Learning Reportcard (SMLR)" to provide guidelines for comprehensively and rigorously conducting and documenting SML research. We review the literature concerning extant steps and SML process frameworks and integrate them into a comprehensive reportcard. Finally, we review 121 relevant articles, which were published from 2010 to 2018 in renowned IS outlets, such as *Management Information Systems Quarterly (MISQ)*, *Information Systems Research (ISR)* and *Journal of Management Information Systems (JMIS)* and the proceedings of the *International Conference on Information Systems (ICIS)* and the *European Conference on Information Systems (ECIS)*. We use this broad sample to analyze how and where the SML documentation of current articles could be improved. This article therefore contributes to a complete and rigorous application and documentation of SML research, which promotes meaningful and reproducible results.

The remainder of this article is structured as follows: We introduce the fundamentals and positioning in the upcoming Section 3.2. Then, we derive and describe the problem characteristics and key choices of each SML endeavor in Section 3.3, followed by the introduction of the Supervised Machine Learning Reportcard (SMLR) addressing them. In Section 3.4, we apply this reportcard in an empirical study to relevant IS articles and analyze their precision when it comes to SML application and documentation. In Section 3.5, we conclude with recommendations, a summary and limitations of the study.

3.2 Fundamentals and Positioning

When it comes to their type of learning, machine learning techniques can be classified as either supervised or unsupervised ones[2] (Mohri et al., 2013). In fact, most real-

[2]Other sources, for example, Fu (2003), also consider reinforcement learning as a third type. However, there is no academic consensus on this definitory classification.

word applications of machine learning are of supervised nature (Jordan & Mitchell, 2015), whereby SML aims to predict the (discrete or continuous) value of an element by using a data set of observations in which this element is already known and labelled with the correct value (Rätsch, 2004). Precisely, we define supervised machine learning as follows—based on Mohri et al., 2013, p. 5: Supervised machine learning is the concept of learning a function mapping an input to an output based on labelled training data, i.e. a sample of input-output pairs. For discrete target values, the problem is called a classification problem, for example, when determining product returns in e-commerce (Heilig et al., 2016). In contrast, predictions of continuous variables, such as forecasts of electricity prices (Feuerriegel & Fehrer, 2016), are subsumed as regression problems. Here, the output of the SML algorithm is not a class, but a numerical value that specifies the predicted attribute.

An SML endeavor, i.e. the application of SML methods to a problem, may serve different purposes and its specific design heavily depends on the particular problem. Shmueli and Koppius (2011) differentiate these purposes in either *explaining* or *predicting* a phenomenon. Regarding the first, statistical models can support explanatory-oriented research for testing causal hypotheses. For instance, if a researcher aims at explaining patterns in the data with a linear regression, individual model results (like the loading of the regression coefficients, the coefficient of determination R^2 , or p-values) might already fully warrant applying the model; there is no further need to evaluate its predictive power on an unseen test or validation set for possible deployment within information systems artifacts (Gong et al., 2018; Li et al., 2016; Martens & Provost, 2014).

On the other hand, *predictive* models can be used to anticipate unseen or future observations. In order to do so, researchers need to analyze SML's potential to solve an empirical prediction problem. Thus, they need to show its effectiveness in their field studies by reporting on the predictive qualities of a trained model. Researchers might compare an SML endeavor to different benchmarks and, consequently, not only show its basic functionality, but also the efficiency of leveraging SML for a certain, possibly productive task (Pant & Srinivasan, 2010). For instance, they may analyze whether a machine can perform a task better than a human (Han et al., 2015). Depending on the scope, this step may even require to implement a predictive model and embed it into a software tool, for example, to continuously make predictions (Oroszi & Ruhland, 2010). The focus of our work is on SML applications for *predictive* purposes.

When discussing replicability or reproducibility of SML studies for predictive purposes, we need to distinguish different possible levels of documentation. The

spectrum of reproducibility originally developed by Peng (2011) for the field of computer science, is well applicable to our IS SML endeavors. On that basis, Figure 3.1 denotes the range of options that increasingly allow reproduction of results: While mere results in a publication do not support any reproducibility, the exposure of method details, code and/or data will help to do so. He argues for the publication of "linked and executable code and data" along with the core article as a gold standard to assure reproducibility.

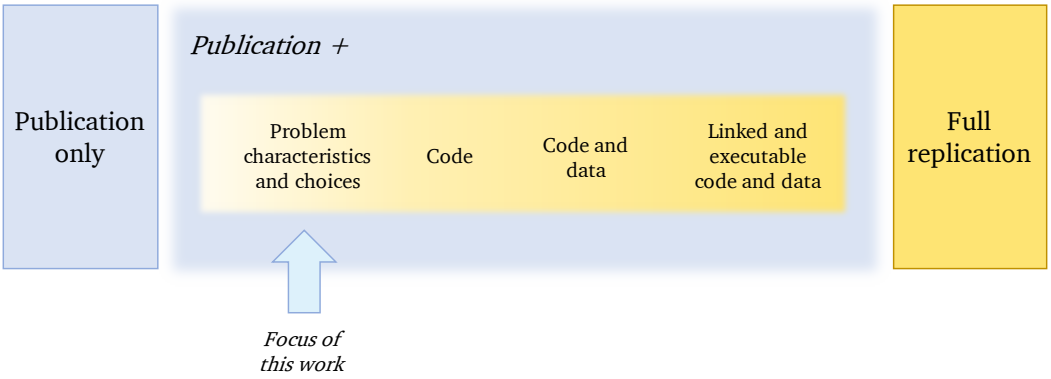


Fig. 3.1.: The spectrum of reproducibility; extended figure based on Peng (2011).

However, typical IS studies cannot comply with a publication of code and/or data due to confidentiality issues (Gimpel et al., 2018; Sharp & Babb, 2018; Timmerman & Bronselaer, 2019), at least if not publicly available data sources are used. For the work at hand, we will, therefore, primarily focus on the documentation of the problem characteristics and choices of applying SML—but still stress the importance of providing code and data whenever possible.

When it comes to process models that support SML for predictive tasks, a variety of different possibilities exist—the most common being Knowledge Discovery in Databases (KDD) (Fayyad et al., 1996), Cross-Industry Standard Process for Data Mining (CRISP-DM) (Wirth & Hipp, 2000) and Microsoft Team Data Science Process (Microsoft, 2020). Although these process models are extremely popular, they are very broad and do not go deep enough to derive measurable criteria for SML endeavors. As they are designed for more general data mining and machine learning purposes, they are (by design) not detailed and lack helpfulness and transparency for our purpose. The same shortcoming of high-level abstraction applies to other, less popular process models (Anand & Büchner, 1998; Brodley & Smyth, 1995; Cabena et al., 1998; Cios et al., 2000; Witten et al., 2011). Since these process models are highly generic and can be applied to any kind of data analysis projects—and not SML exclusively—they only focus on a limited part of the overall choices and problem

characteristics (Kurgan & Musilek, 2006). Furthermore, they do not include precise guidelines for the performance estimation and deployment of an SML endeavor, which are especially important in IS (Shmueli & Koppius, 2011). A process model is also not suitable for communicating results in a scientific publication.

In this article, we therefore derive problem characteristics and key choices as part of the Supervised Machine Learning Reportcard (SMLR); every SML endeavor needs to consider and document them to enable readers and reviewers to fully grasp and judge the individual project—also for replication studies of machine learning in IS research (Hutson, 2018; Olorisade et al., 2017; Voets et al., 2018). Similarly to the proposed reportcard for IS research, related "checklists" were proposed in other disciplines—with the idea to append them when submitting a manuscript to a conference or journal. A number of articles originate from the field of medicine and aim to educate physicians the application of machine learning (Mongan et al., 2020; Pineau, 2020; Qiao, 2019; Winkler-Schwartz et al., 2019). While these articles share some problem characteristics and choices with IS research, their main goal is to map them to the specific needs of a clinical audience.

In the field of CS, three main articles are important: Pineau (2020) proposes a short checklist to foster reproducibility in general machine learning endeavors. He emphasizes precise descriptions in the areas of models, theory, data, code and results, e.g., to include clear README files. In the area of Natural Language Processing (NLP), Dodge et al. (2019) stress aspects of result reporting and especially hyperparameter tuning. To allow for more realistic results, they propose that researchers utilize their novel technique of *expected validation performance*. Furthermore, they elaborate on the documentation of the used hardware. While hardware is an important metric in CS to estimate runtimes and complexities of machine learning models (Dodge et al., 2019; Pineau, 2020), these aspects play a minor role in the reproducibility of the more application-oriented IS—and will be neglected in the remainder of this work. Mitchell et al. (2019) present a "model card" with a focus on fairness and ethics of machine learning models, as they conclude fairness and bias topics are not (yet) integrated into the minds of data scientists.

Apart from CS and with a strong focus on the industrial sector, Studer et al. (2020) propose an adapted version of CRISP-DM for the application of machine learning in the automotive sector with a checklist on specific quality assessment measures. In contrast to these related checklists, our proposed SMLR a) focusses on the holistic SML process from problem statement to productive deployment, b) details the necessary problem characteristics of specifically SML (and not Machine Learning (ML) in general) and c) presents the findings with an IS audience in mind. Where

appropriate, we will highlight where insights from other articles influenced the design of our presented SMLR.

3.3 Towards Rigorous Supervised Machine Learning Documentation

The results of the literature review confirm that so far no process model systematically captures all the problem characteristics to be reported and choices to be made in SML projects in the field of IS. Thus, we set out to collect and merge the necessary problem characteristics and key choices from various sources: We gather individual parts of the entire process from relevant literature and augment other parts based on logical reasoning and best practices gained from the execution of typical SML projects.

3.3.1 Problem Characteristics and Key Choices of Supervised Machine Learning

For the subsequent analysis, we further divide an SML endeavor into the following three main steps: model initiation, model performance estimation, and, if applicable, model deployment (Hirt et al., 2017)—as illustrated in Figure 3.2. In the model initiation step, the objectives for the endeavor are formulated and the matching data set is gathered, prepared, and characterized. Having initiated a model, its performance will be estimated by training and testing models on a data set D in which the target to be predicted is known. First, models learn patterns in the data from a training subset $D_{tr} \subseteq D$ and then apply it towards a test set $D_{Te} = D \setminus D_{Tr}$ of the data, which was not used for training. Cross-validation approaches are applied to perform this with various alternative D_{Tr}/D_{Te} splits.

When conducting SML endeavors, it is important to specify problem characteristics (e.g., class distribution) and elaborate on the choices made (e.g., performance measure). Additionally, it is necessary to state these key insights when publishing the results, because only with this context information can the reader judge the endeavor's rigor and meaningfulness. For instance, if the author does not specify if hyperparameter optimization was used in the SML process, it is difficult to verify whether the models' performance could be further improved or if the author has

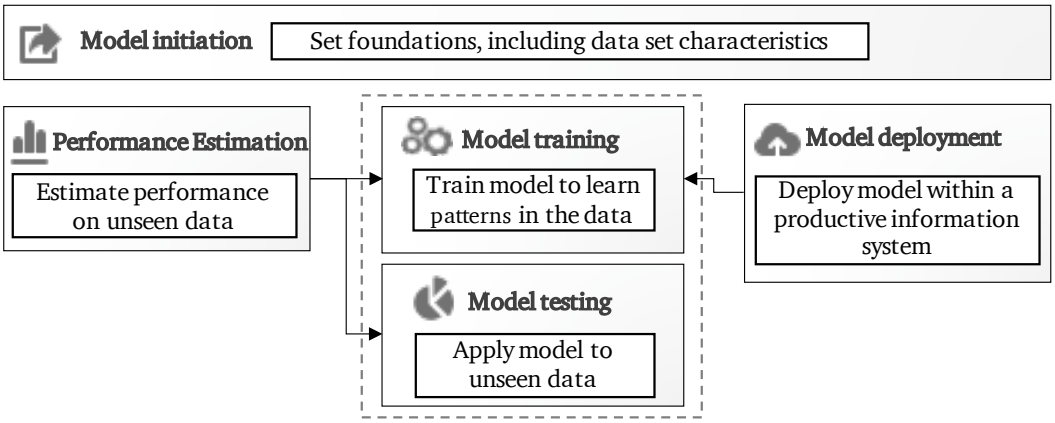


Fig. 3.2.: Overview of supervised machine learning steps.

simply accepted the performance of the first best tuple of hyperparameters (Dodge et al., 2019).

As previously explained, the goal of the endeavor needs to be precisely defined: It should show the purpose and the targeted application (Mongan et al., 2020). The necessary activities of initiation and performance estimation are linked to the first two, while model deployment is also important when implementation is the goal. Model performance estimation aims to estimate a model's performance on unseen data based on a set D_{Te} of data for which the feature to be predicted is known. This is a typical SML step across all disciplines which leverage it, for example, medicine (Shipp et al., 2002) or physics (Rupp et al., 2013). However, when conducting an SML endeavor in IS, not only performance estimation is an inherent step, but also model deployment. This implementation within a productive software tool continuously exposes the model to new, incoming data (Shmueli & Koppius, 2011). While model performance estimation builds on both training and testing activities, model deployment only leverages the training to create a deployable model. For instance, within a model performance estimation not all data can be used for model training, as a certain share needs to be saved for validation and/or testing purposes. For model deployment, however, it is important to use as much data as is available—because more data enables the model to achieve better performances (Banko & Brill, 2001). Therefore, after estimating the model performance, the final model is built by using all available data D in the model deployment phase.

Model Initiation

When conducting SML, a model needs to be defined. A model can be considered as a tuple of parameters that describe which algorithm is used, how its parameters are initiated, and what the general process is like. These basic assumptions and surrounding conditions are defined in the model initiation. They serve as the basis for the subsequent model building, for model evaluation (as part of performance estimation), as well as for model deployment.

First, it is important to state the problem which the SML endeavor aims to address (Qiao, 2019). This requires specifying a target value and the SML problem type—for instance, binary / multi-class classification or regression problems. It should be clear from the start what the problem type is ("What should be solved") (Wirth & Hipp, 2000). Next, the different aspects of the data used and its characteristics are important to estimate the complexity of the task and also to enable meaningful judgement of the final results at a later point. This starts with the data gathering and precise definitions on how it is performed (Oquendo et al., 2012; Winkler-Schwartz et al., 2019). SML requires a target value, which can either be collected together with the data or it can be separately labelled (automatically or manually) after the collection. In any event, it needs to be explained if and how the labelling takes place.

If the volume of the data is too large to be analyzed, it is possible to conduct a sampling[3], which pulls a representative subset of the larger data set (Dhar et al., 2014). Especially in recent years, the process of sampling has not only been relevant to retrieve a representative data set, but also a fair one without any biases (Barocas et al., 2017). With a data set to analyze, additional problem characteristics and key choices need to be specified. The data distribution is of major importance, since it ultimately determines the interpretation of the results (He & Ma, 2013). For instance, in a binary classification on a data set with a minority class distribution of 10%, an accuracy of 90% is easily achievable by simply predicting all observations as belonging to the majority class. This is, furthermore, also a question of the performance metric, which we address at a later point.

Irrespective of the performance metric, however, the number of classes and their shares need to be specifically mentioned for every classification problem (e.g., as

[3]It should be noted that when it comes to machine learning, the term sampling can be used in three different scenarios with different objectives: It can be used to pull representative data as part of data gathering (as described above), it can be used in the distribution of data for a fold as part of the cross-validation (stratified sampling), or it can be used to counterbalance a minority class as part of the model training set (e.g., oversampling).

a table). The same applies to regression problems (e.g., a representation as a boxplot) to enable the reader to understand the basic problem. Furthermore, it is important if and which data preprocessing methods are applied—for any type of data. For instance, in the specific case of natural language processing (NLP), the possibilities of transforming unstructured text data into structured, machine-digestible formats are manifold (Manning & Schütze, 2000). We, therefore, need to specify which transformation techniques are applied and why they are applied for a specific problem. Apart from the preprocessing, statements about the data quality are of interest. Data quality covers many aspects, including correctness ("is it true?"), accuracy ("how precise?"), completeness ("is it complete?") and relevance ("is it related to the initial problem?") (Wang et al., 1993). Sparsity and noise are two examples of data quality characteristics—and there are a number of different complexity measures available to assess them (Ho & Basu, 2002).

Model Training and Testing

Training and testing are essential parts of each machine learning endeavor. However, the purpose of these activities needs to be clearly defined: We particularly distinguish between estimating the model's performance on unseen data ("Performance Estimation") and deploying a model within a software tool ("Model Deployment").

In the model training phase, the sampling of data, which occurs prior to training a model, can have a significant impact on the performance (Chawla, 2010). Popular sampling techniques for dealing with uneven class sizes are undersampling, oversampling or Synthetic Minority Over-sampling Technique (SMOTE). *Undersampling* is applied when the number of random sample instances taken from the majority of observations is limited to match the size of the minority data set used for training purposes (Rahman & Davis, 2013). In contrast, *oversampling* randomly duplicates instances from the minority class so that researchers can work with more instances than originally available (Rahman & Davis, 2013). *SMOTE* creates new additional synthetic instances to match the number of training set elements in the majority class (Chawla et al., 2002).

The core of the model training phase consists of selecting an algorithm, as well as its parameters, which creates another set of choices. For instance, popular machine learning frameworks like the python-based "scikit-learn" (Pedregosa et al., 2011) and the Java-based "WEKA" (Hall et al., 2009) feature more than sixty, respectively, thirty supervised learning algorithm implementations. SML algorithms can be classified in different ways (Caruana & Niculescu-Mizil, 2006; Hastie et al., 2009; Kotsiantis,

2007). Aggarwal and Zhai (2012) divide supervised algorithms into the major classes of linear algorithms (e.g., Support Vector Machines or regressions), decision trees, pattern (rule-)based algorithms, probabilistic and Naive Bayes algorithms, and meta-algorithms. Each of these classes has its advantages and disadvantages—in general, as well as in relation to the specific data and problem they are applied to. While we cannot go into the details of each class, Kotsiantis (2007) provides more details on the particular selection criteria.

When it comes to model testing, it is important to early define one or multiple performance metrics, which serve as the central criteria to estimate alternative models' performance and to finally evaluate the success of the SML endeavor. Common metrics used for classification tasks are, for instance, accuracy, precision, sensitivity, specificity, recall, F-measure or AUC (Powers, 2011). Metrics for regression tasks, on the other hand, include Mean Squared Error (MSE), R^2, Correlation Coefficient (CC), Normalized Root Mean Squared Error (NRMSE), Signal-Noise Ratio (SNR), Coefficient Of Determination (COD), as well as Global Deviation (GD) (Spuler et al., 2015). When it comes to choosing one or multiple metrics, it is again important to consider the nature of the problem, as well as the data set. For instance, although recall is a valuable metric to present the fraction of relevant observations among the retrieved observations, it is not meaningful on its own, since it can easily be brought to 100% by simply predicting all observations as belonging to the positive class. The inherent tradeoff between precision and recall is designed into the set of F-metrics (Goutte & Gaussier, 2005). In the case of regression, R^2 and explained variance are popular choices. Additionally, for both regression and classification, the plotting of a learning curve can be meaningful, because it can show the training and test set errors for each fold of the cross-validation and the respective amounts of data, which helps estimate the bias-variance tradeoff (Blanc, 2016).

Performance Estimation

Based on the performance estimation it is possible to draw conclusions on how the trained model performs on unseen data. In order to do so, it leverages the previously described steps of training and testing. The important step to conduct is splitting the data set to allow for these two activities. There are two different options when it comes to data splitting, namely percentage split and cross-validation (Abdullah et al., 2011). A simple split into a (larger) training set and a (smaller) test set is called a percentage split. The machine learning model is trained on the training set and then applied to the test set for evaluation. In IS research, data is often precious with a limited amount of available observations. Therefore, the prediction

performance on the test set may vary significantly in the case of a percentage split, because, depending on which instances are present in the training set, it may or may not be trained as "well" (James et al., 2013). Generally, the error resulting from this prediction can be divided into bias, variance, and irreducible error (Friedman, 1997). In order to counteract the random effect of choosing data for the sets, a k-fold cross-validation can be implemented. Here, the original data is divided into k folds of equal size. The model is trained with (k-1) folds (training set) and applied on the remaining fold, called validation set or local test set. This process is repeated k times with each of the k folds. The aggregated performances from the individual iterations are averaged and represent a more meaningful performance assessment than a single percentage split (Golub et al., 1979). For both cases, percentage split and cross-validation, stratified sampling allows for maintaining the original data set's distribution within the training and test set (Neyman, 1934), which reduces the randomness associated with allocating the two subsets.

If the goal is to simply demonstrate the capabilities of one machine learning model, one-time splits, such as percentage or k-fold, can be sufficient. If, however, the plan is to try out different models, optimize parameters, and estimate the error of a model on unseen data, additional steps should be undertaken. If any optimization takes place, it is important to test the model on completely unseen data—that is, data, which has never been used in any training or optimization iteration (Cawley & Talbot, 2010). A so-called hold-out set or global test set should never be used to change models or the choice of them, but preferably only to evaluate them once (Tušar et al., 2017). In order to address this, the nested cross-validation first splits the data into training/validation set and a hold-out set. Then, cross-validation with parameter optimization can be applied within an inner cross-validation, thereby making it possible to select and evaluate—but not again optimize—the best performing models within the outer cross-validation. To summarize, when it comes to model performance estimation, separating the data into multiple sets is of importance and depends on the use case:

- Training set refers to the data set on which the model is trained.
- Validation set or local test set refers to the data set on which the model is optimized. It must, however, not be used to evaluate the model's performance, otherwise the model tends to overfit. A validation set is crucially important if parameter optimization is performed.
- Hold-out set or global test refers to the data set according to which the model is evaluated, but according to which it is never optimized.

Model Deployment

The final model deployment phase aims at generating, implementing, and distributing a previously built supervised machine learning model within a software tool. Data contains information and is valuable—therefore, using the complete data set is meaningful for the final machine learning as depicted in Figure 3.3 (Gama et al., 2004). It would incorporate parameters, which were typically previously selected from the performance estimation. These parameters also help in understanding the robustness of the model (i.e., its tendency for overfitting). For instance, analyzing the optimal parameters of the cross-validation's inner folds might reveal that a specific parameter combination occurs multiple times, or, if the model is very stable, all the time. This combination of parameters might then be directly used for the final training. Alternatively, an additional cross-validation with the complete data set can be utilized to choose the parameters for final training.

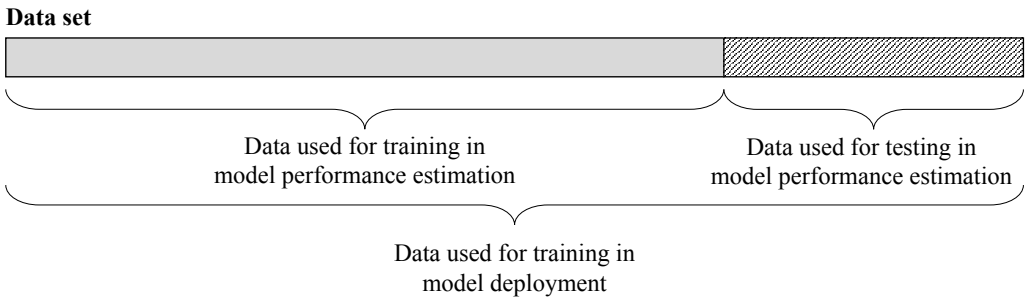


Fig. 3.3.: Data sets for training, testing, and final deployment.

Then, an export of the final model, also called serialization (Zaharia et al., 2018), is needed to save the state of the model and the used preprocessing pipeline for further usage. Having concluded the serialization phase, the serialized object can be built into a workflow, such as a connected web service, to predict the target value of new, incoming data. Hereby, data is sent to the serialized object to be preprocessed and classified by the model. It is important to consider the validity of this final model, for example, how robust is it to changes in the data (Gama et al., 2004) and/or whether its performance is continuously maintained (Feurer et al., 2015). Since the model building data might be topical at that point in time, the data might change in the future. It is important to address this, preferably directly by continuously updating the model automatically, or, at least, by (qualitatively) estimating the performance for future changes (Baier, Kühl, et al., 2019). For instance, in the case of sensor data in a production line, the predictive model might still be valid for a long time—as long as the produced goods remain the same. However, if elements

of the production line change or new goods are produced, the model needs to be updated. In sum: It is important to address how the model copes with new, incoming data and, consequently, whether or not the model is continuously improved—and if not, why it is not necessary.

3.3.2 The Supervised Machine Learning Reportcard (SMLR)

For each step of an SML endeavor that were laid out in Section 3.3.1, we aim to identify key choices and problem characteristics to systematically capture and document them. In Figure 3.4, we present the Supervised Machine Learning Reportcard (SMLR), which allocates the identified problem characteristics and key choices alongside these steps. When conducting and describing a supervised machine learning endeavor, they should be addressed and defined.

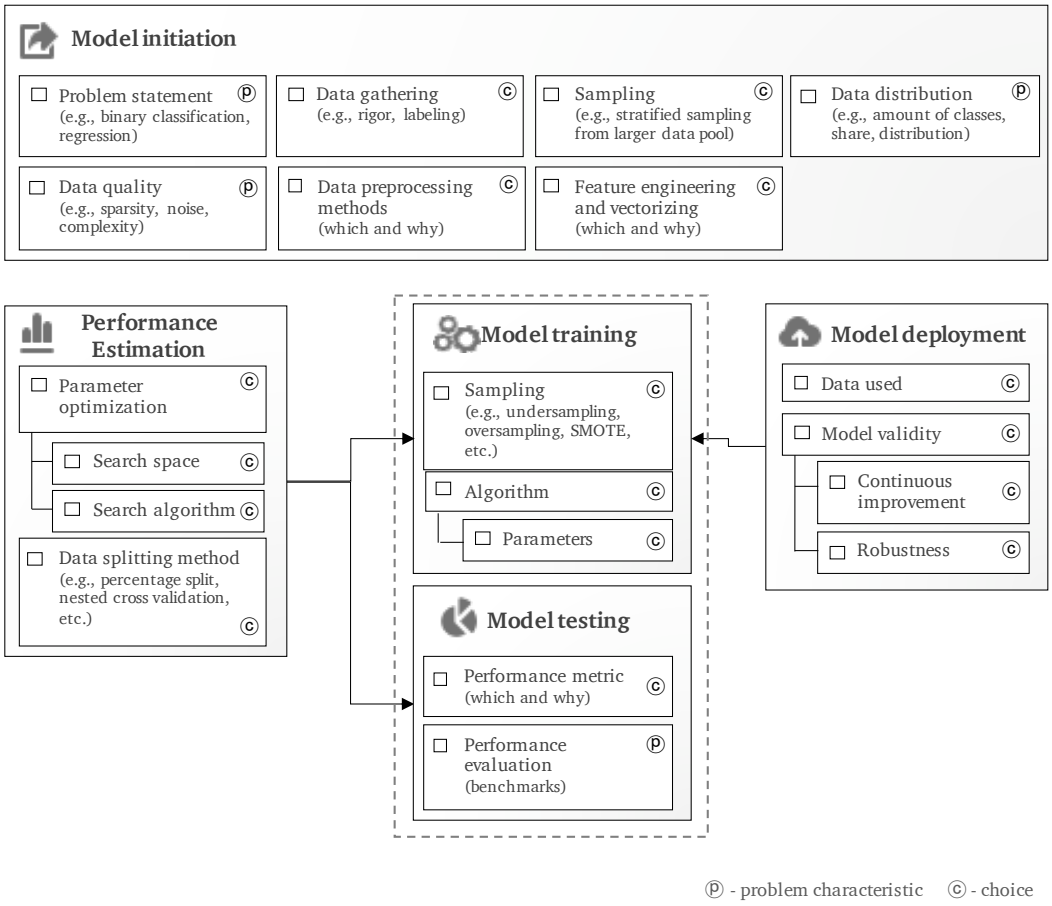


Fig. 3.4.: Overview of supervised machine learning steps and corresponding problem characteristics and key choices.

During the model initiation phase, the problem statement itself is a key characteristic, which classifies the supervised machine learning problem as being either a binary, a multiclass or a regression problem. Since every supervised approach requires data, a detailed description of the data gathering process, as well as the construction of a ground truth data set, should be provided. In order to better understand the data itself, data distribution should be described, as well as the overall data quality, that is, for example, the sparsity and noise of the data. Depending on the distribution of classes, sampling of data points might be necessary and needs to be described by the authors (e.g., type of sampling). Lastly, data preprocessing (Kotsiantis, 2007), as well as feature engineering and vectorizing (Domingos, 2012), not only have a major influence on the overall performance of the trained model, but also bear the risk for major methodological mistakes, such as data leakage. It is important to consider different methods, as well as reasons for their usage.

In the error estimation phase, the model's performance on unseen data should be estimated. Thus, information about the algorithm, the parameter search space, and the search algorithm (e.g., grid search, random search), as well as the data splitting method (e.g., percentage split, cross validation) needs to be specified. In the proposed reportcard, we list model training and testing as two separate units, which require thorough description. During the model training phase, data can be sampled to train a better prediction model. Furthermore, researchers should describe the algorithm that was used, as well as its implementation. This requirement goes beyond simply reporting the name of the approach. Especially for neural networks, researchers need to rigorously document the architecture of their model which for instance includes the type of network layers (e.g., convolutional, recurrent, or fully connected layers) applied and the number of neurons per layer.

The choice of a suitable performance metric for a given problem is essential for the success of a supervised machine learning endeavor. Whereas accuracy might represent a model's performance well in a class-balanced scenario, its descriptive capability typically decreases when it comes to highly imbalanced data. Each performance metric has its advantages and disadvantages. It is advisable to either use multiple (e.g., Accuracy + Precision + Recall + AUC) or composed (e.g., F-score) metrics, as single metrics can be easily tuned and do not represent a holistic overview of the qualities of the predictive model. Furthermore, the results need to be contextualized according to a performance evaluation/benchmark. For instance, if the utilized data set has been used in other articles or even data science challenges like Kaggle, the performance results obtained from these works should serve as a benchmark for direct comparison. If such results are not available, obvious benchmarks should be referred to. These could be either naïve models (e.g., a

random guess or the prediction of the majority class/mean from the training set) or simpler models (e.g., a basic linear or logistic regression). By providing this context, the reader can better understand the quality of the obtained performance.

The performance of an estimated model can be used to show the effectiveness of a model. If it needs to be implemented for predictive modeling as part of the model deployment phase, the model is put into practice to solve the initial problem. In this scenario, the algorithm, as well as the previously identified parameters and sampling method should be used for model training. Furthermore, the data, which was used for training the final model, should be described. Since models can only represent a hypothesis based on training data, its validity decreases as the corresponding real-world situation changes. In order to address these changes, researchers should address the model validity and possible continuous improvement techniques, as well as the model's application to unseen data (robustness).

To ensure the completeness of our approach, we compare the characteristics and choices included in the reportcard with two widely used process models for data science projects, namely Microsoft Team Data Science Process (Microsoft, 2020) and CRISP-DM (Wirth & Hipp, 2000). This analysis reveals that the reportcard in fact covers all important aspects of a machine learning endeavor. We can only determine a gap between the reportcard and the two process models regarding the documentation of requirements from the field as well as details on the business assessment. However, those two aspects usually do not apply to the academic context. A detailed comparison with Microsoft Team Data Science Process and with CRISP-DM can be found in the appendix in Table A.1 and Table A.2.

The first and foremost aim of this work in general and the SMLR in specific is to generate awareness for the identified problem characteristics and key choices when conducting SML. However, if applicable, it can be also utilized as a framework to document these precise choices. To demonstrate a possible application, we depict a typical machine learning challenge—using the Iris data set (Fisher, 1936)—and report on the results in Table 3.1.

Tab. 3.1.: Exemplary reportcard based on the Iris data set. Bold writing indicates a problem characteristic or choice from the Reportcard.

Problem statement	Predict iris flower class based on the four attributes Petal Length, Petal Width, Sepal Length, Sepal Width		
Data gathering	Pre-defined data set by scikit-learn package for Python (Pedregosa et al., 2011), originating from Fisher (1936)		
Data distribution	Three flower classes setosa, versicolor, virginica with 50 instances each; 150 instances in total		
Sampling	No sampling		
Data quality	No missing values		
Data preprocessing methods	No preprocessing		
Feature engineering and vectorizing	No additional features apart from Petal Length, Petal Width, Sepal Length, Sepal Width, no		
Performance estimation			
Parameter optimization	Yes		
	Search Space	RBF kernel	$\gamma \in \{0.001; 0.0001\}$ $C \in \{1; 10; 100; 1000\}$
		linear kernel	$C \in \{1; 10; 100; 1000\}$
	Search algorithm	Grid Search	
Data split	Nested cross-validation, 3 outer folds, 5 inner folds		
Algorithm	Support Vector Classifier		
Sampling	No sampling		
Performance metric	F1-score as a compromise between precision and recall		
Performance evaluation	Average F1-score performance on outer folds: 0.9778, which is a nearly perfect score		
Model deployment			
Data used	Full data set (150 instances)		
Model validity	**Continuous improvement**	No continuous improvement	
	Robustness	No statement about the suitability possible	
Sampling	No sampling		
Algorithm	Support Vector Classifier		
	Parameters	RBF kernel	$\gamma = 0.001$ $C = 1000$

3.4 Empirical Study

With the SMLR at hand, we review renowned articles from IS literature to identify the strengths and possible improvements on the basis of the presented key choices and problem characteristics.

3.4.1 Methodology and Data Set

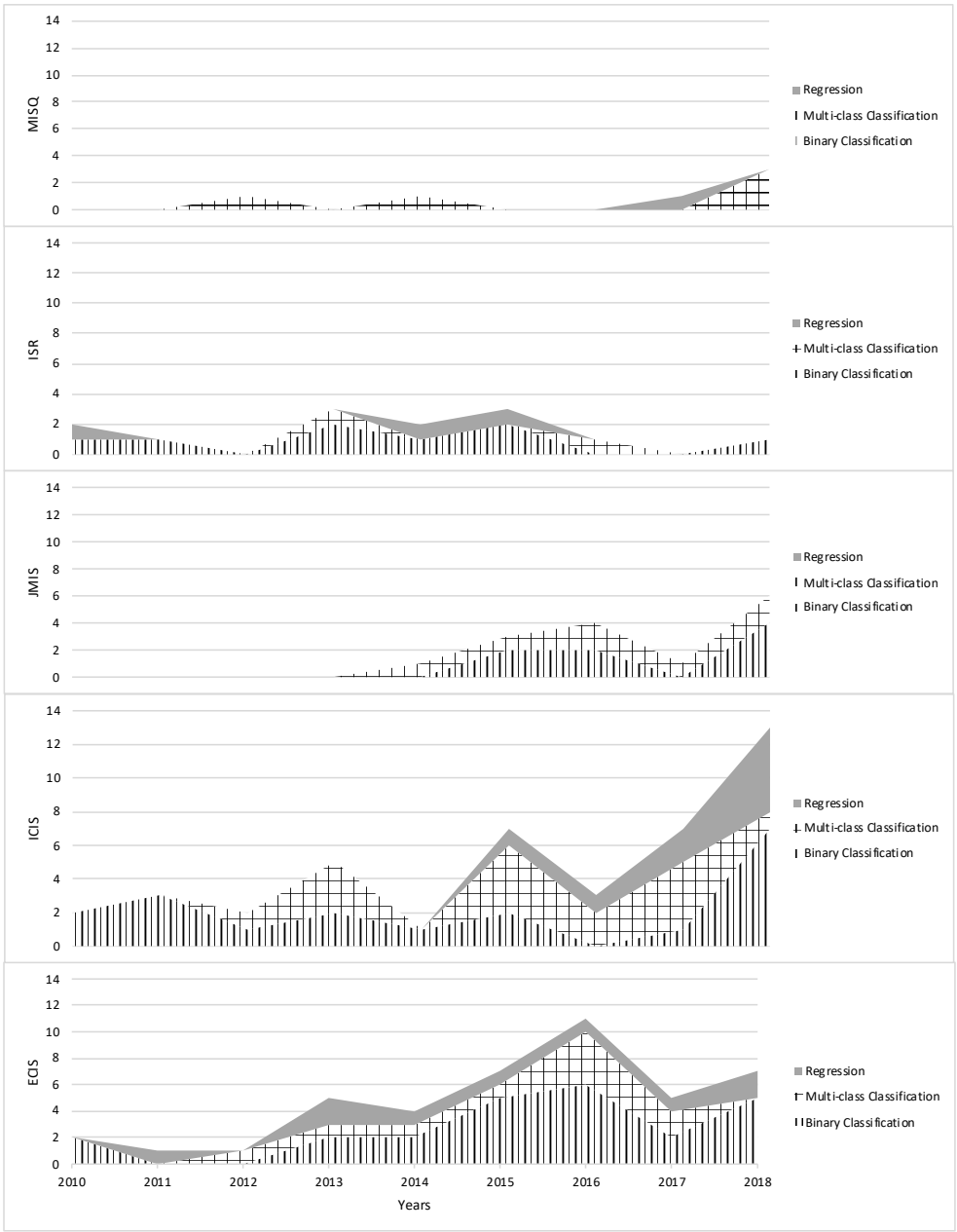
For our study, we aim at covering a broad range of high standard, high quality publications in IS. The JOURQUAL3 rating, which conducted a total of 64,113 journal and conference evaluations from 1,100 professors (VHB, 2012) serves as our basis. We focus on the top three journals and top two conference proceedings in the IS community (Hennig-Thurau et al., 2004), namely *Management Information Systems Quarterly (MISQ)*, *Information Systems Research (ISR)*, *Journal of Management Information Systems (JMIS)*, of as well as the proceedings of, respectively, the *International Conference on Information Systems (ICIS)* and the *European Conference on Information Systems (ECIS)*.

Tab. 3.2.: Number of screened and relevant articles for each outlet from 2010 to 2018.

	MISQ	**ISR**	**JMIS**	**ICIS**	**ECIS**	$\sum$
Screened articles	288	463	390	3,118	2,257	6,516
Relevant articles	7	13	15	43	43	121
Binary classification	1	8	8	19	24	60
Multi-class classification	5	2	7	15	10	39
Regression	1	3	0	9	9	22

In order to obtain a meaningful number of articles for our study, we cover the time range from 2010 to 2018. In total, we download and screen 6,516 articles (see Table 3.2). Among those papers, we identify those articles where the application of SML plays a major role. Naturally, there are "borderline cases" where SML is only applied on a side note and documented within a few sentences or small paragraph—while the overall goal of the research article is of different nature and SML is not at the core of the project. To name a few examples: Huang et al. (2017) apply SML for an automated sentiment labeling, Walden et al. (2018) utilize SML for an aspect of their experiment analysis and Ivanov and Sharman (2018) merely apply SML in

Tab. 3.3.: Amount of supervised machine learning articles in the outlets of MISQ, ISR, JMIS, ICIS, and ECIS from 2010 to 2018.



the appendix for a robustness check. For our study of rigor SML application, we exclude these cases as SML was not the designated main method for the respective articles. However, we want to stress that our proposed SMLR would be a meaningful addition to the documentation of these small applications, too: While researchers would not need to go into detail in the body of the text, they could just append the filled reportcard at the end of the article for the interested reader and replicant.

In a first step, we identify 121 full-research and research-in-progress articles, which describe an application of SML as detailed in Table 3.2. It is interesting to note how the importance of SML in IS developed over the years. In 2010, only six articles were published which applied SML in their research; in 2018, their number peaked with 30 research articles. More details on the chronological development in the distinct outlets are presented in Table 3.3.

Next, we thoroughly examine all 121 articles across the entire time frame regarding the reportcard steps with their problem characteristics and key choices previously defined in Section 3.3. We distinguish between binary classification, multi-class classification, and regression problems (Chollet, 2018). The majority of SML-based articles (60) solves binary classification problems (e.g., Amrit et al. (2015), Oh and Sheng (2011), and Pant and Srinivasan (2010)), followed by 39 articles with multi-class (e.g., Dorner and Alpers (2017), Geva and Oestreicher-Singer (2013), and Wang et al. (2013)) and 22 articles with regression problems (e.g., Ding et al. (2015), Feuerriegel et al. (2014), and Riekert et al. (2017)).

Next, we describe our findings with regard to the different steps of model initiation, performance estimation, and model deployment, which we have defined in Section 3.3. These findings are summarized in Table 3.4 and will be discussed in the following. Table A.3 and Table A.4 in the appendix show the individual analyses for journals and conferences. It is important to note that we assess all publications according to the same, objective criteria. We do not consider whether each of the indicators is meaningful for the individual publication; for example, it might not be necessary for a study on the feasibility of SML for a certain business challenge to deal with the necessary steps for deployment.

3.4.2 Model Initiation

Describing the data characteristics is a fundamental part of understanding the model that is built on top of it. At first, it is necessary to name the data source and/or the data collection process. In 12% (15/121) of all the reviewed articles,

Tab. 3.4.: Overview of Supervised Machine Learning Reportcard steps and their documentation.

Step	Indicator		Described in articles			Positive Example
Model initiation	Problem statement		100,00%	(121/121)		Abbasi et al. 2012
	Data gathering		87,60%	(106/121)		Lin et al. 2017
	Data distribution		74,38%	(90/121)		Stange and Funk 2016
	Sampling		14,88%	(18/121)		Dhar et al. 2014
	Data quality		67,77%	(82/121)		Hopf et al. 2017
	Data preprocessing methods		76,03%	(92/121)		Johnson, Safadi, and Faraj 2015
	Feature engineering and vectorizing		74,38%	(90/121)		Martens and Provost 2014
Performance estimation	Parameter Optimization	Search Space	12,40%	(15/121)		Riekert et al. 2017
		Search Algorithm	13,22%	(16/121)		Zhou 2017
	Data split		95,87%	(116/121)		Urbanke, Uhlig, and Kranz 2017
	Algorithm		100,00%	(121/121)		Oh and Sheng 2011
	Sampling		6,61%	(8/121)		T. Wang et al. 2013
	Performance metric (reasoned)		48,76%	(59/121)		Fang et al. 2013
	Performance evaluation		48,76%	(59/121)		X. Han, Wang, and Huang 2017
Model deployment	Data used		1,65%	(2/121)		Abbasi et al. 2018
	Model validity	Continuous Improvement	1,65%	(2/121)		Seebach, Pahlke, and Beck 2011
		Robustness	15,70%	(19/121)		Koroleva and José Bolufé Röhler 2012

neither the data's origin, nor the source from where the authors have gathered it, is clearly stated. The quality of data determines the quality of the model; however, 32% (39/121) of the screened articles do not provide any information on data quality. 26% (31/121) of the articles do not describe the statistical distribution of the applied data set. This relates both to the distribution of the target variable and to the information about the attributes, which is used for prediction. When this information is lacking, it is impossible to judge the final model's performance for a given metric. Furthermore, if the distribution is unknown to the reader, the performance values can be meaningless—for example, a reported accuracy of 99% with a 1% minority class can already be achieved by simply assigning all instances to the majority class.

A good example of a sound data description is provided in Bretschneider and Peters (2016) who refer to the total number of messages and the number of harassment messages (target variable), which is included in their dataset. Data preprocessing and the engineering of features are also essential choices during an SML endeavor. However, 13% (16/121) of the reviewed articles do not include any information about the preprocessing or feature engineering activities that are chosen. Yet, this information is very valuable to any researcher or practitioner who wants to build a predictive model in the same domain. For instance, if we do not know how quality issues, such as incomplete data, have been handled, the results may be flawed.

Furthermore, it is impossible for other researchers to re-create results if the data's preprocessing techniques are omitted, because various different possibilities for preprocessing exist. Stange and Funk (2015) thoroughly explain how they transform real-time advertising data before feeding this data into the model training phase. Thereby, they enable others to benefit from their knowledge.

The performance assessment of a model highly depends on the chosen performance metric. This is therefore, a critically important decision for every SML endeavor. Due to the importance of this step, it is vital to specify the performance metric and the reason why it has been chosen. Nevertheless, only 49% (59/121) of all the reviewed articles actually give the reason for the choice of their evaluation metric. For instance, Riekert et al. (2016) state that they apply accuracy as evaluation metric—however, they do not explain why this is the best suited (and meaningful) metric for their underlying problem.

3.4.3 Performance Estimation

In only 19% (23/121) of the reviewed articles authors mention the parameters, which they use in the model training phase. Model's performance can vary significantly depending on the chosen parameters and therefore the parameter space has to be thoroughly defined and described[4]. In fact, 96% (116/121) of all the reviewed articles include information about how they split the dataset into a training set and a test set (e.g., Chatterjee, Saeedfar, et al. (2018), Lash and Zhao (2016), and Urbanke et al. (2017)). If authors do not disclose this information, the reader cannot judge whether induced results are truly rigorous, because it might even imply that they did not split their data at all. If model training and model testing are performed on the same dataset, the measured performance is misleading and unrealistically high (James et al., 2013).

In order to comprehensively understand a trained model's performance, it is important to compare it to previously built models—or other approaches that strive to solve the same problem. Thus, if any previous research or algorithm deals with the same problem or data set, the performance of the developed model should always be compared to the previous model. If there is no previous research, performance should be compared to other metrics, for example, random guesses (Li et al., 2013) or standard SML algorithms. Kozlovskiy et al. (2016) provide a good example by comparing their model's performance to a random guess. Only 49% (59/121) of the reviewed articles actually introduce a performance comparison (e.g., Cui

[4]However, this does not apply to, for example, linear regression, since no parameter choice is required.

et al. (2012), Geva and Oestreicher-Singer (2013), and Han et al. (2017)). The remaining articles merely introduce the results of the predictive models without any comparison, in which case a reader can hardly judge the actual quality of the presented model.

3.4.4 Model Deployment

The articles in our study show the least reportcard compliance when it comes to the model deployment phase. As we pointed out earlier, the deployment phase is not a mandatory/necessary phase for each SML endeavor in IS research. In certain cases, authors may only want to prove the feasibility of an approach, which includes the application of SML. If a project focuses on this, it is not necessary to build a deployable solution and describe how this is best achieved. Nevertheless, only 26% (31/121) out of all the reviewed articles in our study at least describe the thoughts about a possible model deployment and the corresponding implications. This is only a small share of all screened articles, although IS, as a research discipline, should have a strong focus on producing final, implementable results and implications for practice (Gholami et al., 2016). On the other hand, we also found some evidence for solutions, which were deployed (e.g., Schwaiger et al. (2017)), including explanations on how the authors built their tool and which choices are necessary for deployment in an industry setting. However, even the examples which discuss the model deployment phase do not emphasize which data can be used for the final, to-be-deployed model.

Another consideration is model validity in general and model updates in particular (Baier, Kühl, et al., 2019; Studer et al., 2020). An SML model is built upon data. One assumes that underlying concepts in this data are extracted to fulfill a given task. If an SML model is then deployed, these concepts should not change over time; otherwise the model has to adapt to such "concept drifts" (Gama et al., 2004). If, for example, a model that classifies user-written texts on a social media platform according the age of their authors is not updated from time to time, the prediction quality will decrease—language (i.e., phrases used by certain age groups) will change over time. Thus, we claim that the preservation of model validity needs to be properly addressed.

3.5 Conclusion

Supervised Machine Learning (SML) has become a popular method to solve problems in Information Systems (IS) research and other disciplines. Although SML offers many possibilities for proving effectiveness, efficiency, and application in the problem spaces of predictive modeling, it is important to conduct this research in a rigorous and comprehensive manner. Only by doing so, IS researchers enable their peers to understand and reproduce the conducted research. In this article, we have developed a Supervised Machine Learning Reportcard (SMLR) capturing important key choices and problem characteristics, which need to be considered in every SML endeavor. We elaborate on them and their importance. In an empirical study, we use this reportcard to analyze whether recent articles published in renowned IS outlets already apply the necessary scrutiny in SML descriptions—and we identify several shortcomings in the documentation of SML. For instance, not all the reviewed articles justify the chosen performance metrics and only a minority of them uses benchmarks to help the reader understand the evaluation of the models.

The article at hand has two major limitations. First, we only review articles from five journals/proceedings and only consider instances from 2010 to 2018. While the selection is based on an acknowledged ranking (VHB, 2012), other rankings on important outlets obviously exist. As suggested by this ranking, we treat journal and conference publications alike, although journal publications are typically more mature and show longer histories of revisions. On the other hand, conference publications are timelier and a good indicator for upcoming topics and methods of the community. For the interested reader, however, we append differentiated analyses in the appendix. Regardless of rankings and precise outlets, the general message still remains that we can observe a lack of documentation. This lack can have two reasons: Either the identified key choices and problem characteristics were not considered, or they fell victim to shortening, for example, as part of the review process or the compliance to submission guidelines. Our study can, therefore, only analyze whether important key steps are addressed within the articles; our study does not allow for conclusions to be drawn on the actual research conducted.

The proposed SMLR may prove helpful in future SML endeavors and serve as a guideline to more rigorous, comprehensive research in this area. Once implemented, the reportcard will enable a more transparent view on SML articles—and enable their reproducibility in the future.

4 Challenges in the Deployment of Machine Learning[1]

4.1 Introduction

Due to the large increase of data in recent years, various industries are trying to reap the benefits of this new resource for their service offerings. Machine learning is playing an important role in nearly all fields of business, ranging from marketing over governmental tasks to scientific-, health- and security- related applications (Chen et al., 2012). Furthermore, many companies rely on machine learning models deployed in their information systems for increasing the efficiency of their processes (Schüritz & Satzger, 2016) or for offering new services and products. (Dinges et al., 2015). As Davenport (2006) describes, companies which are able to leverage their data sources through analytical tools achieve a substantial competitive advantage. But also for empirical science, machine learning enables novel ways of analyzing high-dimensional experimental data. This growth in popularity in both science and industry can also be explained by a massive increase in computational power (Jordan & Mitchell, 2015).

However, the wide-spread application of machine learning is rather young and therefore still confronted with many obstacles. Major challenges that have emerged recently in research and practice are datasets with high dimensionality (Cai et al., 2018), model scalability (Hazelwood et al., 2018), distributed computing (Zhou, 2017) and the live application of machine learning on streaming data (Zhou, 2017). In addition, related work has argued that published machine learning research is sometimes not driving sufficient real-world impact (Boutaba et al., 2018). The performance differences in developed algorithms rapidly diminish once applied onto real applications (Rudin & Wagstaff, 2014). Furthermore, it has been criticized that

[1]This chapter comprises an article that was published as: Baier, L., Jöhren, F., & Seebacher, S. (2019). Challenges in the Deployment and Operation of Machine Learning in Practice. *Proceedings of the 27th European Conference on Information Systems (ECIS)*, Stockholm & Uppsala, Sweden. https://aisel.aisnet.org/ecis2019_rp/163. Note: The abstract has been removed. Tables and figures were reformatted, and newly referenced to fit the structure of the thesis. Chapter, section and research question numbering and respective cross-references were modified. Formatting and reference style was adapted and references were integrated into the overall references section of this thesis.

research indeed performs evaluations on real-word datasets but that it does not appropriately communicate the results back to the application domain (Wagstaff, 2012). This is closely related to the criticism on using unrealistic evaluation metrics (Rudin & Wagstaff, 2014). These challenges mainly refer to technical issues. However, the successful implementation of a machine learning project also requires the consideration of organizational aspects. Therefore, in this work, we are interested in the predominant challenges during the practical implementations of machine learning projects. This leads to the following research question:

> **Research Question A**
> Which challenges in the application and deployment of machine learning can we identify in practice?

For answering this question, we perform at first a structured literature review of challenges named in literature which are organized along the categories pre-deployment, deployment and non-technical challenges. Subsequently, we conduct a study with 11 semi-structured interviews with machine learning practitioners working in various industries for identifying relevant challenges in their daily work. Subsequently, we perform a comparative analysis between challenges identified with the interviews in practice and the results from literature. In contrast to previous work focusing on technical challenges, we also identify non-technical ones such as a proper expectation management as well as challenges with creating new digital services based on machine learning. Our overview of challenges can guide the development of more realistic machine learning models in academia and can also be used as a support tool for practitioners in order to more efficiently plan and execute their machine learning projects.

The remainder of the paper is structured as follows: Section 4.2 covers related literature that we base our research on. In Section 4.3, we describe our methodology for the interview study before we present our results in Section 4.4. Section 4.5 discusses our results and compares literature and interview challenges. The final Section 4.6 describes theoretical and managerial implications, acknowledges limitations and outlines future research.

4.2 Related Work

Various challenges regarding the application of machine learning are considered in literature. In order to give a broad overview, we considered various domains

and the whole life cycle of a machine learning project. We perform our literature review with AISel and Scopus as databases according to the methodology described in Webster and Watson (2002). We draw upon CRISP-DM (Wirth & Hipp, 2000) which is a standard process model for data analytics projects for organizing the resulting challenges. However, some of the steps of CRISP-DM overlap and it is difficult to distinguish between them with regard to arising challenges in practice (e.g., data understanding and data preparation). Furthermore, business understanding is usually not covered in scientific literature. Therefore, we merged all challenges related to steps before the deployment of an machine learning model to pre-deployment challenges and the subsequent challenges to deployment challenges. Furthermore, we realize that machine learning projects often are also accompanied by other challenges which cannot be classified into the previous two categories. Therefore, the category non-technical challenges is added. Figure 4.1 introduces the applied categories.

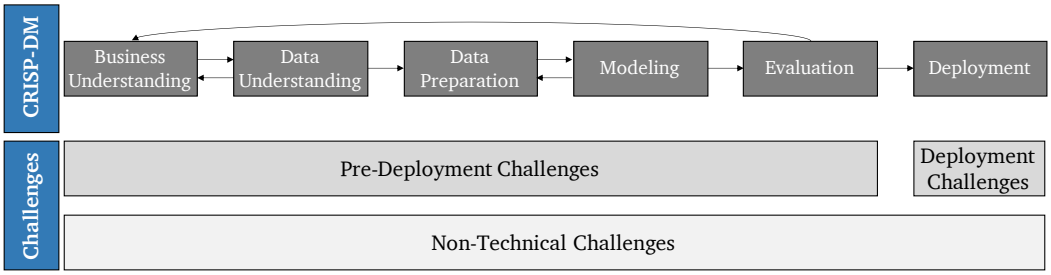


Fig. 4.1.: Conceptual phase model for categories of challenges.

Additionally, after performing the literature search, we merged the identified challenges into 6 distinct clusters consisting of similar challenges: Data structure, Implementation, Infrastructure, Governance, Customer relation and Economic implications. Those are marked by blue boxes in Figure 4.2 which introduces the relevant challenges identified in literature.

Pre-Deployment Challenges:

Data is the fundamental basis of every machine learning project. The proper data structure with the right quality and also a sufficient amount of data samples is a prerequisite for a successful project. Often only a small amount of data being available is a problem (Boutaba et al., 2018; Brodley et al., 2012; García-Laencina et al., 2008; Zhang et al., 2019).

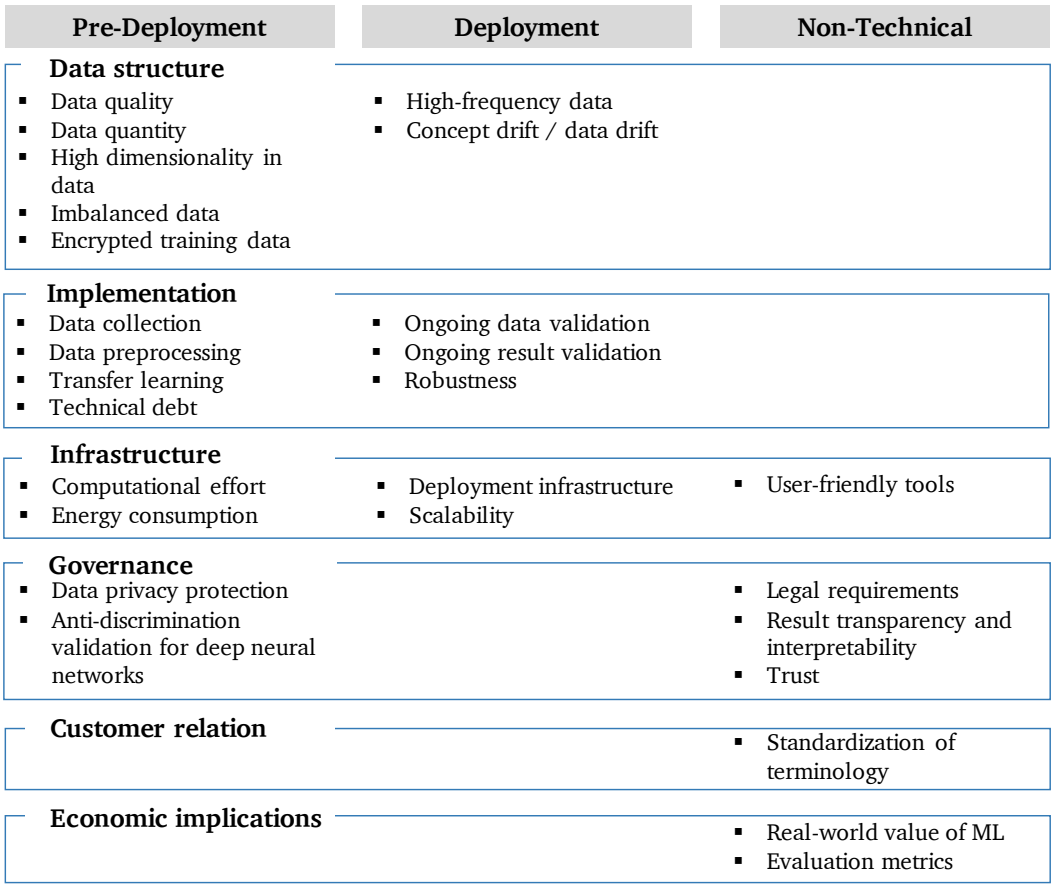


Fig. 4.2.: Challenges identified in literature.

If data is available, incomplete data, incorrect entries, or noisy features often make it difficult to achieve satisfying results (Baesens et al., 2014; Brodley et al., 2012; Kocheturov et al., 2019; Werts & Adya, 2000). One more problem is imbalanced or biased data. Even though a lot of solution approaches have already been presented, there is still room for improvement (Blenk et al., 2017; Kocheturov et al., 2019; Zhou et al., 2016). Applying machine learning on encrypted data sets is also challenging. However, even though some progress has been made for challenges such as training on small encrypted data sets, there is still a lot of work to do for training on large encrypted data sets (Graepel et al., 2012; Sarwate & Chaudhuri, 2013; Xie et al., 2014).

Moreover, big data brings along its own challenges. In many cases, it is still a huge problem to train algorithms on large amounts of data (Saidulu & Sasikala, 2017; Zhang et al., 2019). Especially, handling big data in reasonable time is challenging (Lopes & Ribeiro, 2017). Furthermore, the high dimensionality of datasets in big data applications leads to a more complex feature engineering as well as requires other

preprocessing steps (Cai et al., 2018; Domingos, 2012; Ferguson, 2017; Kocheturov et al., 2019; Sarwate & Chaudhuri, 2013).

Furthermore, transfer learning for applying models on similar tasks across various domains (Suthaharan, 2014) can be challenging due to different data distributions. It is difficult to weight already learned patterns against information from new training data in this context (Jordan & Mitchell, 2015; Silver, 2011; Wang et al., 2016; Ying et al., 2015). Additional problems during the modeling phase are the concept of technical debt, describing the additional time needed in the future to adapt unclean code compared against clean code (Sculley et al., 2015). This is a challenges which arises during pre-deployment but which also might consequences during the deployment phase.

Infrastructure issues such as reducing the computational effort for model training and thereby lowering the memory requirements and energy consumption as well as increasing training and performance speed are often mentioned challenges with regard to the model design (Jordan & Mitchell, 2015; Saidulu & Sasikala, 2017; Shafique et al., 2017; Xie et al., 2014; Zhang et al., 2019). These problems especially come along with big data sets. Solutions need to be found for making the models applicable in practice (Dietterich et al., 2008).

In addition, data privacy protection and data security are governance challenges that need to be considered when applying machine learning models (Lopes & Ribeiro, 2017). Legal frameworks like the European General Data Protection Regulation increase the complexity of the deployment of well-functioning solutions (Malle et al., 2017).

Since various authors use different terminologies when describing similar phenomena, we will use the terms "evaluation" and "validation" synonymously. Especially for ’black box’ models like deep neural networks validation is challenging (Staples et al., 2016; Zhou, 2017). A certain level of transparency is necessary, if automated decisions are supposed to be based on such models in the future. It is crucial to ensure that the machine learning model does not discriminate based on any racial, sexual or other characteristic during its decision process (Anderson & Anderson, 2015; Staples et al., 2016).

Deployment Challenges:

During deployment, incoming data arriving with high frequency and large quantity can be challenging (Polyzotis et al., 2017). Concept drift, which describes a change

in the distribution of input data or the distribution of the target variable, is an additional relevant challenge during deployment (Baier, Kühl, et al., 2019; Gama et al., 2014; Heit et al., 2016; Saidulu & Sasikala, 2017; Tsymbal, 2004; Widmer & Kubat, 1996). Dietterich et al. (2008) name being able to handle changing distributions as one of the requirements for theoretical models to be applicable in practice.

Ongoing validation is an additional challenge for the implementation of machine learning models in practice. Algorithms developed and validated in research environments are not automatically applicable and easily validated for large data sets during deployment (Staples et al., 2016). This applies to validating incoming data with regard to quality and completeness as well as the resulting model predictions (Polyzotis et al., 2017). Furthermore, robustness is named as a major challenge. This refers, among others, to detecting and handling outliers appropriately. Moreover, reasonable results still need to be issued when the quality of the input data decreases (Boutaba et al., 2018; Hazelwood et al., 2018; Zhou, 2017). Ensuring robustness is described as very crucial and difficult, especially in autonomous driving applications (Koopman & Wagner, 2017).

The scaling of small models developed on local hardware to deploying it on a large infrastructure with big amounts of data can cause problems. On the one hand, the infrastructure itself often leads to difficulties. Building up infrastructures with massive amounts of computing power (Hazelwood et al., 2018; Lopes & Ribeiro, 2017; Shea et al., 2018), handling the energy consumption of those architectures (Hazelwood et al., 2018; Shafique et al., 2017) and working with infrastructures like mobile devices or cars (Koopman & Wagner, 2017) is challenging. On the other hand, applying the algorithms on large amounts of data or on various types of infrastructures requires dedicated knowledge (Boutaba et al., 2018; Dyck, 2018; Parker, 2012). Ensuring that the models process incoming data and generate decisions within narrow time windows can be challenging, especially in use cases such as credit fraud detection (Baesens et al., 2014).

Non-Technical Challenges:

The application of machine learning models for people with no background in data science is still quite challenging. Therefore, the introduction of user-friendly tools enabling non-technical employees to build their own models is required, since this would drastically increase the real-world impact of machine learning techniques

(Dyck, 2018; Ferguson, 2017; Zhou, 2017). This is closely related to the concept of self-service analytics (Acito & Khatri, 2014).

Legal requirements often pose a significant challenge for a machine learning project. This relates to data privacy protection as well as decisions on who is going to be accountable for false decisions based on machine learning models (Koopman & Wagner, 2017). In addition, the results of machine learning models need to become more transparent and understandable for domain experts (Leung et al., 2016; Werts & Adya, 2000). Especially deep neural networks appear as a problem when it comes to transparency and interpretation (Nunes & Jannach, 2017). However, in many domains results must be fully understandable to be really valuable (Nunes & Jannach, 2017; Rudin & Wagstaff, 2014). A problem closely related to understanding and transparency is trust. Only if users really trust the results of machine learning models, they will rely on them when facing the challenge of making important decisions. Since the level of transparency for many machine learning model types is still low, also trust remains an open challenge (Baesens et al., 2014; Nunes & Jannach, 2017; Shafique et al., 2017). Furthermore, the highly specialized and little standardized terminology used in machine learning is stated as a problem for novices in the field (Rudin & Wagstaff, 2014; Wagstaff, 2012).

It is often argued that machine learning solutions developed in research have little or no real world value (Boutaba et al., 2018; Domingos, 2012; Sarwate & Chaudhuri, 2013; Werts & Adya, 2000). More realistic evaluations of model results need to be implemented to solve this problem (Heit et al., 2016; Rudin & Wagstaff, 2014). Journals and editors need to support this development by requesting rigorous assessments of developed solutions under real world conditions (Wagstaff, 2012).

More standardization is needed for evaluating machine learning models (Shafique et al., 2017; Spangler et al., 2000) and the corresponding economic implications. On the other hand, evaluation metrics always have to be considered in the industry context where they are applied. Frequently, the same metrics with equal value ranges are compared for various application fields, even though the range implies completely different meanings (Wagstaff, 2012).

As shown, literature deals with a lot of different challenges for applied machine learning and motivation for research papers are often driven by real world problems. However, we want to examine if the relevant challenges in practice match with the ones stated in literature. The interviews intend to identify a potential gap between the challenges stated in literature and the ones named by practitioners.

4.3 Research Methodology

In order to gain a comprehensive overview of the practical challenges of machine learning projects, 11 semi-structured expert interviews are conducted. Following the approach of Helfferich (2011), an interview guideline is used, structuring the interviews with regard to pre-deployment, deployment and non-technical issues.

4.3.1 Sampling

We apply a purposive sampling approach, including interview partners from various industries. Thereby, we aim to comprehensively cover occurring challenges of machine learning projects by including a variety of different perspectives and applications. Moreover, we ensure that different company sizes and maturity levels regarding data science projects are represented within the interview study. The interview partners (IP) are working in the following industries: Automotive (A) and other Manufacturing (M), Process (P), Power Generation (PG), Health Care (HC), Information Technology (IT), and machine learning as a Service (MLaaS). The consultants (CO) from the MLaaS companies cover the additional fields of Retail (R), Finance (F), E-Commerce (EC), Insurance (I), and Media (ME). Table 4.1 shows a complete overview of the different IP as well as the industry of their respective company.

All experts are developing machine learning solutions within specific projects in their daily tasks. Moreover, each of them has at least one year of experience, except two IP who have been working for six months in the field. Five of the eleven IPs work in machine learning consultancies and six of them hold machine learning positions within a specific company. Nine of the eleven experts live and work in Germany, one in the United States and one in Canada.

4.3.2 Data Collection and Analysis

All interviews are either conducted in person at the interviewee's office, via video call, or via phone call. The interviews were recorded after consent was granted and for further analysis transcribed. A qualitative content analysis (Krippendorff, 2004) is conducted to analyze the interviews. In order to remain open for the identification of new aspects and challenges of machine learning projects, the interview material is coded by applying open coding. Two researchers independently conduct the analysis.

Tab. 4.1.: Industry overview of interviewees.

	Role		Industry									
	Consultant	Industry Expert	M/A	P	PG	HC	IT	R	F	EC	ME	I
IP α	X		X				X		X			
IP β	X							X		X		
IP γ		X					X					
IP δ		X	X	X	X							
IP ϵ		X	X									
IP ζ		X				X						
IP η	X								X			
IP θ		X	X	X								
IP ι		X					X					
IP κ	X		X							X		
IP λ	X		X					X		X	X	X

The resulting code system is discussed and merged after each interview. As the involved researchers did not uncover additional insights after the fifth interview, the final coding system was fully developed, along with the corresponding coding rules. The remaining six interviews are used to evaluate the reliability of the coding system by applying the intercoder accordance. Therefore, the same two researchers code the remaining interviews, using the derived coding system. The number of matching codes per interview is computed, resulting in an average value across all interviews of 77,5%, which underscores the reliability of the derived coding system (Krippendorff, 2004). In order to be able to compare the results of the literature review (see Section 4.2) with the findings of the expert interviews, the derived codes were sorted according to the main categories 'non-technical challenges', 'pre-deployment', 'deployment'. A comparative analysis of machine learning challenges from literature with barriers of industry projects is provided in Section 4.5.

4.4 Results

This chapter introduces the challenges resulting from the interviews as well as first insights on best practices to deal with those challenges. Again, we differentiate between pre-deployment, deployment and non-technical challenges. Furthermore, we also apply the six clusters of challenges as defined in Section 4.2. Figure 4.3 gives an aggregated overview of the identified challenges in both, the interviews (marked with an asterisk) as well as the literature (marked with a four corner star).

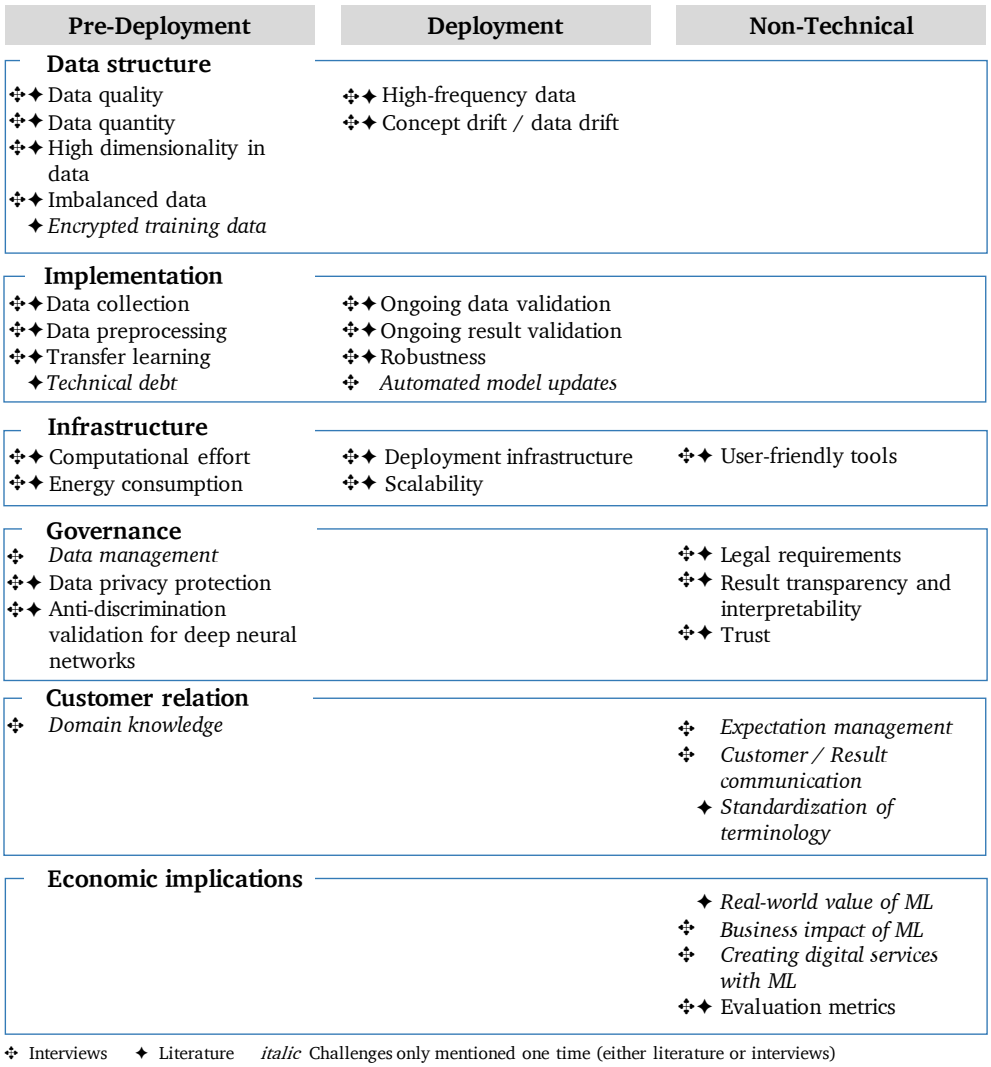


Fig. 4.3.: Challenges identified in interviews as well as in literature.

Usually, the deployment of machine learning models is performed by a specialized team of technical employees in collaboration with the corresponding department which is requesting a solution for its business problems. Therefore, in the following, departments requesting machine learning projects are referred to as 'customers'. We do not differentiate whether the service provider is an external machine learning consultancy or a dedicated machine learning team within the same company.

4.4.1 Pre-Deployment

Challenges referring to the data structure are frequently named by the interview partners, especially problems with data quality and quantity. Usually, data quality is examined before the start of a new project. However, a realistic assessment whether all required data sources are available is often only possible during the project (α, γ, ϵ). Furthermore, recognizing quality problems within the data is often rather difficult without domain expertise. Therefore, data scientists and domain experts need to collaborate closely to identify data quality problems. Imbalanced training data also complicates the application of machine learning models, e.g., in use cases to predict faulty products with a dataset with only very little faulty products at all (ϵ). Limited training data was also frequently mentioned as a challenge. However, some interview partners also stated that this does not affect their daily work. Especially when customers have a lot of experience with machine learning, they usually collect required data before the project start (λ).

Both problems, data with low quality and limited training data, are handled almost similar by all interview partners. First, domain knowledge is taken into account in order to increase data quality. Sometimes, this knowledge is also used as input for the model assumptions. Second, practitioners try to build initial machine learning model as simple as possible, which can provide reasonable results even with a small amount of data. In parallel, more data is collected and the implementation of the model continues as soon as new data is available. However, several interview partners also reported that projects are cancelled if data is too scarce (α,β,γ,ϵ,η). In health care, the problem of data collection is even more complex than in other domains (ζ). This is explained by two reasons: First, the progress of digitalization in general is slower in health care compared to other industries. Some data is not even available in digital form at all. Second, legal regulations for medical data are especially strict.

Data preprocessing is also named a fundamental problem (ϵ) because it is as a very time-consuming task which requires the vast majority of a project's time. Therefore, data preprocessing needs to be automated and accelerated (θ). Increasing the performance of machine learning algorithms and reducing their training time has no practical benefit, if data preprocessing remains as time-consuming as it is today (ι).

The actual modeling work is hardly mentioned as challenging. This refers to activities such as the decision on the type of algorithm or implementing the actual model which has been simplified enormously by the development of open-source frameworks.

Only occasional challenges like transferring a built solution to another domain (θ) or use cases like autonomous driving (ϵ) are indicated. One interview partner (ι) explained that they are working on making deep neural networks more energy-efficient. Instead of using the whole range of available computing power, they want to develop an algorithm which only focuses on the important part of a neural network to improve prediction performance.

Furthermore, governance issues such as legal and access rights often play an important role. Data management in companies across different industries is often organized poorly (γ, η). This is a challenge which is only mentioned during the interviews. Data access guidelines are usually very strict and complex. In general, several approvals are required to access relevant data (γ, ϵ, ζ). Before the actual work on data pipelines and modeling can start, it is often necessary to perform time-consuming tasks on data infrastructures (η). Hence, the wish for more sophisticated data structures in companies in general was stated (ϵ). Additionally, technical transparency is seen as a challenging, e.g., it is difficult to train deep learning algorithms which do not mimic the discriminatory behavior represented in the input data (β).

4.4.2 Deployment

Data with high volume as well as drifts in the input data are frequent challenges. Although there are several technical solutions for automatically recognizing shifts within the input data, like using Kafka input streams, manual checks are done most of the time (β, δ, κ). Changes within the input data are mentioned as a problem, especially for the validation of the model results (ϵ). Manual model adjustments are often performed to match the models to the new data distributions. Only in one case the models are able to adapt themselves automatically to data drifts (η).

In case of machine learning model updates, it is necessary to provide a neat documentation of all models including older versions. It needs to be documented which data has been used for training the model and under which conditions the model was performing well. In addition, an easy rollback to older versions must be available (β, δ). Therefore, a serving infrastructure with a proper model management is required. This allows an easy handling of different model versions as well as the opportunity to frequently update models (δ, ϵ). Furthermore, automated data pipelines pose a problem (κ) since they need to be able to combine database and machine learning model management. Further, templates for machine learning models and an automated, ongoing computation of prediction scores should be included. Cloud solutions offer

standardized solutions for these challenges. Interview partners report fewer issues when using cloud services (λ). However, access to those is often restricted due to data privacy reasons or other restrictions. In general, robustness and stability are seen as major problems in deployment (β, δ, ι). Models must still provide reasonable results when facing minor data changes or a reduction in data quality.

In addition, ongoing validation of deployed machine learning solutions is mentioned as a problem. It is described as a time consuming, unstructured, and unstandardized process (ϵ, η). A key solution in most cases is a dedicated monitoring approach, which is either done automatically, manually, or with a combination of both. Continuous evaluation is the most important principle (β) and a clear definition of the corresponding metrics is required. Three different tests are proposed: consistency checks for the input data, continuous monitoring of the model predictions, and the effect of model predictions on prior defined KPIs. Such a continuous monitoring approach is the basis for a stable system. Manual sanity checks are a widely used mechanism to discover discrepancies in different areas within the pipeline. Further, the results are regularly investigated by domain experts (β, γ, δ, ϵ). In addition, automated consistency checks are used to compare current with previous prediction results. If predefined thresholds are violated, notifications can be triggered. Often, traffic light systems are used as a visualization tool (δ).

Infrastructure is one of the main problems during deployment of machine learning models. Challenges are not only related to deployment infrastructures for running the machine learning models, but also to setting up relevant data infrastructures (ζ, η). Three frequent challenges occur with regard to model deployment architectures according to our interview partners: First, data scientists often need to work with the infrastructure already available on the customer side. Therefore, data scientists have to adapt their solutions to various different infrastructures. This problem arises since approval processes for investments in new infrastructure are very time-intensive and complicated. Furthermore, many customers are very inexperienced with machine learning solutions and do not know if the investment is worth it. Second, standardized architectures for local solutions are scarce. Even if customers are willing to build up new infrastructure, it is difficult to install a consistent local infrastructure. However, cloud solutions already offer this standardization extensively. Third, the actual deployment environment of machine learning models differs significantly. It requires fundamentally different approaches for running a model on either a large cluster in a manufacturing plant, directly in a car or on a mobile device.

Scaling up a model to deployment architectures also brings along additional challenges such as code parallelization. Only few programming languages are easy to parallelize and most of them are limited to computations in the internal memory. Usually, this is solved by building reasonable data partitions. Other approaches rely on using methods that allow out of core calculations (β) or adding more hardware (α, γ). However, the latter can be complicated. Methods for dimensionality reduction such as PCA or auto-encoders are applied for reducing the high dimensionality of datasets (δ). Usually, batch use cases can be handled by frameworks like Apache Spark (α, β). Real-time use cases which require immediate feedback, are more challenging and can require more advanced architectures (β). According to interview κ, only few use cases exist so far where big data infrastructures are really required. Many customers have so little experience that a well equipped, local server combined with a proper feature engineering approach is sufficient and significantly easier to run.

4.4.3 Non-Technical Challenges

During our interview study, we recognize that problems in the daily work with machine learning models are often also related to non-technical topics. Communication with the customer or translating machine learning results into real business impact are just two examples. Additionally, more standardization and user-friendliness for application of machine learning models are mentioned as a challenge. Easily applicable tools need to be developed in order to enable non-technical employees to apply machine learning models (δ).

The effect of machine learning models on business impact refers to two aspects: First of all, there are very different experience levels with regard to machine learning on the customer side. Many customers do not have a clear understanding of machine learning techniques and the corresponding benefits (α). This fact needs to be considered during the development of the customer relation. Second, providing transparency with regard to model results is challenging. It is difficult to convince customers to trust the machine learning results and to apply them for making crucial decisions (γ). Eventually, this challenge might be solved with technical solutions such as advanced frameworks for the visualization of important features. However, transparency itself remains a non-technical challenge. Standard metrics further complicate the problem of transparency. Most metrics are not easily understood by people without a machine learning background. Customers usually have difficulties in translating those metrics (e.g., accuracy) into relevant KPIs such as revenue (β).

Therefore, it is necessary to define individual, customer specific metrics at the beginning of a project to evaluate the results (ϵ) and the economic implications of the model. Furthermore, simpler, more understandable models are applied compared to complex deep neural networks. Customers often behave rather conservatively and select more understandable models over better performing ones (η). Technical solutions for increasing transparency are rare (η). Only few support tools such as Lime for visualizing deep learning models (ϵ) or Starlack for making R algorithms (η) are available.

Still, customer questions often cannot be answered in a satisfying way. Therefore, support from higher management positions is frequently required to communicate results and apply those accordingly (γ).

Although many companies already apply machine learning successfully in support systems, it is still difficult to create valuable digital services based on machine learning solutions (κ). This might also be related to the different accuracy needs for different domains. Changing legal requirements, such as data protection regulations, can further complicate the successful economic application of machine learning projects (β, ϵ).

4.5 Discussion

The interview results confirm the majority of challenges which are mentioned in related literature (c.f. Section 4.2). However, we also identified gaps between the challenges stated in literature and the ones mentioned in our study.

Researchers seem to be aware of many problems that practitioners are confronted with during the development of machine learning models in practice. However, solving these issues appears to be demanding, which might be due to two reasons. First, there might exist dedicated tools but those are not used by the majority of practitioners during their daily work either because those tools are not available (e.g., too costly) or because their usage is difficult. Second, adequate solutions for these problems have not been developed yet. However, we did not specifically ask whether the first or the second reason are the main driver of the respective challenge and therefore cannot make any statement about this.

Yet, the identified challenges are restricting practitioners in their daily work and therefore hindering a more widespread use of machine learning in practice. However, we cannot guarantee that individual highly advanced technology companies do not

already possess sophisticated tools for some of the challenges which we discuss in the following. With this discussion, we want to raise awareness for the need of standardized solutions, which are easily applicable within companies with differing machine learning maturity levels.

Pre-Deployment

Data in general is a major challenge during model development in practice as well as in literature. Data collection and preprocessing requires the majority of time during machine learning projects. Data is often widely spread across the information systems of a company, is unstructured, and in a bad quality. Transforming data to the proper format usually requires a lot of manual work. The interviews specifically referred to data management with complicated access rights as a substantial challenge. Researchers typically are not confronted with this issue because they work with predefined datasets. Additionally, knowledge of several people (e.g., domain experts) needs to be merged to properly understand the data and raise the quality level of the data in practice. Furthermore, several interview partners referred to projects which were discarded after project start because of poor data conditions. This clearly indicates the critical importance of an adequate data structure as well as an appropriate data processing. Major problems in this category can easily jeopardize an entire machine learning project.

General solutions for the automation of data preprocessing and data structuring were directly requested by several of the interviewed experts. The problem has also been mentioned for decades in literature (e.g., Spangler et al. (2000)). There is a clear need for tools supporting the whole data pipeline. Such tools could also support the faster evolution of machine learning techniques across different industries and application fields.

The selection of suitable machine learning algorithms and their improvements was named as challenging in both interviews and literature. However, several experts also noted that researchers too often focus on improving algorithms by small percentage points on statistical metrics while at the same time loosing sight of the complexity in real application domain. This problem has been stated before in literature, but we want to emphasize that researchers should also proof the applicability of their work in real-world environments. In that sense, after performing the interviews, we regard the identification and selection of the single best machine learning model as less critical. With an adequate preprocessing and a parameter optimization, a

prediction model will perform well enough to bring a machine learning project to a successful end.

Literature, in contrast to the interviews, also refers to the problem of technical debt. This is certainly a challenge in practice, however it is less critical compared to other challenges since this issue usually can be solved with a corresponding time investment. Encrypted training data is another challenge only mentioned in literature. This might have not been a challenge so far in practice because encrypted data require rather sophisticated machine learning approaches. Currently, due to the novelty of deploying solutions, many companies still address the easiest and most promising use cases.

Deployment

Deploying machine learning models often is still a challenging task. This is also reflected by the fact that solutions in practice are often highly individual and require a lot of manual work. There are almost no standardized solutions for machine learning infrastructures in many domains. Hence, an individual infrastructure has to be built for many projects which is severely complicating the deployment. Unsuitable or missing infrastructure is a significant challenge for any machine learning project. If no proper deployment infrastructure can be set up by the project team, the entire project is prone to failure. Otherwise, infrastructure issues still will significantly extend the timeline of a project due to long-lasting investment decisions, especially in larger companies.

Ongoing validation and data drifts are common challenges for deployed models. However, little automated strategies are available for handling these problems during deployment. The validation of deployed models is done with manual sanity checks in most cases. These often require the combined knowledge of data scientists and domain experts, which makes it a time consuming and complicated task. Data drifts are automatically detected by some models though, but are usually handled by manual model adjustments.

Automated model updates and adaptations are therefore a challenge which requires further research. This will either simplify and fasten the retraining process or even lead to tools which completely handle concept drifts autonomously without any human intervention. It is critical to develop a proper strategy to ensure the long-term validity of a deployed model already during the initial development of the machine learning model. Otherwise, the project is likely to fail to meet the performance

expectations over time and customers with less technical experience might be disappointed and therefore be less open to new machine learning projects.

Proper infrastructure as well as ongoing validation both increase the robustness of deployed machine learning models in general which is widely confirmed as challenge during the interview study. Model robustness can also be enhanced by the algorithm development itself. Therefore, new or adapted algorithms should not only exhibit an increase in performance metrics, but also a higher level of robustness when confronted with erroneous data.

Non-Technical Challenges

Many machine learning projects are also considerably restricted by non-technical challenges. Transparency is indispensable for successful machine learning solutions and is often specifically demanded by customers and a proper understanding of model results is a necessary prerequisite for trust in machine learning models. Only trusted results will be considered for evaluating important decisions. Even though literature has extensively argued for more transparency, little progress has been achieved. This is especially true for deep neural networks which so far are very little explainable. Practitioners often use advanced visualization tools to increase transparency.

Creating more real-world value of machine learning solutions is an important challenge according to literature. Results from the interviews clearly support this statement. However, research papers usually are rather vague what real-world value actually means. During our interviews, we discovered that this challenges can be viewed from 4 different aspects:

First, it is necessary to express machine learning model results in terms of real-world business value and not in statistical metrics. In practice, evaluation metrics are usually defined individually for every machine learning projects and those metrics translate prediction results into important customer KPIs. However, this is very time-consuming and standardized real-world metrics could facilitate this process.

Second, a proper expectation management with the customers during a machine learning project is crucial. Many customers are inexperienced in the field of machine learning which is why they cannot realistically assess what machine learning is able to accomplish. It is crucial that the project objectives are reasonably defined before any technical experiments start and that these objectives are also communicated appropriately to all stakeholders. These challenges have not been mentioned in

literature so far, probably because researchers usually do not work in such complex project environments. However, a general framework for depicting the value and potential for economic applications of machine learning with corresponding business impact is in our opinion a valid research goal for resolving this challenge.

Third, the communication with the customer as well as the comprehensible explanation of machine learning model results is important. Customers do not only want to understand the effect of machine learning results on their KPIs but they also want understand the influence of different features on the prediction results. In practice, this leads to the application of rather simple algorithms, even though more complex models usually easily outperform those. However, customers are often willing to accept lower performance in exchange for higher transparency.

Fourth, creating valuable digital services or products based on machine learning model results is still quite challenging. It is difficult to convince people to pay for newly created services which are entirely based on machine learning or for existing services that are enhanced with machine learning capabilities.

Expectation management, an adequate customer communication and the creation of valuable digital services based on machine learning are all critical challenges with regard to a more widespread use of machine learning. Concerning a single project, they will typically only lead to significant delays. However, if relevant stakeholders such as responsible line managers are not convinced by the capabilities of machine learning after the end of a project, this might have long-lasting consequences. Those managers are usually the ones who are identifying and providing use cases suitable for a machine learning application. Furthermore, they normally also provide the necessary funding for such a project. This means that if those stakeholders are not satisfied with the results after the execution of a project, success in future projects will be less likely.

Many of the introduced challenges actually go beyond the actual deployment of machine learning solutions. However, CRISP-DM (Figure 4.1) as a standard process model ends with the evaluation of the overall project after deployment. Therefore, after having analyzed this large set of challenges, we argue for an extension of CRISP-DM because many activities such as an appropriate transfer of machine learning results or the ongoing monitoring and adaptation due to data drifts is not considered so far.

4.6 Conclusion and Outlook

The application of machine learning has spurred many new technological developments in both research and industry over the past years. However, many questions with regard to the application of machine learning in real-world applications are still unanswered. In this work, we identify typical challenges that are hindering machine learning practitioners in their daily work. We conduct a structured literature review as well as semi-structured interviews with 11 machine learning practitioners working in different industries.

Compared to publications addressing machine learning in a scientific context, our results show that practitioners do not only face traditional challenges such as data quality and data preprocessing, but are also confronted with a variety of additional problems during the deployment of machine learning solutions. This especially refers to a proper setup of the necessary infrastructure as well as solution strategies for handling concept drift and ensuring long-term validity of machine learning models. We therefore argue for more research with respect to these challenges since they can easily jeopardize the success of an entire project. Furthermore, practitioners frequently encounter non-technical issues such as the expectation management of customers (e.g., managers or non-technical employees) with regard to the deployed solutions as well as a proper communication of the results. Frameworks for depicting the value of machine learning can be a valuable resource in that case and could therefore be a valid research contribution.

Our research generates several contributions to the field of machine learning. First, we provide an overview of challenges of machine learning projects based on a structured literature review. These challenges are organized along the categories pre-deployment, deployment and non-technical challenges. Furthermore, we identified 6 overarching clusters of challenges: Data structure, Implementation, Infrastructure, Governance, Customer relation, Economic implications. Second, we provide an overview over challenges that machine learning practitioners are confronted with during their daily work (c.f. Figure 4.3). Based on both overviews and literature, we perform a comparative analysis, thereby, identifying similarities and differences between the challenges mentioned in literature, originating from a scientific context, and the practical barriers, which were identified through the interview analysis. These results have both implications for academia and industry. One the one hand, the total overview of identified challenges may be used to develop more realistic machine learning models in academia and provides guidance for future research.

On the other hand, it serves as guidance for practitioners in the implementation of machine learning models.

Besides these contributions, our work faces a set of limitations. First, we conducted a limited amount of eleven interviews with machine learning practitioners. Furthermore, due to the availability of suitable interview partners, we could not cover all industry sectors with several interview partners. Nevertheless, we are confident about the completeness and validity of our results, as we did not encounter new challenges with the inclusion of new interview cases. Second, most of our interview partners are currently working in Germany, which might lead to a certain bias in our results. Third, due to the chosen qualitative approach, only limited statements can be made about the severity of one challenge in comparison to another, as well as about the prioritization of the different research needs.

Future work could overcome these shortcomings by performing an interview study with an international sample and could also identify corresponding best practices. During our study, we also realized that there are large differences in the perception of machine learning between experts and involved company managers. Therefore, a subsequent interview study with machine learning experts as well as with company managers would surely generate valuable insights. Additionally, based on these results, a larger quantitative study (e.g., survey-based) could be initiated and performed. This would allow the derivation of quantitative findings and the identification of the magnitude and severity of each challenge as well as the corresponding research need. Those findings could subsequently be used to derive holistic research priorities which promote the general progress of the field as a whole. In general, machine learning as a field has rapidly evolved over the past years. Therefore, it is necessary to continuously align the challenges occurring in the practical application with research pursued in academia.

Related literature has noted before that large parts of machine learning research are too narrowly focused on optimizing performance on benchmark datasets while not creating sufficient real-word value (Rudin & Wagstaff, 2014). With our work, we want to initiate discussions and projects with the aim of closing the gap between academic and practical application. Solutions for the identified research needs can help to strengthen the practical implications of machine learning solutions.

Part III

Challenges for the Application of Concept Drift Handling

5 Handling Concept Drift for Predictions in Business Process Mining[1]

5.1 Introduction

Machine learning plays a major role in the recent developments of artificial intelligence (Kühl et al., 2019). It is widely considered to be one of the most disruptive technologies in the last decades. Its fast progress is fueled by both the development of new learning algorithms and the huge availability of low-cost computation and data (Jordan & Mitchell, 2015). Machine learning is applied across all sectors and in all functional business areas, such as research and development, marketing or finance (Chen et al., 2012). Many companies rely on machine learning models for offering new services or for improving their existing ones (Schüritz & Satzger, 2016). As Davenport (2006) has shown, companies leveraging their data sources achieve a substantial competitive advantage. Especially in the area of services, there seems to be large untapped potential in both, research and practice (Ching, 2018; Ostrom et al., 2015).

To address this promising gap, predictive services offer the possibilities to implement machine learning into different application fields (Baier, Kühl, et al., 2019). Typically, techniques of supervised machine learning provide the basis for such predictive services (Jordan & Mitchell, 2015) which are trained by using historical data of input features and a label. Subsequently, the model is used to continuously compute predictions on a stream of incoming data. However, data streams typically change over time. This is one of the major challenges for applying machine learning in practice (Baier, Jöhren, et al., 2019) since the prediction quality is very sensitive to the input data (Tsymbal, 2004). Therefore, the problem of changing data stream

[1] This chapter comprises an article that was published as: Baier, L., Reimold, J., & Kühl, N. (2020). Handling Concept Drift for Predictions in Business Process Mining. *Proceedings of IEEE 22nd Conference on Business Informatics (CBI)*, Antwerp, Belgium, pp. 76-83. https://doi.org/10.1109/CBI49978.2020.00016. Note: The abstract has been removed. Tables and figures were reformatted, and newly referenced to fit the structure of the thesis. Chapter, section and research question numbering and respective cross-references were modified. Formatting and reference style was adapted and references were integrated into the overall references section of this thesis.

over time has been examined under the term “concept drift” (Widmer & Kubat, 1996).

Usual strategies for handling concept drift rely on dedicated drift detection algorithms (Gama et al., 2014). As soon as a drift is detected, the corresponding machine learning model will be retrained. However, it remains an open research question which data instances for the retraining of the machine learning model should be applied (e.g., data before or after the detection). Therefore, we aim at systematically examining the difference between different retraining options which is expressed in RQ B.1.

> **Research Question B.1**
> Which data should be used for the retraining of a machine learning model when a concept drift is detected?

Subsequently, we apply our findings of RQ B.1 in a real-life use case in business process mining, a typical example of a predictive service. Business process management in general, and business process mining in particular, have received a lot of attention recently in top management because it improves decision making in organizations (Rosemann & vom Brocke, 2015; Van Der Aalst et al., 2016). New applications are extended by the use of predictive analytics (Zur Muehlen & Shapiro, 2015). Since business processes are inherently dynamic, those new features are largely exposed to concept drift (van der Aalst et al., 2012). This requires the adaptation of existing methods to ensure their validity over time. Therefore, we want to examine the effects of the different data options on this use case which is regularly confronted with concept drift in the second research question.

> **Research Question B.2**
> What are the effects of the different retraining options in a real-life use case in business process mining?

The remainder of the paper is structured as follows: Section 5.2 presents related work on which we base our research. Section 5.3 introduces different aspects which can be considered for the retraining of a machine learning model after detection of a drift. Section 5.4 presents the chosen use case as well as the evaluation of the different options discussed in the previous section. The final section discusses our results, describes theoretical and managerial implications, acknowledges limitations and outlines future research.

5.2 Related Work

This section gives a brief overview of related work about concept drift as well as its detection. Furthermore, related work regarding process mining is introduced.

5.2.1 Concept Drift

Machine learning can create ongoing value when the corresponding prediction models are deployed in connected information systems and deliver ongoing recommendations on continuous data streams. However, data streams usually change and evolve over time. This is also reflected in changes in the underlying probability distribution or their data structures (Aggarwal et al., 2003). The challenge of changing data streams for machine learning tasks has been described with the term "concept drift" (Widmer & Kubat, 1996) in computer science. A concept $p(X, y)$ is defined as the joint probability distribution of a set of input features X and the corresponding label y (Gama et al., 2014). In real applications, concepts often change with time (Tsymbal, 2004). This change can be expressed in a mathematical definition as follows (Gama et al., 2014):

$$\exists X : p_{t0}(X, y) \neq p_{t1}(X, y)$$

This definition explains concept drift as the change in the joint probability distribution between two time points t_0 and t_1. Therefore, machine learning models built on previous data (in t_0) might not be suitable for making predictions on new incoming data (in t_1). This change requires the frequent adaptation of the prediction approach.

Changes in the incoming data stream can depend on a multitude of different internal or external influences. Usually, it is impossible to measure all of those possible confounding factors in an environment—which is why this information cannot be included in the predictive features of a ML model. Those factors are often considered as "hidden context" of a predictive model (Widmer & Kubat, 1996). Concept drift is usually classified into the following categories (Žliobaitė, 2010): Abrupt or sudden concept drift where data structures change very quickly (e.g., sensor failure), gradual and incremental concept drift (e.g., change in customers' buying preferences) or seasonal and reoccurring drifts (e.g., A/C sales in summer). There exists also a more fine-grained taxonomy (Webb et al., 2016) which also considers the magnitude of the drift for instance.

A wide variety of approaches for the handling of concept drifts has been proposed (Gama et al., 2014). However, most approaches rely on an explicit drift detection which detects changes in the data distribution and triggers corresponding adaptations. Two of the most popular algorithms are Page-Hinkley (Page, 1954) and ADWIN (Bifet & Gavaldà, 2007). Page-Hinkley works by continuously monitoring an input variable (e.g., the input data or the prediction accuracy). As soon as the variable differs significantly from its historical average, a change is flagged. ADWIN, in contrast, is a change detector which relies on two detection windows. As soon as the means of those two windows are distinct enough, a change alert is triggered, and the older window is dropped.

5.2.2 Process Mining

Business process mining is a research discipline that originates from business process modeling and analysis on the one side and data mining on the other side (Manoj Kumar et al., 2015). The goal of process mining is to discover, monitor and improve operational processes by extracting data from event logs (Van Der Aalst, 2011). This way, business processes are analyzed in the way as they are really executed (Van Der Aalst & Weijters, 2004). These event logs can be created by extracting the digital traces of business processes that are stored in today's information systems, e.g., ERP or CRM systems (Van Der Aalst et al., 2007). The minimum information needed for an event log is therefore a unique CaseID to identify and differentiate each case and an event with relating timestamp to define the activity of the process. This combination is important, so that the real sequence of the events can be ensured.

Process mining can be differentiated into three types (van der Aalst et al., 2012) where the first type is *discovery*. After extraction of the event logs, a process model can be built. This also allows to understand different variants of business processes (Dumas et al., 2013). The second type is *conformance*. In this case, existing process models can be compared with an event log of the same process and discrepancies between both can be discovered. The third type relates to *enhancement* where existing process models are extended. This can also refer to operational support where predictions and recommendations based on prediction models from historic information can be used to optimize running cases (Van Der Aalst et al., 2011). An application could be the prediction of the remaining time of a case (Verenich et al., 2019) or the prediction of the next executed activity in a case (Márquez-Chamorro et al., 2017).Furthermore, there are approaches predicting whether a case will be completed (Di Francescomarino et al., 2017). With such predictions, the organizational procedures can be optimized, and personnel planning is more

accurate. For instance, it can be very valuable for a customer to know the remaining process time of his insurance claim or when his product order will arrive.

A very important challenge in process mining is the occurrence of concept drift (van der Aalst et al., 2012) which refers to processes that are changing while being analyzed. For instance, the sequence of events can change, e.g., two events that occurred in parallel are now occurring one after another. Processes may change due to a variety of reasons, from seasonal effects over market changes to organizational adjustments. Business processes are inherently dynamic over time and therefore prone to change. Nevertheless, concept drift research in business process mining is rather scarce. Sudden concept drift in process mining, such as rearranging or replacing activities, has been examined (Manoj Kumar et al., 2015). The authors propose to detect those drifts by computing correlation between event classes. Another approach proposes a framework which computes dedicated features on the event logs and subsequently compares those features over different windows to detect concept drift (Bose et al., 2014). In this context, this method to detect drifts is similar to traditional concept drift approaches described in subsection 5.2.1. More advanced options use an adaptive approach based on a Chi-square test which also allows to detect different types of process drift (Maaradji et al., 2017). Other research aims at better understanding the type or the degree of change (Yeshchenko et al., 2019) or providing more robustness to process drift detection methods (Ostovar et al., 2020).

The approaches described above focus on concept drift in the type of event or their order in a process. This is related to the first type of process mining (*discovery*) that focuses on deriving a process model. The ultimate objective of this analysis is to identify and better understand the activities that trigger process drift in the first place.

However, this analysis does not contain any predictive component. Existing work has not yet considered concept drift in the *enhancement* type of process mining where predictions based on machine learning are computed to optimize operations (Van Der Aalst et al., 2011). Compared to previous work, this also requires strategies for an adaption of prediction models over time.

5.3 Data Selection for Retraining

This section introduces the two different learning modes for machine learning models and provides an overview on which data can be used for the retraining of a model if the training process has to be started from the beginning.

5.3.1 Learning Mode

In the context of data streams and ongoing predictions, two learning modes for machine learning models can be differentiated: retraining and incremental learning (Baier, Kühl, et al., 2019).

The method of retraining is illustrated in Figure 5.1. The figure shows that in the beginning the model is trained on an initial batch of data. After the initial model has been trained, new incoming data instances X result in predictions y (e.g., y_1 in Figure 5.1). This happens iteratively for every new data instance in the data stream until the drift detection method issues an alert which requires an adaptation of the prediction model. Correspondingly, the old model is discarded, and a completely new prediction model is trained which is subsequently applied to every incoming data instance (e.g., the new prediction model after retraining is applied for the first time by predicting y_{378} and the following data instances in Figure 5.1).

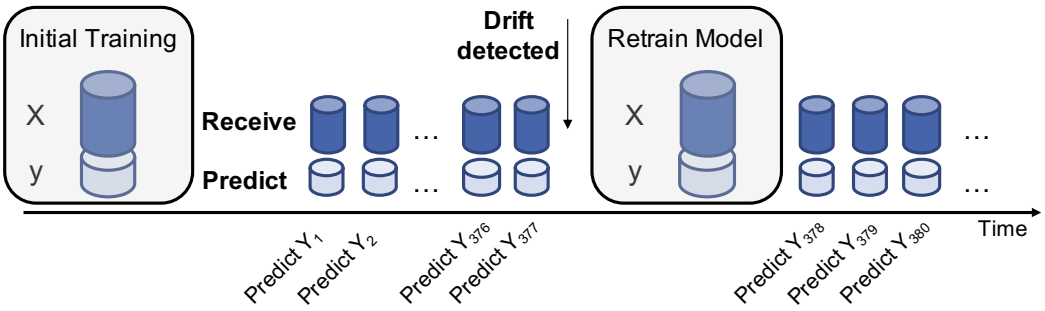


Fig. 5.1.: Depiction of learning mode retraining.

Incremental learning, in contrast, works by continuously updating the prediction model. Comparably, the starting model is trained on an initial data set. When new data instances arrive, the model issues a prediction. However, as soon as the true target label of this data instance is known, this information is used to incrementally improve the prediction model. The main advantage of this approach is that every new labeled instance arriving will be used for model improvement and thus, the model automatically adapts to changing concepts. This approach is comparable

to a sliding window approach. In general, the incremental updates will not be computed after a single new data instance has arrived but rather after the reception of a small batch of data instances (e.g., 10 or 20). This reduces the computational complexity. Unfortunately, only few machine learning algorithms such as Naïve Bayes, Neural Networks or Hoeffding Trees (Pfahringer et al., 2007; Žliobaitė et al., 2016) implement the opportunity for incremental updates.

Despite the continuous updates of the prediction model, this approach might be confronted with degrading performance over time. For instance, the incremental updates of the model cannot adapt to very quick changes which occur during sudden concept drifts. In this case, it might be also necessary to discard the current model and train a new model. This would depict a combination of both learning modes retraining and incremental updates.

5.3.2 Data Selection for Retraining of the Machine Learning Model

In case of concept drift, the previous model will be discarded, and a new model is trained as depicted in Figure 5.1. However, when implementing this approach, we need to select the data that is used for the retraining of the machine learning model. So far, literature does not provide any knowledge on which data of the data stream should be used for the retraining of the prediction model. Therefore, we implement and evaluate three different data selection strategies which we call *next*, *mixed* and *last*. The difference between these approaches is depicted in Figure 5.2.

The approach *next* is displayed in the upper part of Figure 5.2. As soon as a concept drift is detected, the model collects the next batch of instances with corresponding labels (e.g., two new data instances in the figure). When this next batch is complete, the retraining is started and subsequently the new model is applied. This also means that the previous model is used to predict the next batch after the concept drift since it is also necessary to issue predictions for those instances (and the new model has not been learned yet). The intuition guiding this approach is that data following a concept drift, complies with the new concept and is therefore an optimal basis for a new model.

The other approaches *mixed* and *last* are also displayed in Figure 5.2. In case of the *mixed* approach, the model retraining relies on data from before and also after the detection. Compared to the first approach (*next*), the new model can be applied faster since it requires less data after the concept drift. The *last* approach entirely relies on data which was acquired before the concept drift detection alert. This

means that the new prediction model will be applied right on the next data instances after the detection of a drift. This approach might work well because drift detection algorithms usually work with a slight delay. Therefore, the data batch before the alert might already belong to the new concept.

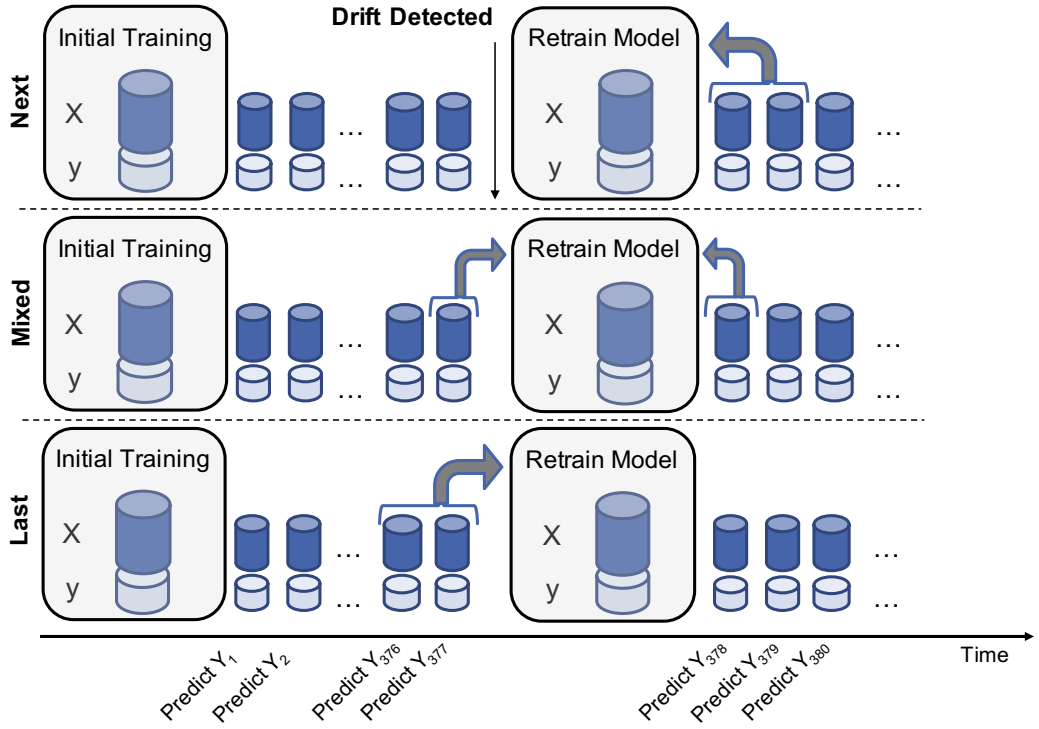


Fig. 5.2.: Three different approaches for retraining of model.

During the application of our use case, we aim to systematically test all three approaches in order to quantify the differences between those and also to give recommendations for future implementations.

5.4 Use Case in Process Mining

A process mining solution provider gives us access to a data set of the purchase to pay (P2P) service process of a large German company. This process contains all activities related to the procurement of a product or service. A simplified P2P process starts with the creation of a purchase order and is followed by the reception of the respective goods by the logistics department and the invoice which is then processed over various financial departments in the company. An exemplary process of this P2P process can be seen in Figure 5.3.

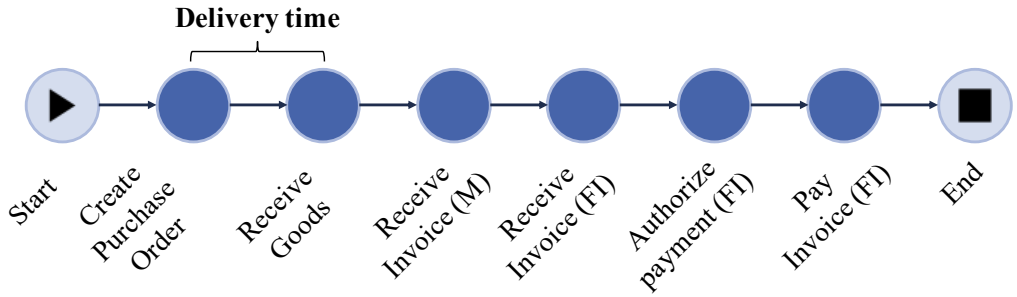


Fig. 5.3.: Typical process variant for a P2P process.

In this use case, we want to predict the throughput time or delivery time (marked in bold) between the creation time of the purchase order and the reception of the goods. This information is quite important for the company since all subsequent process steps such as production can be optimized, and significant cost savings can be realized. The data is extracted from the business intelligence platform Qlik and then preprocessed in Python. The foundation of the data set is an event log that is enriched with numerous additional attributes to fully describe the process. The attributes are anonymized and transformed to ensure that the data is not retraceable. In total, we receive data about 70,774 purchase transactions from 2016 until 2018 which we can use to train and evaluate the machine learning approach. Importantly, those transactions are displayed in chronological order, which is a necessary prerequisite for an analysis of concept drift over time.

We use the package scikit-mulitflow (Montiel et al., 2018) as the basis of our analysis since it extends the machine learning package scikit-learn with a stream data framework. It allows to process data sets and simulate them as a data stream. Furthermore, different concept drift detectors are implemented and can be evaluated. We extend the package by implementing the different training modes (*last*, *mixed*, *next*) which we discussed in Section 5.3.

5.4.1 Data Analysis

We first perform an exploratory data analysis to analyze the available features and build a predictive model that can be used for the analysis of concept drift in process mining. Table 5.1 gives an overview on available features of the data set. Categorical features are one-hot-encoded for the subsequent data processing. *Material class* refers to the product category of the purchased product. Regarding this feature, we only use the first four numbers of the material class in order to reduce the number

of different categories resulting in 123 different categories in total. Furthermore, we have information about the *purchase order value*. The purchase order value is an important feature for our endeavor since it is a clear indicator of the relevance of the respective purchase order for the company. However, the distribution of the order value is highly skewed which might pose a problem for the prediction model. Therefore, the values are transformed with a Box-Cox transformation (Box & Cox, 1964) into a gaussian distribution.

Tab. 5.1.: Overview of predictive features.

Feature	**Variable type**	**Number of items/ Range of values**
Material class	categorical	123
Document type	categorical	7
Plant code	categorical	4
Purchase order value	numerical	1 – 458,079
Supplier	categorical	799
Bank country	categorical	18
Supplier country	categorical	14
Purchasing group	categorical	75
Throughput time [h] (Target)	numerical	1 – 120,000

Other features included in the data set are the *country of the bank* were the payment is executed and the *document type* of the purchase order. The *document type* includes information about different ways to create a purchase order: e.g., the order is created manually by an employee in the purchasing department or is based on existing long-time contracts. Other options include the automatic creation by an MRP-system. The *country of the supplier* is also relevant for the analysis. Obviously, a purchasing process requires more time if the supplier is located in another country because this leads to additional steps during the sales process such as customs papers, currency conversion or additional insurance of the transport. The feature *plant code* stores information about the plant which initiated the purchase process. *Purchasing group* is the department or group at which the purchase order is created and processed. Furthermore, we also have information about the *supplier* itself who is distributing the requested product.

The *target variable* in this use case is the *throughput time* or delivery time of a purchase order. This refers to the amount of time between the first two steps depicted in Figure 5.3. By considering the delivery time, we ensure that the start of the purchase process is considered as well as the most important event for production and workforce scheduling, namely the arrival of the ordered goods. The prediction

of the estimated arrival time of a product is important because planning processes can be optimized with this information. This might result in significant cost savings as well as the minimization of production time due to the optimization of waiting time.

A histogram of the throughput time can be seen in Figure 5.4. For approximately 50% of all purchase orders, respective products and goods are received within 14 days (<336h). Regarding the remaining purchase orders, another 25% of those have a delivery time within 60 days. The other purchase orders even have a larger delivery time, up to 537 days.

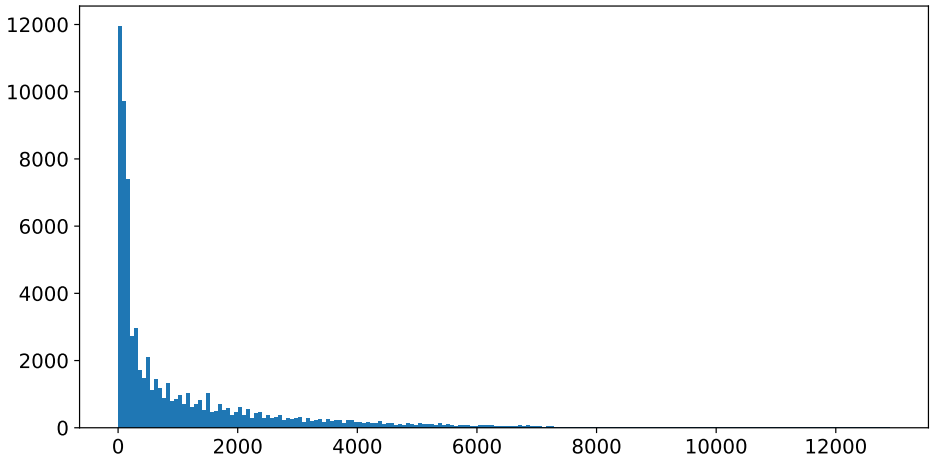


Fig. 5.4.: Histogram of the throughput time [h].

Due to the challenging distribution of the target variable, we transform the use case into a multi-class classification problem. Although this leads to an abstraction and loss of information, this step is meaningful for an initial analysis of the use case. To transform the target variable, all purchase orders are divided into three equally sized classes of throughput times as can be seen in Table 5.2. Therefore, the first class contains purchase orders with a delivery time of up to 6 days. The second class contains purchase orders with a delivery time between 7 and 39 days and the last class contains all cases for which the delivery takes more than 40 days. We train a machine learning model which predicts whether a purchase order will belong to the short, medium or large throughput time class.

Tab. 5.2.: Overview of multi-class target variable.

	Short time	**Medium time**	**Large time**
Delivery time	0 – 6 days	7 – 39 days	$>$ 39 days

5.4.2 Evaluation of Prediction

We first perform a pretest with various machine learning algorithms in their standard parameter configuration (Pedregosa et al., 2011): Naïve Bayes, Neural Network, Support Vector Machine and Decision Tree. The results depicted in Table 5.3 are computed by performing a 70%-30% train-test-split on the first 2,000 data instances. We assume that those data instances all belong to the same concept as there is no significant change observable in the input data. Therefore, we can safely apply the machine learning algorithms without considering and handling concept drift. Note that prediction performance on later parts of the data set might be lower due to the challenges induced by concept drift.

Tab. 5.3.: Pretest with different models on subset of data.

Model	Accuracy
Naïve Bayes	0.767
Neural Network	0.805
Support Vector Machine	0.697
Decision Tree	0.740

Naïve Bayes, Neural Networks and Decision Trees all achieve similar accuracy values. We choose Naïve Bayes classifier as the prediction algorithm which is due to two reasons: First, Naïve Bayes implements incremental learning which allows incremental uptates of the prediction model. Second, computational complexity of Naïve Bayes is rather low compared to other machine learning algorithms which allows frequent retraining of the model without the necessity for a large computational infrastructure.

Our work mainly focuses on the quantification and handling of concept drift. However, we do not have any knowledge whether there are any drifts at all in the data set or at which point in time they are occurring. Therefore, first of all, we analyze the impact of concept drift by applying a Naïve Bayes classification without any concept drift detection method—called "static model"—to the entire data set of 70,774 data instances. Subsequently, we apply Naïve Bayes classifier in combination with a Page-Hinkley test and ADWIN as drift detection methods. As evaluation metric, we use the accuracy by measuring how often the algorithm predicts the appropriate throughput time class. This metric is chosen since the instances are distributed equally over all three target classes.

The course of the accuracy of the static model without concept drift detection and incremental learning can be seen in Figure 5.5. The first 2,000 data instances are used for the initial training. Subsequently, we compute the first predictions and the

accuracy level moves at around 0.7. Then, there is a first drop in accuracy after approximately 25,000 instances. However, the prediction performance recovers to around 0.7 shortly after. Subsequently, after approximately 35,000 instances, the prediction quality of the model decreases significantly. Supposedly, a concept drift has occurred because the model that is only trained on an initial data batch does not issue any useful prediction anymore. The accuracy over all predictions reaches 0.5400.

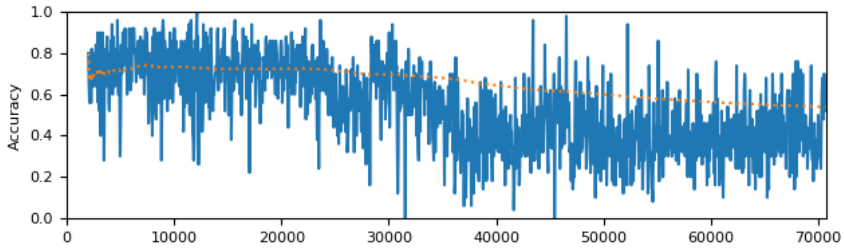


Fig. 5.5.: Accuracy of Naïve Bayes without retraining and no drift detection method.

As usual, it is difficult to determine the underlying reasons for this concept drift with certainty (Žliobaitė et al., 2016). However, after a thorough analysis of additional data—which is not available the moment when the prediction is computed—we identify a possible explanation. The feature *automation* contains information about the percentage of process steps in the entire P2P process which are executed automatically by corresponding information systems, while the other steps are executed manually. Thereby, the feature *automation* contains information about the level of automation in all processes. In order to analyze the development of this feature over time, we compute and plot a rolling mean ($window = 1000$) of this feature which is depicted in Figure 5.6.

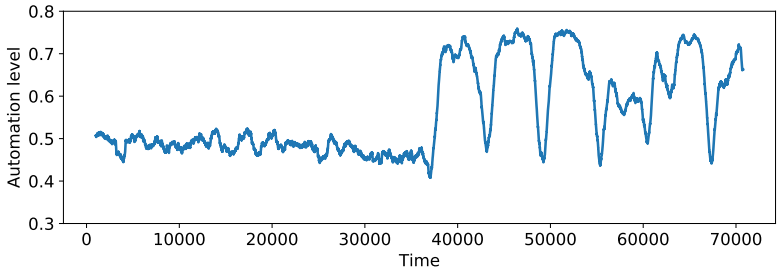


Fig. 5.6.: Rolling mean (window 1000) of feature *automation*.

At first, the rate of automation is rather stable before it rises abruptly and then fluctuates at a higher level. This plot clearly indicates on how the automation rate in the organization increases over time and thus, this may be one of the causes for concept drift and according changes in product delivery times. The sudden rise in automation maps rather well to the decrease in prediction accuracy in Figure 5.5.

Relating to section 5.2, this abrupt change can be seen as a sudden concept drift. Since this feature is not known at the time of prediction, it can be interpreted as a hidden context influencing the prediction.

Due to the detected drift, we apply a Page-Hinkley test as concept drift detection method in combination with the Naïve Bayes classifier. In case of drift, the model is retrained. The course of the accuracy of the model can be seen in Figure 5.7.

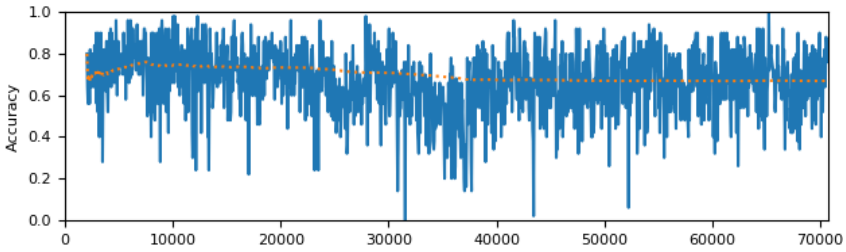


Fig. 5.7.: Accuracy of Naïve Bayes with Page-Hinkley.

At the beginning, the figure looks similar to the model without drift detection (Figure 5.5 above). After approximately 35,000 instances, this model performs better because the drift is detected, and a retraining of the Naïve Bayes is triggered. The accuracy rises again and then stays same level with its corresponding fluctuations leading to an overall accuracy of 0.6732 (see Table 5.4). This is equivalent to a performance increase of 24%. Furthermore, we extend this approach by activating incremental learning. This means that the model is constantly updated with new training data after it has issued prediction for those data. The application of incremental learning alone leads to a performance of 0.6717. With both retraining and incremental learning, the overall prediction accuracy reaches 0.6938.

Tab. 5.4.: Performance of drift detection strategies on process mining data set.

Change detection	Incremental learning	Accuracy	Performance increase
None (baseline)	No	0.5400	– (baseline)
None	Yes	0.6717	24.39%
Yes (Page-Hinkley)	No	0.6732	24.67%
Yes (Page-Hinkley)	Yes	0.6938	28.48%

We perform a grid search on the first 10,000 data instances in order to optimize the parameters of the drift detection method ADWIN ($\delta = 0.001$) and Page-Hinkley ($\lambda = 0.6$). With those parameters, we evaluate the different data selection strategies as discussed in Section 5.3. Table 5.5 depicts the accuracy score of a Naïve Bayes classifier with incremental learning in combination with a Page-Hinkley test or ADWIN as drift detection. Furthermore, we examine the influence of four different

batch sizes (500, 1000, 2000, 5000) on the overall prediction accuracy. This refers to the amount of data instances which are provided to the model in case of retraining. The best results are marked in bold in Table 5.5.

As depicted in the table, the data selection strategy *last* performs always best. For our use case, Page-Hinkley appears to be the more suitable drift detector resulting in higher performance. Interestingly, the prediction accuracy decreases with increasing batch size which might indicate that the approach does not adapt fast enough with larger batches for retraining. Furthermore, the performance difference between the different data selection strategies also rises with the size of the batches. For instance, the difference between *last* and *next* for Page-Hinkley with batch size 500 equals 0.0073 in comparison to 0.0164 for Page-Hinkley with batch size 5000.

Tab. 5.5.: Performance of data selection strategies on process mining data set.

Change detection	Batch size	Incr. learning	Last	Mixed	Next
Page-Hinkley	500	Yes	**0.7010**	0.6961	0.6937
Page-Hinkley	1000	Yes	**0.6965**	0.6920	0.6903
Page-Hinkley	2000	Yes	**0.6938**	0.6845	0.6821
Page-Hinkley	5000	Yes	**0.6842**	0.6757	0.6678
ADWIN	500	Yes	**0.6856**	0.6849	0.6843
ADWIN	1000	Yes	**0.6854**	0.6838	0.6825
ADWIN	2000	Yes	**0.6803**	0.6775	0.6750
ADWIN	5000	Yes	**0.6758**	0.6704	0.6675

In general, the evaluation section clearly shows how the prediction performance can be increased by implementing drift handling strategies. Both, incremental learning as well as drift detection with retraining, have significant influence on the accuracy. Best results are achieved with the combination of both approaches.

5.5 Conclusion

Process mining relies more and more on techniques of machine learning. This work explores the challenge of concept drift for ongoing value creation in process mining. Specifically, we apply a concept drift detection algorithm on a use case which aims at predicting the delivery time for all purchase orders of a company. With this information, the company can optimize its internal service processes. We can show that concept drift handling significantly outperforms a static model in the given use case. Best results are achieved by combining incremental learning with retraining in case of concept drift. Regarding the best training data selection

strategy for retraining, the *last* approach appears to be the best performing option. This means that data scientists should rely on the last collected data batch for the retraining of the prediction model.

The contribution of this paper is twofold. First, we systematically explain and depict the options for training data selection for the retraining of machine learning models in case of concept drift. Second, we apply and evaluate those options in a real-life use case in process mining where we can measure a significant increase in prediction performance from 0.5400 to 0.7010. Regarding the managerial implication, this work clearly shows the importance of a continuous monitoring and adaptation scheme of predictive services in operation. Otherwise, they can quickly lose their validity and corresponding service offerings will not deliver expected benefits.

However, more research is required to understand the full effects of concept drift and the best strategies to deal with this problem. This work only describes and evaluates three options for the training data selection in case of retraining. Future work needs to evaluate more sophisticated approaches. Additional limitations regarding the use case arise through the transformation of the target variable from a regression problem into a multi-class classification problem. Furthermore, we only evaluate the data selection on one use case. More general recommendations could be derived by applying those options onto more use cases and benchmark data sets.

This paper clearly shows the importance of constant monitoring of predictive services for the detection of concept drifts. Frequent retraining and adaptations of a machine learning model are necessary requirements to keep and guarantee a high prediction performance. If practitioners consequently implement necessary monitoring activities, the economic benefits of predictive services and supervised machine learning solutions can still even be increased.

6 Preserving Validity of Predictive Services over Time[1]

6.1 Introduction

Due to the large increase of data in recent years, various industries are trying to reap the benefits of this new resource for their service offerings. Machine learning is playing an important role in nearly all fields of business, ranging from marketing over governmental tasks to scientific-, health- and security-related applications (Chen et al., 2012). Many companies rely on machine learning models deployed in their information systems for increasing the efficiency of their processes (Schüritz & Satzger, 2016) or for offering new services (Dinges et al., 2015). As Davenport (2006) describes, companies which are able to leverage their data sources through analytical tools achieve a substantial competitive advantage.

However, it is worth regarding how such predictive services based on machine learning are built, deployed and executed in the long run. Traditionally, supervised machine learning models are trained using historical data containing input features and a corresponding target (Hirt et al., 2017). Subsequently, the model is used to continuously make predictions for a specific service (e.g., the failure of a machine) on a stream of unseen incoming data. We define such a service as a "predictive service". However, data streams typically evolve over time and thus, their data structure or the underlying probability distribution changes (Aggarwal et al., 2003). This depicts a challenge since supervised machine learning models are very sensitive to changes in their input data, e.g., to the adjustment of production parameters (Tsymbal, 2004). Even small deviations can have significant impact on the deployed model—drastically influencing its prediction performance and the utility of the

[1]This chapter comprises an article that was published as: Baier, L.; Kühl, N.; Satzger, G. (2019). How to Cope with Change? Preserving Validity of Predictive Services over Time. *Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS)*, Maui, Hawaii, USA. https://doi.org/10.24251/HICSS.2019.133. Note: The abstract has been removed. Tables and figures were reformatted, and newly referenced to fit the structure of the thesis. Chapter, section and research question numbering and respective cross-references were modified. Formatting and reference style was adapted and references were integrated into the overall references section of this thesis.

predictive service (Russell et al., 2015). However, it is difficult to detect this change in the input data and, furthermore, to adapt the model accordingly (Tsymbal, 2004). In the field of computer science, the phenomenon of a changing relation over time between the input features and the target labels is predominantly called "concept drift" (Gama et al., 2014).

An example for an application with evolving data over time is a predictive service which monitors the output quality in a chemical production process and predicts corresponding failures (Žliobaitė et al., 2016). Such a predictive service relies on the input data generated by the sensors that the production machine is equipped with. Sensors wear out over time (Kadlec & Gabrys, 2011) and the resulting measurements change accordingly, leading to different input data. However—without the necessary precautions—a machine learning model is not prepared for this change since this pattern has not been observed before in the training set. Thus, meaningful quality predictions are impossible to make in the long run, and the service does not keep up to its promised validity. Therefore, we define a general research question which guides this research paper:

General Research Question C
How can we design an effective and efficient automated artifact for predictive services, which ensures their long-term validity?

Based on this general research question, we aim to describe the current status of predictive services.

Research Question C.1
How can we distinguish between various forms of existing predictive services with regard to their lifecycle?

For answering this question, this work introduces a definition of predictive services as well as a framework for characterizing predictive services with respect to their validity over time. The framework can be used as a support tool for practitioners during the introduction of a predictive service so that all relevant design options are considered. Furthermore, the framework allows a thorough analysis as well as comparison of existing approaches. In demarcation to existing frameworks such as Gama et al. (2014), our framework includes the setup as well as operation phase of predictive services. Subsequently, we classify available research papers into the framework resulting in a heatmap which serves as a foundation for deriving a

research agenda of valuable research topics: The different availability of true labels during operation as well as methods for domain expert integration.

The remainder of the paper is structured as follows: Section 6.2 presents related work on which we base our research and introduces a definition of predictive services. Section 6.3 presents a framework for characterizing aspects of predictive services and classifies existing practical research on that basis. Section 6.4 introduces research opportunities that are derived from challenges identified in the previous section. Section 6.5 discusses our results, describes theoretical and managerial implications, acknowledges limitations and outlines future research.

6.2 Foundations

To allow for a common understanding, we first introduce the theoretical foundations for the examination of the validity of predictive services. We give a brief overview on machine learning for services, followed by an overview of research in computer science that deals with concept drift. Subsequently, we introduce predictive services.

6.2.1 Machine Learning for Services

Machine learning in general has recently received a lot of attention due to the massive flow of available data and increasing computation power. Traditionally, approaches are divided into supervised and unsupervised machine learning (Han et al., 2012). Supervised machine learning depends on labeled examples in the training data, whereas unsupervised machine learning aims at detecting unknown patterns in the data. Most real-word applications of machine learning are of supervised nature (Jordan & Mitchell, 2015; Kotsiantis, 2007). Therefore, we focus on supervised approaches in the following. Well-known application examples are the prediction of a credit rating or the fingerprint matching on current smartphones.

The importance of analytical and machine learning solutions for service science has been highlighted by the introduction of service analytics (Fromm et al., 2012). Service analytics describes the dedicated application of analytical tools such as machine learning on data created in service systems to improve or extend existing service offerings. In this context, continuous data streams over time play an important role. Machine learning is, for instance, applied to monitor click streams on web pages or to monitor events and notifications (Kambatla et al., 2014). All those

examples are confronted with changing data streams over time. Therefore, the next section introduces definitions as well as solutions developed for this challenge from a computer science perspective.

6.2.2 Concept Drift

The computer science community has examined the challenge of changing data streams in machine learning over time under the term "concept drift" (Widmer & Kubat, 1996). A concept p(X,y) is described as the joint probability distribution over a set of input variables X and the label or target variable y. However, "in the real world concepts are often not stable but change with time" (Tsymbal, 2004, p. 1). This leads to the problem that machine learning models built on previous data are not valid anymore for new incoming data which requires regular model updates or retraining. There exists a variety of descriptive definitions of concept drift (Tsymbal, 2004; Widmer & Kubat, 1996; Žliobaitė et al., 2016). A mathematical definition is given by Gama et al. (2014): $p_{t0}(X, y) \neq p_{t1}(X, y)$ The definition states that we are facing a concept drift if there is a difference in the concept at t0 compared to the concept at t1. This change of the joint distribution is challenging for supervised machine learning models since they are typically trained on a fixed initial training set (Tsymbal, 2004). However, if the features and the label of the training set just belong to the concept at t0, the model is only trained to recognize objects of the first concept whereas it does not know how to handle instances belonging to the second concept at t1. Changes in the incoming data stream can depend on many internal or external factors. Therefore, it is intuitive that different types of changes in data streams can be identified. One popular classification of concept drift depicts four different types (Žliobaitė, 2010): Sudden concept drift, incremental concept drift, gradual concept drift and reoccurring concept drift such as seasonal patterns. Webb et al. (2016) also provide a more detailed taxonomy with categories such as drift and concept duration as well as drift magnitude.

Gama et al. (2014) introduce a framework which focuses on algorithmic methods for changing data streams. The framework consists of four categories: Memory, change detection, learning and loss estimation. A description on application-related use cases is given by Žliobaitė et al. (2016). They provide a list of 54 research papers that implement solutions and methods for changing data streams with real data. Concrete use cases which consider the challenges of concept drift occurrence can be clustered into monitoring and control tasks (Ivannikov et al., 2009; Pawling et al., 2007), information management (Delany et al., 2006; Fdez-Riverola et al.,

2007) and analytics and diagnostics tasks (Giacomini & Rossi, 2009; Harries & Horn, 1995).

Based on the foundations in the previous two subsections, we introduce predictive services in the following.

6.2.3 Predictive Services

We define predictive services as services based on predictions that are acquired through the application of supervised machine learning models on data available in its service system environment. Predictive services are fully deployed on a productive IT infrastructure and thereby are constantly issuing new predictions. The final objective can either be the delivery of the prediction itself (e.g., forecast for the market demand for a product) or an action based on the prediction (e.g., the automated adjustment of the production schedule for a product).

We assume that the validity of predictive services can be affected in two ways: First, the environment of the service changes, which influences the resulting data, and thus, the quality of the prediction. This is the case when a sensor on a production machine wears out over time and delivers less reliable results. Second, the application of the service itself affects its predictive power over time. The second case can be illustrated by a predictive policing service indicating the neighborhoods in a city with the most criminal activities. The local police will accordingly reinforce their presence in this area which results in a decreasing criminal statistic over time. This development, however, will invalidate the recommendations of the predictive service which continues to classify this neighborhood as a high-risk area (Perry, 2013). After all, any kind of predictive service is facing the challenge of a changing environment over time; it is just a matter of the time span that is considered.

The example above illustrates the complexity of ensuring the long-term validity of predictive services. Therefore, the problem requires a comprehensive and inter-disciplinary analysis. On the one hand, it is necessary to thoroughly examine the technological side of the problem. On the other hand, the economic side must be also considered, and benefits or downsides of possible solutions must be assessed.

The next section introduces a framework which can be used to set up a predictive service and to prepare it for changes in the data stream in order to guarantee the validity over time.

6.3 Conceptual Framework

The framework can be understood as a tool to support the initiation and implementation of a predictive service. It gives guidance for decisions during the setup phase of a predictive service but also provides solutions for challenges during the operation and use of the predictive service. Furthermore, it allows to differentiate between characteristics of predictive services. At first, we explain the methodology that we applied for the development of the framework. Subsequently, we introduce the framework itself which is split into three parts. The first part relates to necessary design decisions during the setup of a predictive service. The second part displays the algorithmic options for keeping the validity over time. The third part presents characteristics that need to be considered during operation, especially the availability of true labels and the constant evaluation of the service.

6.3.1 Methodology

Gama et al. (2014) provide a taxonomy which explains the different algorithmic options for handling changing data streams. This taxonomy is the basis for our framework and is mainly reflected in the second part (c.f. section 6.3.3). However, their taxonomy is missing design decisions during setup as well as operation of predictive services. The consideration of both phases is (besides the algorithmic methods) crucial for the development of a successful predictive service. Our framework is therefore built as an extension to the prevailing taxonomy.

We developed our framework by a rigorous analysis of existing use cases in research that examine concept drift. We base this analysis on the 54 research papers which are presented in Žliobaitė et al. (2016), as those include papers from a wide range of application tasks. We remove all research papers with unsupervised approaches and those that do not provide sufficient information for in-depth comparisons resulting in 23 remaining research papers. Based on a forward and backward search on this list, we identified 11 additional research papers. In total, we included 34 research papers into our detailed analysis[2]. During the analysis, we iteratively added or

[2]Agrawal et al. (2018), Black and Hickey (2002, 2004), Bose et al. (2014), Delany et al. (2006), Ding and Li (2005), Ekanayake et al. (2009), Fdez-Riverola et al. (2007), Forman (2002), Gago et al. (2007), Harries and Horn (1995), Harries et al. (1998), Huang and Dong (2007), Ivannikov et al. (2009), Kadlec and Gabrys (2011), Klinkenberg (2005), Krawczyk (2017), Kukar (2003), Kurlej and Wozniak (2011), Kurlej and Woźniak (2011), Laghmari et al. (2018), Lebanon and Zhao (2008), Liao et al. (2007), Luo et al. (2007), Mohamad et al. (2016), Mourão et al. (2008), Pawling et al. (2007), Soares and Araújo (2016), Sun et al. (2016), Tsymbal et al. (2008), Widyantoro and Yen (2005), Xu et al. (2017), Zhou et al. (2008), and Žliobaitė et al. (2014)

removed categories as we progressed with the number of research papers. The items are based on existing literature. If we could not identify suitable literature, we added the items based on our analysis of the research papers. The resulting framework needs to be understood as an exploratory tool which still develops over time as new research papers are included.

6.3.2 Setup Decisions for Predictive Service

Before a predictive service can be offered, fundamental decisions about the setup of the service must be made. Table 6.1 depicts the different categories for the setup phase.

Business focus: When designing a predictive service, one of the first steps is to clearly define the business focus. What is the benefit that the service is delivering and who is the final user/customer of it? The customer can either be external (e.g., a service provider offers constant social media analytics to a customer with a tool) or internal (e.g., predictive service is used for the improvement of internal processes) (Schüritz & Satzger, 2016).

Tab. 6.1.: Setup decisions for predictive service.

Business focus	External	Internal	Unknown		
Data input	Structured	Unstructured			
Machine learning task	Regression	Classification			
Domain expert knowledge	Label provision	Feature generation	Model building	Change detection	None
Type of change	Sudden / Abrupt	Incremental / Gradual	Reoccurring	Unknown	

Data input: A differentiation with regard to the data input which is used for the predictive service is necessary. Structured data in form of tables (e.g., Žliobaitė et al. (2014)) can easily be utilized by most machine learning algorithms and change detection approaches. However, unstructured data (e.g., text data (Lebanon & Zhao, 2008)) is more complex to process and requires more advanced handling techniques, especially for the change detection.

Machine learning task: A clear definition of the relevant machine learning task behind the predictive service is indispensable. If the aim is to predict the continuous value of a target, regression techniques have to be applied (e.g., Ivannikov et al. (2009)). If the aim is to predict the class membership of an object, classification will be used (e.g., Black and Hickey (2004)).

Inclusion of domain expert knowledge: The knowledge of domain experts is a valuable resource for the validity of predictive services (Žliobaitė et al., 2016). Several ways in which domain experts can support the development of valid predictive services have been identified. The simplest way to include domain experts into the process is the provision of true labels for the service. For instance, a predictive service is monitoring the quality in a chemical production facility. True labels for the chemical product can be acquired from experts who examine selected samples in a laboratory. Domain experts can also be included into the feature generation process for the machine learning model (Kubat et al., 1998). Especially experienced machine operators often know which hints and signals are relevant for the prediction of a machine failure and jointly it can be thought how to transform this information into a feature for the learning algorithm. It is also possible to explicitly apply knowledge of domain experts during the model building process, e.g., through the inclusion of fixed decision rules. Domain experts can also be relevant for the explicit detection of changes in the data. Human experts supported and empowered by advanced visual analytics tools can provide more insights than an algorithm alone (Keim et al., 2010).

Type of change: During the setup phase of a predictive service, expected changes of the data stream which affect the validity can already be identified. If this information is known a priori, suitable algorithms can be chosen beforehand. The different types of changes are based on the definition by Žliobaitė (2010). Sudden concept drift refers to situation where the data changes abruptly from one time point to another. Incremental and gradual concept drift both refer to situations where the change in the data stream happens slower over time. The two types are merged here since in real use cases the two are mainly not differentiable. Reoccurring concept drift refers to situations where data changes regularly to already known patterns such as seasonal contexts.

6.3.3 Algorithmic Decisions

The second part of the framework relates to the algorithmic and technical characteristics of the predictive service. This subsection is built on the research paper by

Gama et al. (2014) which identifies four categories for dealing with changing data over time:

Memory, Detection model, Learning, Loss estimation. The items for each category are also based on the work by Gama et al. (2014), however their item specification is very detailed. During our analysis, we realized that items can be merged without information loss. table 6.2 contains the corresponding categories as well as the items that we specified during our analysis.

Memory: Due to the massive amount of data produced in data streams, it is often infeasible to consider all data instances of a data stream. This category deals with the memory management of the predictive service. How many instances are stored for training or retraining of the algorithm? The quantity can range from a single instance to multiple or all instances. Often, algorithms only consider a window of the last n instances which are deemed to be still relevant to the algorithm. In cases with massive computing power or limited size of data in the stream, the algorithm might also consider all instances. It is also possible that only a sample of past data is used.

Tab. 6.2.: Algorithmic decisions for predictive service.

Memory	Single	Multiple (window)	All (gradual forgetting)	All (no forgetting)	Sampling
Detection model	Sequential analysis	Control chart	Two distributions	Contextual	Others
Learning mode	Retraining + single	Incremental + single	Retraining + ensemble	Incremental + ensemble	
Loss estimation	Model independent		Model dependent		

Change detection: Change detection refers to the mechanism that is applied to detect a change in the data stream. Various approaches have been proposed in research. In sequential analysis, the values of new data instances are compared to older values on the basis of statistical tests. Other approaches rely on statistical process control which is widely applied in chemical production processes. The algorithm tracks the number of correct predictions over time and if the amount of false predictions exceeds a predefined threshold, an alarm is triggered. However, this approach requires the instant provisioning of the true labels after the prediction. Another way is the application of two time-windows with different size. The statistical data distributions of the two windows are compared with statistical tests. In case

of a difference, a change or concept drift has happened. Contextual approaches use time-related measures for change detection.

Learning: As soon as new true labels for previous predictions are available to the predictive service, the machine learning algorithm behind it might be adapted. Usually, two different options are available: Retraining, where the old model is discarded and a new one is trained from the scratch or incremental updates, where the current model is slightly modified. Incremental learning is closely connected to the idea of continuous learning where the model never stops to learn according to the circumstances. Concerning the type of model, it can be differentiated between a single model or ensemble models where several models are combined for a prediction.

Loss estimation: Supervised machine learning models rely on feedback/true labels to optimize their performance. One can differentiate between model-dependent and model-independent loss estimation methods. Model-independent loss estimation approaches are more popular where a metric such as accuracy is computed and evaluated over time. However, some machine learning techniques such as Support Vector Machines allow the detection of changes in the data based on internal algorithmic characteristics.

While we now discussed the necessary characteristics of the setup of valid predictive services, the next section describes challenges during the operation of predictive services.

6.3.4 Operation of Predictive Service

During the operation of a predictive service, constant updates and improvements are necessary. Therefore, relevant topics are the acquisition of true labels as well as the evaluation criteria as depicted in table 6.3.

Label: The availability of true labels during operation is the most relevant feedback for the optimization of a machine learning algorithm deployed on a data stream. Therefore, this category is highly important to guarantee the validity and proper functionality of predictive services. Label availability is differentiated into three items: Full label, limited label and no label availability.

Full label availability refers to the case where the predictive service can receive access to all true labels after the prediction. Thus, the service receives feedback to every single prediction that it issued before, and the algorithm constantly receives

new training data for improvement. Weather predictions are an example for this item. If the service issues a weather prediction for the next day, we can always receive the true label for the weather on the following day—and continue to learn on these insights.

Tab. 6.3.: Operation of predictive service.

Label availability	Full	Limited	None
Evaluation criteria	Statistical evaluation metrics	Statistical evaluation combined with business impact	

Full label availability refers to the case where the predictive service can receive access to all true labels after the prediction. Thus, the service receives feedback to every single prediction that it issued before, and the algorithm constantly receives new training data for improvement. Weather predictions are an example for this item. If the service issues a weather prediction for the next day, we can always receive the true label for the weather on the following day—and continue to learn on these insights.

Limited label availability means that only a fraction of all true labels can be accessed after the prediction. In this case, the algorithm only receives feedback on its performance for a few instances. A further differentiation can be made by determining whether it is possible to select the instances for which labels are acquired (e.g., true quality of a specific chemical product can be determined by a laboratory analysis) or whether it is a random sample. An example for this is a predictive service determining customer satisfaction and true labels are received by sending a survey to all customers. However, we do not know who is going to respond to the inquiry. Therefore, the instances in the sample cannot be influenced and are random.

No label availability describes a situation when it is impossible to acquire labels. During training of the prediction model, a full data set with labels is available. However, during operation, when the predictive service is deployed, no true labels for previous predictions can be received. Therefore, the machine learning model cannot adapt its predictions to changes in the data. This demands methods that are specifically robust to outliers and unexpected deviations in the data (Russell et al., 2015). Reasons for no label availability can be that it is too costly to acquire the true labels. In other situations, it might just be impossible to receive the true labels, e.g., a machine part for whose functionality we can receive true labels with sensors in

a specialized test bench; however, in the field of application these sensors are not available and therefore labels are impossible to derive.

Evaluation criteria: The traditional evaluation of the performance of machine learning models is based on statistical evaluation metrics such as accuracy, recall or F1-score (Han et al., 2012). These metrics are suitable for expressing the mere algorithmic performance on the use case that is reflected. However, since this work considers the explicit service based on the algorithm, it is also necessary to study the business impact of the predictive service, especially the influence of validity over time (Gama et al., 2014). One way is to examine the influence on profits. Many use cases where predictive services are applied also lead to imbalanced cost of prediction mistakes. In case of predictive maintenance, it is costlier to not predict and therefore miss the failure of a machine resulting in a very expensive stop of the whole production instead of triggering a false alarm. It is also necessary to consider the environment where the predictive service is deployed. This refers to computational but also memory constraints in the IT infrastructure. Investment and setup costs also need to be considered. This category is closely linked to the business focus category in section 6.3.2.

6.3.5 Heatmap of Research Papers

In the following paragraph, we classify the 34 research papers which we used for the development of the framework. The result of this approach is a heatmap which is depicted in table 6.4.

Many application cases utilize several of the design options in parallel or test different variations in their approaches. Therefore, the sum of papers per row often exceeds 34. The heatmap indicates the different design options which were chosen by the different researchers. This allows to understand which of the available solutions and methods are really implemented for use cases and how often they are used. As stated above (section 6.3.1), the heatmap has to be understood as an exploratory tool since we do not map all existing research papers.

The heatmap indicates that current use cases dealing with changing data over time mainly use structured data for a classification problem with sudden or incremental changes in the data (e.g., Black and Hickey (2004), Delany et al. (2006), and Zhou et al. (2008)). There seems to be a lack in the consideration of economic challenges. Many projects do not name a specific business focus behind the implemented prediction model (e.g., Ivannikov et al. (2009)). The reason for this may lie in the

academic nature of the projects. Furthermore, most use cases rely on statistical evaluation only (e.g., Bose et al. (2014) and Kadlec and Gabrys (2011)). However, this consideration lacks evidence whether its economically viable and useful to implement such a service.

Tab. 6.4.: Heatmap of existing research classified into framework.

Business focus	External 4	Internal 4	Unknown 26		
Data input	Structured 25	Unstructured 9			
Machine learning task	Regression 5	Classification 29			
Domain expert knowledge	Label provision 17	Feature generation 18	Model building 2	Change detection 1	None 5
Type of change	Sudden / Abrupt 24	Incremental / Gradual 31	Reoccurring 2	Unknown 0	
Memory	Single 1	Multiple (window) 25	All (gradual forgetting) 9	All (no forgetting) 4	Sampling 3
Detection model	Sequential analysis 1	Control chart 12	Two distributions 3	Contextual 5	Others 14
Learning mode	Retraining + single 15	Incremental + single 11	Retraining + ensemble 2	Incremental + ensemble 9	
Loss estimation	Model independent 33	Model dependent 1			
Label availability	Full 21	Limited 13	None 0		
Evaluation criteria	Statistical evaluation metrics 31	Statistical evaluation combined with business impact 3			

0 | >0 & <5 | ≥5 & <10 | ≥10 & <20 | ≥20

Additionally, so far, the knowledge of domain experts is mainly used for label provision and feature generation (e.g., Ivannikov et al. (2009) and Klinkenberg (2005)). Efficient methods for expert integration into model building and change detection are missing. Most research projects also assume a full availability of true labels for the predictive service (e.g., Fdez-Riverola et al. (2007), Harries and Horn (1995), and Pawling et al. (2007)). Only few approaches have been developed for a limited label availability (e.g., Kurlej and Wozniak (2011)) and there is no approach in our paper selection which deals with no label availability. However, those two are the categories that prevail in real-world applications.

6.4 A Research Agenda for Preserving Validity of Predictive Services Over Time

The heatmap in the previous section indicates that there is still a lack of dedicated solutions for challenges during the design and operation of predictive services which remain valid over time. Based on our analysis, we identify two areas where current research approaches lack solutions so far.

> **Research Question C.2**
> Which are suitable methods for ensuring the validity of predictive services with limited availability of true labels in operation?

True labels for a prediction are a very relevant feedback mechanism for any kind of machine learning algorithm. However, for a predictive service in operation, this information is only partly available—if at all (Krawczyk et al., 2017). The proposed framework already depicts the different possibilities for the available number of labels. Additionally, Žliobaitė et al. (2016) define temporal dimensions when the true label is available to the predictive service. They differentiate this temporal dimension into real-time, time-lag and on demand. Real-time availability means that the labels are available in the next time period after the prediction. In other situations, true labels might arrive after a fixed or variable time lag. Asking a user for feedback is an example for a use case where the true labels can be acquired on demand. If we combine the temporal dimensions with the volume dimensions, several different scenarios emerge which are depicted in Table 6.5.

There exist various algorithms for predictive services with full label availability during operation. However, solutions for the other scenarios when only limited or

Tab. 6.5.: Different scenarios for label availability.

Time / Volume	Next time period	Time-lag	On demand
Full	e.g., Klinkenberg (2005)	e.g., Black et al. (2002)	e.g., Fdez-Riverola et al. (2007)
Limited	?	?	?
None	? (no time differentiation)		

no true labels are available to the predictive service are sparse so far. This is depicted by the question marks in Table 6.5. RQ C.2 aims at developing and establishing methods for each of the scenarios with a question mark. In case only a limited number of labels is available, it might be possible to derive the missing labels with the help of the existing ones (e.g., in form of a semi-supervised approach (Zhu et al., 2003)). Another approach might be an efficient method for the integration of expert knowledge which leads to the next research question.

Research Question C.3
How can expert knowledge be leveraged to increase the long-term validity of predictive services?

The knowledge of domain experts is a very valuable resource in any form of analytical solution. This research question deals with the challenge on how this expertise can be leveraged to increase the validity of predictive services. Therefore, this question aims at examining and evaluating methods for expert knowledge integration. Several areas for expert integration are already presented in the framework in section 6.3. With regard to label provision, it is interesting to examine which labelled instances are most useful for the predictive service in order to improve its importance. One possible solution could be the application of active learning (Huang & Dong, 2007), a machine learning technique. In case of changing data, the machine learning model asks for expert support in labeling the most important instances for ensuring its ongoing validity. This also relates to the limited label availability in RQ C.2.

Furthermore, a structured method to integrate experts into the model building process is necessary. Possible methods can be derived from approaches in other machine learning areas but also from research streams that already enabled the successful

integration of expert knowledge, e.g., in decision support systems (Kuusisto et al., 2015).

Basing a change detection algorithm on expert input requires a constant monitoring of the predictive service. However, for instance in most production plants, this is the case anyway. This setup allows to use the strengths of each player involved in this scenario. The algorithm can provide a constant monitoring and is not distracted by other activities. The human expert meanwhile can work on other tasks and is only alerted when unusual patterns are detected in the data. Supported by advanced visual analytics, the expert can then for instance identify the type of change that occurred in the data and act accordingly. Another approach is the inclusion of experts directly in the training phase of the prediction model. Domain experts can anticipate possible data drifts and a model can be tuned in order to detect these corresponding drifts.

Independently of the actual method that is applied, the development of an efficient integration method could also increase the acceptance and understanding of domain experts for automated decisions made by predictive services which is a common challenge in practice (Gama et al., 2014). During the answering of the RQs, a strong focus should lie on the economic evaluation of the proposed solution. Resulting costs (e.g., setup costs, computational costs during operation) need to be rigorously compared to the economic consequences of fewer false predictions for the predictive service.

6.5 Conclusion

Companies are increasingly dependent on data for the offering of their services. Predictive services, which are services based on predictions by supervised machine learning, are playing an important role in this context. These services constantly issue predictions over time which are an important decision support or might even act autonomously. Therefore, it is of high importance that predictive services work reliably. However, data streams constantly evolve and change over time and thereby challenge the proper functionality of the predictive service. This work proposes research areas to ensure the validity of predictive services over time. The contribution of this paper is threefold.

First, we provide a definition of predictive services and explain how their validity over time can be influenced by changing data. Second, based on previous research projects that are handling changing data streams, we develop a framework which

gives guidance to practitioners but also to researchers for setting up a new predictive service. Furthermore, it allows to differentiate between existing predictive services. Third, after classifying the existing research approaches into the framework, we identified two areas for improvement: The label availability in operation as well as the integration of domain experts. Correspondingly, we developed a research agenda which aims at developing solutions for those challenges. The derived research agenda is of high importance to any endeavor dealing with predictive services. It is important that such services are resilient against changes in the incoming data streams.

Besides these contributions, this work has limitations. Validity is only one aspect of predictive services which needs to be examined. However, a holistic view on predictive services requires that also other aspects such as organizational challenges are considered. Companies need to ensure that they have the required resources such as a skilled workforce and IT infrastructure available. Furthermore, legal requirements are gaining more and more importance. The introduction of GDPR in Europe poses many challenges for most companies (Zerlang, 2017). Predictive services often rely on personal data (e.g., the operators of a machine) or are based on IP-relevant data sources.

With regard to the developed framework, we are aware that the number of papers that we analyzed is limited, and we do not claim to have included all relevant research papers. As new papers are added to the framework, it still might change and adopt. Since this is work is a research agenda, its content is rather conceptual and further quantitative evaluation of the problems stated is needed. By conducting expert interviews with practitioners, we plan to further refine the research demand and the possible solution space.

The use of predictive services in productive environments is only at the beginning of its development. In the future, more and more services will rely on automated decisions based on machine learning algorithms. Therefore, it is very worthwhile to investigate methods to guarantee the long-term validity of those services.

Part IV

Concept Drift Handling for Regression Problems

7 Handling by Switching Models - the Error Intersection Approach[1]

7.1 Introduction and Related Work

Due to the large increase of data in the last decade, various industries are examining how to reap the benefits of this new resource. Machine learning is playing an important role in this context by transforming and (semi-)automating established business processes, spanning from marketing to operations (Chen et al., 2012). Typically, companies rely on machine learning models for increasing the efficiency of their processes or for offering new or improved services and products (Schüritz & Satzger, 2016). Typical applications of machine learning range from computer vision over speech recognition to natural language processing but also the control of manufacturing robots. Thereby, these techniques are especially influencing data-intensive tasks such as consumer services or the analysis and handling of faults in complex production systems (Jordan & Mitchell, 2015). Nowadays, most of these problems are tackled with supervised machine learning algorithms (Jordan & Mitchell, 2015) where the algorithm depends on labeled training data.

Machine learning can create ongoing value when the resulting models are deployed in the information systems of the respective company and deliver ongoing recommendations and optimized decisions on continuous data streams (Baier, Kühl, et al., 2019). However, data streams usually evolve over time and thus, their underlying probability distribution or their data structure changes (Aggarwal et al., 2003). The challenge of changing data streams for supervised machine learning tasks has been described with the term “concept drift” (Widmer & Kubat, 1996). The joint

[1]This chapter comprises an article that was published as: Baier, L., Hofmann, M., Kühl, N., Mohr, M., & Satzger, G. (2020). Handling Concept Drifts in Regression Problems – the Error Intersection Approach. *Proceedings of 15th International Conference on Wirtschaftsinformatik*, Potsdam, Germany. https://doi.org/10.30844/wi_2020_c1-baier. Note: The abstract has been removed. Tables and figures were reformatted, and newly referenced to fit the structure of the thesis. Chapter, section and research question numbering and respective cross-references were modified. Formatting and reference style was adapted and references were integrated into the overall references section of this thesis.

probability distribution of a set of input variables X and the label y is described as concept p(X,y). However, "in the real world concepts are not stable but change with time" (Tsymbal, 2004, p. 1). This fact indicates that machine learning models built on previous data might not be suitable for making predictions on new data. Therefore, it is necessary to frequently adapt the prediction approach. A mathematical definition of concept drift can be expressed as follows (Gama et al., 2014):

$$\exists X : p_{t0}(X, y) \neq p_{t1}(X, y)$$

This definition explains concept drift as the change in the joint probability distribution between two time points t0 and t1. Changes in the incoming data stream can depend on a multitude of different internal or external influences. Usually, it is impossible to measure all of those possible confounding factors in an environment—which is why this information cannot be included in the predictive features of a machine learning model. Those factors are considered as "hidden context" of the machine learning model (Tsymbal, 2004). Concept drifts in data streams are usually classified into the following types (Žliobaitė, 2010): Sudden or abrupt concept drift refers to situations where the data changes very quickly. A typical example for this drift type is the sudden failure of a sensor. Incremental and gradual concept drift is characterized by slower and more gradual changes, for instance preference shifts in a whole population. Reoccurring drift is determined by seasonal patterns such as ice cream sales in summer. There exists also a more detailed taxonomy for characterizing drifts which also contains categories such as drift duration and magnitude (Webb et al., 2016).

Figure 7.1 gives an overview on strategies which can be applied for detecting and handling concept drift. The first dimension refers to the application of an explicit drift detection algorithm. The second dimension describes the adaptations of the underlying machine learning model. The simplest option is the development of a robust, static machine learning model which is trained once and then deployed for an ongoing prediction (Guajardo et al., 2010), the upper left case in Figure 7.1. Other approaches continuously adapt the prediction model, e.g., with a sliding window where new data instances are continuously used to adapt the prediction model (Kuncheva & Žliobaite, 2009). Such approaches rely on an ongoing adaptation of the prediction model. Depending on the complexity of the model, this requires a lot of computational power. Furthermore, time constraints might also not allow the retraining of the entire model before the next prediction is required, especially in environments with limited resources, e.g., on mobile devices (Oneto et al., 2015). The lower part of Figure 7.1 depicts approaches which rely on a dedicated drift

detection. Drift detection can be handled by an algorithm which detects drifts in the incoming data or the distribution of the prediction error. Based on detected drifts, the model can either be retrained (Ivannikov et al., 2009) or another model can be applied. Approaches with and without drift detection are also named as active and passive approaches (Ditzler et al., 2015). Various explicit drift detection (active approach) approaches have been proposed, among others the most popular ones such as Page-Hinkley (Page, 1954), ADWIN (Bifet & Gavaldà, 2007), EDDM (Baena-Garcia et al., 2006). Those drift detection approaches are often used as benchmarks for new drift detection methods. All of these methods have in common that they observe the misclassification error to detect drifts in the data.

		Model Adaptation	
		No	Yes
Drift Detection	No	Static model (*this work*)	Window-based approach
	Yes	Drift detection with model change (*this work*)	Drift detection with model adaptation

Fig. 7.1.: Model adaptation and drift detection options.

Interestingly, predominant approaches for concept drift adaptation focus on classification tasks (Cavalcante et al., 2016) and require the statistical properties of a target variable with binomial distribution. However, many machine learning challenges need to be modeled as regression tasks, e.g., 20 out of 89 studies applying machine learning and being published in ECIS and ICIS between 2010-2018 are regression problems. In this case, approaches for classification cannot be applied or at least require costly adaptation measures which might potentially harm their performance. Therefore, this work focuses on the application of concept drift strategies for regression tasks which leads to the general research question of this work and the overall research endeavor.

General Research Question D
How can we address concept drifts in regression problems?

Existing approaches for drift detection on regression problems focus on the computation of dedicated drift detection features on the input data in order to detect drifts (Cavalcante et al., 2016). In contrast to this, we want to develop an approach based on the prediction error of various models in regression problems. In statistics, a

similar problem is the detection of structural changes in time series data (Verbesselt et al., 2010; Zeileis et al., 2003), a powerful tool to understand and analyze complex interdependencies such as in econometric models (Fernald et al., 2017). Research in this domain is closely related to unit root testing for time series where the characteristics of a stochastic component (besides a deterministic component) are examined. However, researchers have shown that unit root tests can lead to misleading results when not considering structural breaks in the time series (Perron, 1989). An application of those methods requires the full input data, i.e. the complete time series, as well as a prior definition of the number of structural breaks to be expected (Perron, 1989). Therefore, those methods can only be applied in hindsight after the time series has been completed which makes them less suitable for the application in the scenario depicted in this work. An adaptation of a prediction model months or even years after the occurrence of a concept drift does not promise large increases in predictive performance.

Other techniques rely on ensemble methods which have been widely studied and applied for concept drift (Sun et al., 2017; Xiao et al., 2019). Those methods usually rely on an incremental update of each model's importance and parameters. The importance of one model for the overall prediction is decreased and its parameters are adapted if the prediction error of the last prediction is rather large (Soares & Araújo, 2015).

The novel approach introduced in this paper—labeled as Error Intersection Approach (EIA)—utilizes static prediction models which are alternated based on the development of the error curve. Static models have the advantage that they need to be implemented only once and can also be scrutinized and tested extensively before they are deployed in production for ongoing predictions. Usually, companies are reluctant to deploy models that adapt and change automatically such as the above described ensemble methods due to the fear of bugs and unexpected behavior (Dunning & Friedman, 2017). In general, such black-box approaches are regarded critically due to the limited explainability of the issued predictions. Furthermore, our static model approach compared to dynamic models does not need to be retrained frequently which saves a significant amount of computational power as well as time. This advantage is especially important when machine learning models are deployed on local computing units, such as wireless sensor networks (Alsheikh et al., 2014).

EIA is inspired by the paired learner method for concept drift in classification tasks (Bach & Maloof, 2008). This method uses differences in prediction accuracy between a stable—but more accurate machine learning model and a reactive, simple model to detect drift and to trigger a retraining of the stable model. However, we focus on

regression problems and we also do not want to replace existing models in case of drift:

> **Research Question D.1**
> How can we design drift detectors utilizing multiple static models for regression problems?

For answering this question, we are building EIA based on two prediction models, one simple forecast model and a complex neural network model. EIA analyzes and takes advantage of the different degrees of complexity between the two models.

As application domain, this work performs demand forecast for mobility solutions which has been investigated before in IS and related disciplines, e.g., by predicting demand for carsharing services (Kahlen et al., 2017). However, this work focuses on the prediction of taxi demand in different taxi zones in New York City (NYC). The dataset is publicly available and provides information about every taxi trip which has been performed since January 2009. Due to the long timespan of the dataset, different types of drift can be observed, which indicates its suitability for the task at hand. Related work already investigated the problem of predicting taxi demand based on this dataset with complex prediction models such as LSTM or convolutional neural networks (Xu et al., 2018; Zhang et al., 2017). However, those approaches focus on optimizing the prediction error on shorter time spans. In contrast, we use this dataset for evaluating the long-term prediction of taxi demand on a test set of 6.5 years under the investigation of concept drift—and do so with our proposed approach.

The remainder of this paper is structured as follows: Section 7.2 presents the application domain and illustrates some of the existing drifts in the taxi demand data. Section 7.3 describes the design of EIA and the corresponding benchmarks. Section 7.4 introduces the results and explains the evaluation of our proposed approach. Section 7.6 discusses our results, describes theoretical and managerial implications, acknowledges limitations and outlines necessary next steps.

7.2 Use Case

This section describes the underlying dataset with taxi rides in NYC as well as some exemplary concept drifts which largely influence the prediction models.

7.2.1 New York City Taxi Dataset

The NYC taxi trip dataset (TLC, 2019) is provided by the New York Taxi and Limousine Commission (TLC) and contains information about all taxi trips that are conducted in NYC. We work with the taxi data from January 2009 up to June 2018. TLC provides information about the taxi trips separately for yellow taxis, green taxis and For-Hire-Vehicles (FHVs) respectively. Yellow taxis mainly operate within Manhattan, whereas green taxis are only allowed to operate outside of Manhattan. Furthermore, FHVs include ride-hailing services such as Uber. In this work, we are focusing on the yellow taxis since only their data is available for the overall timespan from the beginning in 2009. By focusing on a long-term duration, we expect more frequent and more significant concept drifts (e.g., weather, rise of Uber) to be present. In total, this gives us access to around 1.4 billion rides with yellow cabs.

With regard to preprocessing the data, we first remove outliers where distance or duration of a taxi ride are equal to zero. All trips before 2016 contain the exact geolocation of the start as well as the end of the taxi trip. All subsequent taxi trips only refer to the more high-level taxi-zones of pickup and drop-off of the passengers. Therefore, we match the previous exact geolocation data with the taxi zones introduced in 2016. Subsequently, we aggregate all taxi trips to identify the hourly demand per taxi zone. In this work, we focus on the 20 largest taxi zones because those already account for 60% of the overall taxi demand. This leads to a demand history with 83,231 hourly taxi demands for each of the 20 taxi zones.

7.2.2 Exemplary Drifts

To lay the foundation for our work, we describe exemplary concept drifts which we have identified in the taxi demand dataset. One source of change is the market entry of new competitors in the passenger carriage business which has already been discussed in related literature (Cramer & Krueger, 2016). Uber already launched its service in NYC in 2011 with a small fleet of drivers. However, the tracking of FHVs by the TLC only started back in 2015. Therefore, we do not have any information with regard to the use of Uber, Lyft etc. before that date. Figure 7.2 shows the overall demand trajectories for both Yellow cabs and Uber over the entire time span. At first, demand for yellow cabs rises steadily between 2009 and 2012. After 2012, however, the overall trend clearly indicates a decreasing demand due to the rise of new competitors. The typical demand pattern during the course of a year stays fairly constant during the whole duration. Referring to the previously introduced concept

drift patterns, the decreasing demand for yellow taxis over time can be described as an incremental concept drift where data patterns slowly evolve. However, we need to be careful with assumptions about the exact time as well as impact of drifts in this real-world dataset since there is no ground truth describing the exact characteristics of this drift as opposed to simulated data (Tsymbal, 2004). Furthermore, changes in the real-world are often related to a multitude of factors.

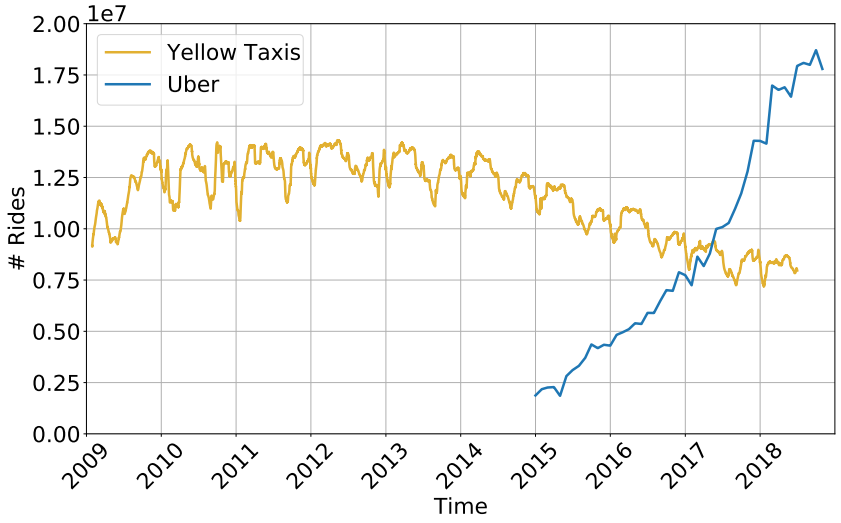


Fig. 7.2.: Overall NYC yellow cab and Uber demand per month.

Another source of drift in the dataset are extreme weather events such as hurricanes or thunderstorms. Taxi demand naturally adapts to those unusual weather situations. Figure 7.3 depicts the taxi demand during the course of Tuesday, January 27th, 2015 (in blue) and the average demand on Tuesdays in 2015 (in red) as well as the 25% and 75%-quantile of the average demand.

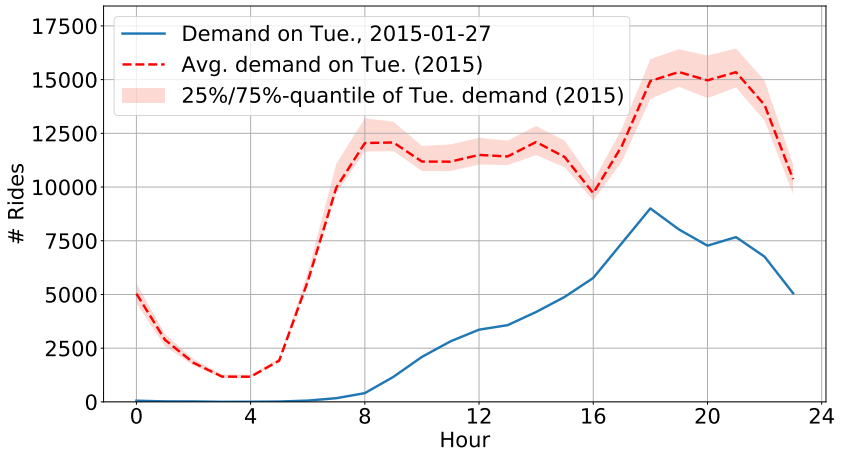


Fig. 7.3.: Taxi demand during a blizzard on 2015-01-27.

It is obvious that the average demand and the demand on January 27th clearly deviate. The blue line indicates nearly zero demand during nighttime and early morning which is due to a blizzard which passed by NYC with declared snow emergencies as well as enacted travel bans. In contrast to the incremental drift example above, this event can be regarded as a sudden concept drift. Another example for sudden drifts in the dataset is the occurrence of special events such as festivals in dedicated taxi zones. In this case, the demand for taxi rides is suddenly increased dramatically in comparison to usual demand patterns.

When applying a machine learning model to predict future taxi demand, we are aware that one could probably increase significantly the predictive power of a model by including external data such as weather data, competitor data or an events calendar. However, the focus of this work is not to provide the best possible demand prediction. We aim at examining and quantifying the effect of concept drifts in real-world situations. Therefore, we will consider weather and other facts as external hidden variables (see Section 7.1) that we cannot observe in the application environment. This requires an adequate preparation and adaptation of the applied prediction model.

In this particular use case, it seems rather easy to identify factors (e.g., weather etc.) which have a large influence on the prediction power of a model as well as how to include this information as predictive features. This might also be due to the nature of the overall project since nearly everyone has already used a taxi as a means of transportation. However, in hindsight, it is often easier to identify unusual demand patterns and subsequently investigate the underlying reason. For a predictive model, though, this information is required in real time. Furthermore, including weather features in this use case and disregarding other unidentified influencing factors might lead to overfitting of drift behavior on weather phenomena.

In other use cases and application areas, it is often very difficult or impossible at all to identify influencing variables apart from the obvious dataset (Stowers et al., 2016). In case those can be identified, it is often impossible to measure and quantify those factors. Therefore, we decide to restrain the inclusion of external data sources in this work. Furthermore, after a thorough analysis of our data, we also have identified a lot of fluctuations and abnormalities in the dataset where it is impossible to identify the underlying reason without additional knowledge. Usually, drift detection and adaption approaches are evaluated based on simulated datasets. In this work, we want to evaluate our drift detection approach (EIA) based on a real-world dataset.

7.3 Design of the Error Intersection Approach

With the introduction of various examples for drifts in the NYC taxi dataset at hand, it is necessary to develop a prediction strategy which accounts for those concept drifts and provides reasonable predictions. As described in Section 7.2, we assume incremental as well as sudden concept drift to be present in this dataset. Since those two types of concept drifts are fundamentally different and require adapted handling strategies each, we decide to focus on sudden concept drift in this work. We propose an approach which relies on two different prediction models.

Ensemble methods have been widely used in concept drift adaptation methods (Gama et al., 2014). Usually, those approaches rely on the combination of various models with an average of the delivered predictions to increase the overall performance. However, in this work, we propose a different approach which uses the predictions issued by two static models with different complexity to detect drift in the data and adapt the prediction accordingly. This approach is depicted in Figure 7.4. As first model, we use a simple model (M_{simple}) which is only influenced by the most recent demand in the respective taxi zone. As second model, we apply a neural network ($M_{complex}$) which receives as input a large demand history over all taxi zones. During normal times, $M_{complex}$ is applied because it successfully captures the general demand structure and therefore is able to compute accurate predictions for the taxi demand in the respective taxi zone. However, during times with sudden concept drifts, $M_{complex}$ cannot provide accurate predictions since the demand patterns clearly deviate from the usual trajectories. In those cases, M_{simple} is applied because it can quickly adapt to current demand changes. By design, this approach is presumably only able to deal with sudden concept drift since incremental drifts require frequent adaptations of the predictions models which is not the focus of this work. The switch between models is triggered by an intersection of the prediction error curves of $M_{complex}$ and M_{simple}. Therefore, we call this the "error intersection approach" (EIA). EIA is feasible since we always receive the true label for the prediction after the course of one hour.

EIA is inspired by a streaming architecture for deploying machine learning models (Dunning & Friedman, 2017) which suggests the deployment of several individual and independent prediction models. This way, it can be guaranteed that a prediction can always be issued in time for a new data instance. Furthermore, EIA is based on the paired learner approach (Bach & Maloof, 2008) which uses differences in prediction accuracy between a stable and a reactive machine learning model to detect drifts.

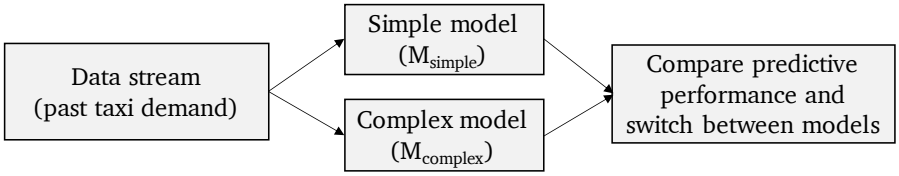


Fig. 7.4.: Design approach of EIA.

M_{simple} is a baseline model often used in time series forecasting literature (Hyndman & Athanasopoulos, 2018). It just predicts the demand value from the last period in the respective taxi zone. This model does not learn any parameters but is very good at capturing current trends.

$M_{complex}$ is a neural network which contains as input features the regions and the current weekday as one-hot encoded vector. Furthermore, it receives the demand of the last 24 hours as well as the demand during the same hour on the same weekday in the four past weeks. Additionally, we include cosine and sine features to depict that hours and months are cyclical features to improve prediction performance as suggested in literature (Hernández et al., 2013).

$$\cos_{hour} = \cos(\frac{h * 2 * \pi}{24}), \sin_{hour} = \sin(\frac{h * 2 * \pi}{24})$$

To compute the respective features regarding months, the denominator is adapted to 12. We use 128 neurons in the hidden layer with a “relu” activation function and the network is trained using 50% dropout. Similar network architectures have been used before for taxi demand prediction (Liao et al., 2018).

7.4 First Evaluation

This section introduces first results with the previously proposed design. Figure 7.5 illustrates the applied combinations of drift detectors and models for the prediction of the taxi demand. The upper part of the table (in red) describes the combinations that have been implemented so far. The lower part of the table (in blue) contains the options that need to be pursued in future work.

As error measure, we apply the Root Mean Squared Error (RMSE) which is the standard metric to evaluate taxi demand predictions on the NYC dataset (e.g., Zhang et al. (2017)). Furthermore, we compute the Symmetric Mean Absolute Percentage

	Ensemble	Drift Detector	Applied prediction model	Retraining after drift detection
This work	No	n/a	1 (M_{simple} *or* $M_{complex}$)	No
	Yes	n/a	2 (M_{simple} *and* $M_{complex}$)	No
	No	Page-Hinkley ADWIN EDDM **EIA**	2 (M_{simple} *and* $M_{complex}$)	No
Future work	No	**EIA**	N models (e.g. LSTM)	No
	No	**EIA**	N models (e.g. LSTM)	Yes

Fig. 7.5.: Overview of applied drift-detector and model combinations in this work and for future work.

Error (SMAPE) as a relative error measure. Demand from 2009 up to 2011 is considered as training data, whereas demand after 2012 is used as test data.

As baseline, the performance of $M_{complex}$ and M_{simple} alone on the dataset is evaluated. M_{simple} does not contain any parameters and therefore cannot be updated. However, with regard to $M_{complex}$, we retrain the weights once a year so that $M_{complex}$ can adapt to the general trend of the taxi demand over the years. This means that the forecast for 2012 is performed with a model trained on data from 2009-2011, the forecast for 2013 is issued by a model trained on data from 2010-2012. As additional baseline, we build an ensemble from both models' predictions since existing drift handling strategies for regression usually are built this way (see Section 7.1). We compute the Exponential Weighted Moving Average (EWMA) of the last 6 predictions errors of both models respectively and determine the sum of errors. Subsequently, we compute the contribution of each model to the sum of errors to determine the weights of each model for the ensemble prediction (e.g., if $M_{complex}$ accounts for $1/3$ of the sum of errors, its weight for the next prediction are $2/3$). Furthermore, we test the established methods Page-Hinkley (PH), ADWIN and EDDM as drift detectors. When those methods detect a drift, the switch between the two prediction models is performed. Since EDDM can only be applied to classification problems, we need to transform the regression problem (Xiao et al., 2019). EIA (see Section 7.3), in contrast, switches between models based on the EWMA of the prediction errors in the last 6 hours. The model with the lower recent prediction error (either M_{simple} or $M_{complex}$) is the active model for computing the next prediction. After issuing the prediction, the error terms are evaluated once more and the model for the subsequent hourly prediction is selected.

Table 7.1 introduces the results for the overall prediction performance of the different approaches on the test set. Not surprisingly, M_{simple} produces the highest RMSE by far, which portrays the worst result. This model is just too simple for producing a good overall forecast. In contrast, $M_{complex}$ already performs well with an RMSE of 50.478. Standard drift detection methods seem to work reasonably on this dataset; however, their application leads to a worse performance compared to $M_{complex}$. EIA is depicted in the last row and shows a better performance than $M_{complex}$ alone. The amount of model switches is depicted in the second column.

Tab. 7.1.: First results of EIA in comparison to benchmarks, based on RMSE and SMAPE (the lower the better).

Approach	# Model switches	RMSE	SMAPE
M_{simple}	n/a	115.871	13.80%
$M_{complex}$	n/a	50.478	6.01%
Ensemble (EWMA)	n/a	58.381	6.75%
ADWIN	70	97.657	11.64%
EDDM	30	112.783	13.47%
EIA	**365**	**50.370**	**5.98%**

The effectiveness of EIA is illustrated in Figure 7.6 which depicts the demand predictions during the blizzard in 2015 (see Figure 7.3). In the beginning, EIA (red dashed line) always chooses $M_{complex}$ because of the lower prediction error (black line). However, at around 4pm of January 26th (marked by a black vertical line), the approach switches to M_{simple} (lower error curve in grey), which can quickly adapt to the unusual demand pattern. $M_{complex}$ clearly fails to predict this behavior correctly (e.g., peak of the error curve at around 6pm).

7.5 Discussion

The absolute difference in RMSE between EIA and $M_{complex}$ seems to be rather small. However, in total, we average over more than 1.13 million predictions. Therefore, we perform a Diebold-Mariano test to compare the predictive performance between EIA and $M_{complex}$ (Diebold & Mariano, 2002). With a p-value of 1.89 * 10-7, we can conclude that there is a significant difference in forecast performance between the two approaches. The absolute small difference can be explained by the fact EIA only chooses M_{simple} in 1.24% of all necessary forecasts (706 out of 56,951 hourly forecasts in total). This means that in 98.76% of all forecasts, the predictions of EIA and $M_{complex}$ are the same. However, during those 706 hourly forecasts where M_{simple} is chosen by EIA, the prediction performance is largely improved by 8.4%

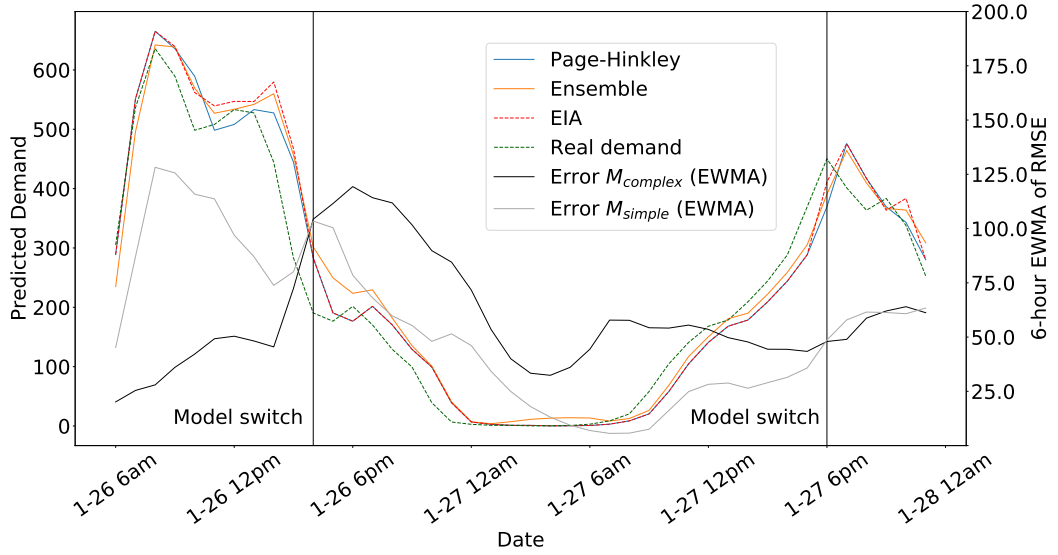


Fig. 7.6.: Predictions of EIA during blizzard on 2015-01-27.

(RMSE EIA: 75.31 vs RMSE $M_{complex}$: 82.21). This is a good indication for the effectiveness of the approach and the share of M_{simple} might be larger on a different dataset with corresponding impact on the difference in predictive performance.

The superiority of EIA compared to established drift detectors such as PH can probably be explained by information asymmetry: EIA uses the information of two prediction models and their error curve to select the current optimal model. PH, in contrast, works by analyzing the development of the prediction error of only one prediction model (in our case, either $M_{complex}$ or M_{simple}) and therefore has access to less information resulting in model switches at unfavorable points in time.

This argument does not hold true for the ensemble approach since both models are used to compute the resulting predictions. However, performance loss in this case might be explained by the overall bad performance of M_{simple}: The weight for each model is determined by considering the past prediction errors which will generally lead to a high weight for $M_{complex}$. Nevertheless, M_{simple} will almost always also receive a weight larger than zero, thereby negatively impacting the ensemble prediction.

Furthermore, we analyze for EIA on which days M_{simple} is applied the most. This way, we can identify in hindsight the days with most significant concept drifts. Table 7.2 shows an excerpt of days with a frequent use of M_{simple} for a prediction as well as the corresponding special events (drift cause) on that day. The second column depicts the absolute improvement in RMSE of EIA compared to predictions by $M_{complex}$ alone. In most cases, drift is triggered by unusual weather events or

public holidays. However, we also find several days (e.g., August 1st, 2013) where we are not able to identify the underlying reason for the drift cause. This depicts the strengths of EIA since it does not require any additional data compared to an explicit integration of features such as weather or public holidays (see explanation in Section 7.2). Furthermore, we compared taxi demand during New Year's Eve for several years: In 2012, for instance, taxi demand peaked before midnight whereas in 2017, the highest demand occurred after midnight. This variability complicates the learning process even if an explicit feature for holidays is included. In future work, we want to perform a more comprehensive analysis of these results in order to understand when it is most suitable to apply the approach.

Tab. 7.2.: First results of EIA in comparison to benchmarks, based on RMSE and SMAPE (the lower the better).

Date	**Abs. RMSE Improvement**	**# Predictions by M_{simple}**	**Probable Drift Cause**
2012-07-04	5.07	14/24	4th of July
2012-10-29	24.41	22/24	Hurricane Sandy
2012-12-25	9.22	17/24	Christmas Day
2013-08-01	9.35	10/24	? (unknown)
...	...	...	...
2017-06-25	5.48	10/24	? (unknown)
2018-03-21	15.21	14/24	Cyclone (Nor'easter)

7.6 Conclusion

In the work at hand, we explore a novel approach ("error intersection approach") for concept drift handling for supervised regression tasks. Established drift detection methods usually focus on classification problems. Our approach, in its core, depicts a strategy to switch between the application of simple and complex prediction models which is designed to deliver superior performance results in real-world data sets prone to concepts drifts. We hypothesize that the drift detector allows to play out the individual strengths of each model, switching to the simpler model if a drift occurs and switching back to the complex model for typical situations. To illustrate our suggestion, we instantiate the approach on a real-world data set of taxi demand in NYC. For this very data set, we are aware of multiple drifts, e.g., short-term drifts such as the weather phenomena of a blizzard. We apply different, typical predictive models for regression tasks and are able to show that our suggestion outperforms all regarded baselines significantly.

Obviously, these results are preliminary and have certain limitations. Our prediction is presumably worse than very complex CNN and LSTM architectures (Liao et al., 2018; Xu et al., 2018). Also, we have not tested other powerful machine learning techniques such as XGBoost (Chen & Guestrin, 2016). However, previous work has evaluated those models only on shorter test periods (2 and 6 months respectively). Furthermore, models from related work reveal no insights on their effectiveness for drift handling, while EIA offers more transparency (e.g., how often were the model switched, when was which model used, etc.) and, therefore, allows for more interpretability (Gilpin et al., 2018). Still, the applied approach is (presumably) only meaningful when sudden concept drift is expected. To further explore this, more research is required to formulate clear guidelines on the precise cases in which we can recommend the use of the suggested approach. Furthermore, our approach is only feasible when the true label for a delivered prediction can be acquired afterwards, which might not be the case in all applications. However, this limitation also holds true for established drift detection methods.

In future work, we aim to further develop EIA regarding several aspects. To examine generalizability, we aim to test the effectiveness of EIA on a different dataset. A simulation study might be a worthwhile tool in this context. Furthermore, we want to extend our work on the drift detection algorithm. Additionally, on the presented data set, it will be interesting to identify regions with the highest drifts where it is most appropriate to apply EIA. Furthermore, EIA in its current form is rather basic, as we only regard one change detection and only switch between two models—a simple and a complex one. In future work, we aim to employ more sophisticated change algorithms, but also investigate approaches with more models, e.g., very simple/average/very complex. Finally, we did not regard models with immediate retraining after the drift detection—which also remains an interesting option for future work.

8 Handling by Switching Adaptation Mode - the Switching Scheme[1]

8.1 Introduction

Artificial intelligence in general and machine learning in specific are omnipresent when it comes to automation capabilities in information systems (Schüritz et al., 2017). While there is an ever-growing body of knowledge on machine learning methods, their application as well as their impact on socio-technical systems, only a minority of research considers the effects when machine learning models are incorporated into (existing) systems. Therefore, it is important to put more focus on this "deployment" step (Shmueli & Koppius, 2011) and the choices associated with it—as a successful deployed machine learning artifact is important to ensure constant performance as well as the trust in the artifact by its users. Especially, trust is of major importance when it comes to the acceptance of new technologies (Söllner et al., 2016). Therefore, it must be in the best interest of researchers and practitioners to ensure that machine learning models are not only explored in theory and isolated proof-of-concepts—but also in their deployed environment to assure long-term trust in the implemented solutions (Wang & Benbasat, 2005).

The aspects of machine learning artifact deployment are manifold (Baier, Jöhren, et al., 2019): ranging from data access (Lennerholt et al., 2019), scalability (Baier, Jöhren, et al., 2019) and security (Barreno et al., 2010) up to interface design (Buitinck et al., 2013). However, one aspect needs to be incorporated into the very early design of the models: The phenomenon of changing data over time, usually referred to as *concept drift* (Tsymbal, 2004). While articles in the field of computer

[1]This chapter comprises an article that was published as: Baier, L., Kellner, V., Kühl, N., & Satzger, G. (2021). Switching Scheme: A Novel Approach for Handling Incremental Concept Drift in Real-World Data Sets. *Proceedings of the 54th Hawaii International Conference on System Sciences*, Maui, Hawaii, USA. https://doi.org/10.24251/HICSS.2021.120. Note: The abstract has been removed. Tables and figures were reformatted, and newly referenced to fit the structure of the thesis. Chapter, section and research question numbering and respective cross-references were modified. Formatting and reference style was adapted and references were integrated into the overall references section of this thesis.

science already engineered different algorithms (e.g., ADWIN) and applied them to synthetic data sets (e.g., STAGGER), most of the work remains on a theoretical level. In our work, we stress the importance of incorporating concept drift strategies into machine learning models and apply them on real-world data sets. We propose a novel strategy called *switching scheme*, which we believe to be a meaningful addition to the tool set of data scientists and IS researchers working with real-world data sets, aiming to ensure long-term validity of their deployed artifacts. The switching scheme—at its core—combines the two principles of retraining and incremental updates of a machine learning model.

We explore existing approaches as well as our own in the application field of demand forecasting, as it poses a popular application candidate within IS (Esswein et al., 2019). In our work, we apply the proposed switching algorithm in-depth to taxi demand data in New York City (TLC, 2019) and, furthermore, implement it additionally on a flight record data set (Ikonomovska et al., 2011) as a robustness check. In terms of concept drift, we focus on incremental drifts in this work, as they are very typical for systems deployed with a long-time horizon, e.g., sensors wearing off over time (Kadlec & Gabrys, 2011). Therefore, we aim to answer the following research question:

Research Question E
How can a forecasting system be designed to handle incremental drift on real-world data?

By answering this question, we contribute as follows: First, we introduce a switching algorithm which combines the advantages of retraining and incremental updates. Second, we benchmark various drift detectors regarding performance on a real-world demand forecasting data set with incremental drift. Third, we can clearly show that drift handling strategies improve prediction accuracy, whereas static models wear out over time and their performance decreases. Fourth, there are differences between drift handling strategies and the differences are significant—however, using *any* drift detection strategy seems to be superior than to apply none at all. As a result, we encourage researchers and practitioners to incorporate concept drift strategies within their deployed machine learning artifacts.

The upcoming Section 8.2 presents related work on which we base our research. Section 8.3 introduces the use case while Section 8.4 gives an overview of the applied drift handling strategies. Section 8.5 describes the evaluation of those strategies and Section 8.6 summarizes our results, acknowledges limitations and outlines future research.

8.2 Related Work

To lay the necessary foundations for the remainder of this work, we briefly introduce research regarding concept drift and demand prediction.

8.2.1 Concept Drift

Concept drift describes the phenomenon of changing data over time in machine learning for data streams (Widmer & Kubat, 1996). A concept $p(X, y)$ is described as the joint probability distribution over a set of input variables X and the target variable y. However, concepts are often not stable in the real world but change over time (Tsymbal, 2004). Concept drifts are usually classified into the following categories (Žliobaitė, 2010): Sudden concept drift where the data changes very quickly (e.g., sudden machine failures), incremental and gradual concept drift (e.g., macroeconomic changes) and reoccurring drift such as seasonal patterns (e.g., AC sales in summer). Successful concept drift handling usually requires various decisions, including the selection of the right training data, the choice of a suitable drift detection method and also how to adapt machine learning model in case of drift (Gama et al., 2014).

Traditional methods for concept drift detection comprise algorithms such as STEPD, ADWIN or HDDDM. The Statistical Test of Equal Proportions (STEPD) is based on the idea of monitoring the recent accuracy of a machine learning model compared to the overall accuracy (Nishida & Yamauchi, 2007). The Adaptive Windowing (ADWIN) approach uses sliding windows with adaptive size to correspond to different rates of change within the window (Bifet & Gavaldà, 2007). Drift is detected by partitioning the window observations into subwindows and comparing the error rate of the classifier among those subwindows. While STEPD and ADWIN require the classification error for drift detection, the Hellinger Distance Drift Detection Method (HDDDM) detects drifts by monitoring the input features (Ditzler & Polikar, 2011). HDDDM detects drift by measuring the Hellinger distance between the distribution of the input features of recent observations and a reference distribution.

While the previous algorithms all originate from the computer science community, many statistical methods for handling changing data patterns exist as well. Unit root testing allows the program to determine whether a time series is stationary, trend stationary or has a unit root (Haldrup et al., 2013). This is a powerful tool to understand and analyze complex interdependencies such as external effects on stock markets, e.g., on the Bitcoin price (Kremser et al., 2019). However, it has

been shown that unit root tests without considering structural breaks can cause false inference for time series predictions (Zeileis et al., 2003). Dealing with structural breaks require the complete time series as well as a prior definition of the number of structural breaks to be expected (Glynn et al., 2007). Therefore, those methods can only be applied in hindsight after the time series has been completed which makes them not applicable to real-world scenarios. An adaptation of a prediction model months or even years after the occurrence of a concept drift does not promise large increases in predictive performance. Another statistical approach for detecting monotonic trends in time series is the non-parametric Mann-Kendall (MK) test which is often applied in the context of meteorological studies (Sonali & Kumar, 2013). The MK test checks whether observations in a time series are following a monotone trend.

8.2.2 Demand Forecast

Demand forecasts are a fundamental concept for optimizing many business processes and numerous IS studies analyze this problem. Examples range from technical applications like the prediction of liquidity demand (Esswein et al., 2019) up to socio-economic ones like the demand of human resources to improve process operations (Stein et al., 2018). Other approaches investigate demand forecasts for emergency medical services (Steins et al., 2019) or to predict demand for automotive spare parts (Steuer et al., 2018). In the mobility sector, the demand for carsharing services has been analyzed (Kahlen et al., 2017).

In general, both statistical and machine learning methods are widely used for traffic and transportation applications. Especially, parametric forecasting models such as the ARIMA model have been commonly used for time series modeling in past studies (De Gooijer & Hyndman, 2006). For instance, ARIMA and Possion models have been combined to predict short-term taxi demand (Moreira-Matias et al., 2013). However, recently, the importance of complex machine learning models such as XGBoost (Liao et al., 2018) and deep learning models has increased significantly. Deep learning applications in this domain consist of traditional multilayer perceptrons, convolutional or LSTM networks and autoencoders as well as combinations of those (Laptev et al., 2017; Zhu & Laptev, 2017).

8.2.3 Research Gap and Contribution

In terms of closely-related research, we previously highlighted works from the streams of concept drift and demand forecasting. We can identify a lack of research on the application of concept drift on real-world data (Mittal & Kashyap, 2018) and regression problems (Baier, Kühl, et al., 2019). Therefore, we choose to investigate the real-world New York City taxi data (see Section 8.3). In regard to that data set, related projects so far try to optimize the forecast using complex prediction models such as Long Short-Term Memory (LSTM) networks (Xu et al., 2018). However, those approaches consider the demand forecasting task as a static problem and focus on building one machine learning model only which achieves high prediction accuracy on short time spans such as months—therefore, neglecting strategic perspectives of the business involved, e.g., resource planing over multiple years. Our work, in contrast, aims at investigating the prediction performance over the course of several years. We systematically test and evaluate the effects of different adaptation strategies on machine learning models over time. Therefore, first, we provide an alternative way of analyzing the demand forecasting problem in NYC, which has not been performed yet. Second, we add knowledge by providing a comprehensive analysis and benchmark of different concept drift detection algorithms on real-world data. Related work mostly evaluates on synthetic data sets (Žliobaitė et al., 2016), constructed for the purpose of containing clear concept drifts, which are not typical in real-world data (Gama et al., 2014). Third, we propose a novel strategy combining updates and retraining of models to address characteristics of real-world data. In contrast to most existing related work, this strategy is not any novel algorithm for drift detection. Instead, we introduce a novel way for the adaptation of the corresponding machine learning model after a drift has already been detected (see Gama et al. (2014)). In fact, concept drifts in the real world are often overlapped by a multitude of influencing factors. Therefore, the impact of concept drifts will usually be delayed which is addressed by the proposed switching between updates and retraining.

8.3 Use Case

The New York City Taxi and Limousine Commission (TLC) regulates the operations of regular yellow taxis and for-hire vehicles such as Uber and Lyft in NYC. Currently, around 1 million trips are recorded every day (TLC, 2019). TLC makes this data available to the public since 2009. Each trip record contains, among other features,

information about the pick-up and drop-off time and location, the trip distance, payment types, fares and number of passengers. While information about the exact pick-up and drop-off location was provided from 2009 to June 2016, the subsequent records only contain a taxi zone ID. NYC is divided into 263 different taxi zones in total covering the boroughs of Manhattan, Bronx, Queens, Brooklyn and Staten Island.

This work focuses on the yellow taxi trip records due to its range over several years including over 1.4 billion records. The large number of records in the data set and the fact that the taxis operate in different taxi zones increases the chance to observe different incremental drifts over time, since trip records reveal certain pattern and habits of the customers (Liao et al., 2018).

Consequently, we use all taxi trip records from 2009 up to June 2018 for this analysis. We remove all trips from the data set with locations outside of New York as well as anomalies regarding trip information (e.g., negative metered distance). Similar to previous work, we approximate the real demand by considering the actual number of pick-ups in each taxi zone (Zhou et al., 2018). We transform this data by aggregating the demand per taxi zone on an hourly basis. The transformed data set therefore includes the taxi demand for a given hour starting in 2009 up to June 2018 for all 263 zones. Comparing the total demand among the taxi zones, the 20 busiest taxi zones already account for almost 60% of the overall demand and are mainly located in Manhattan. Thus, we only consider the 20 busiest taxi zones for further analysis and modeling.

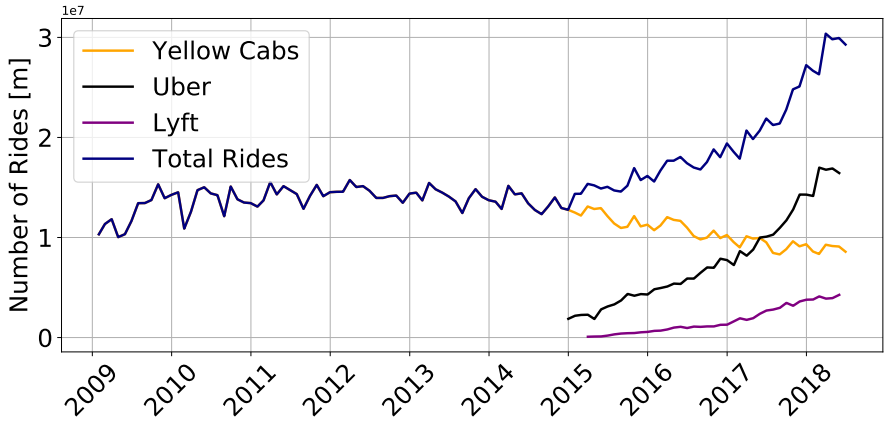


Fig. 8.1.: Taxi demand in NYC per month.

Analyzing the data set for possible drifts, a decreasing trend in the overall taxi demand can be identified by considering the overall demand depicted in Figure 8.1. While the yellow cab demand exhibits a yearly pattern and increases from 2009 to

2011, a downward trend is observable starting from 2014. This is remarkable since the total demand of rides increases. This might be explained with the increased competition among yellow cabs and ride-hailing services (Cramer & Krueger, 2016) but also new forms of transportation such as shared bikes. This form of a slowly changing demand pattern can be regarded as incremental drift.

8.4 Methodology for Handling Incremental Drift

With the foundations of the use case at hand, we now introduce the different drift handling strategies addressing incremental concept drift. Furthermore, we explain how we set up the drift detectors for the taxi demand data set.

8.4.1 Adaptation Strategies

Overall, the applied drift handling strategies can be differentiated on two dimensions: Adaptation and learning mode. The adaptation dimension explains how a model change is initiated, either based on a trigger such as a change detector or based on a fixed periodic interval such as three months without any explicit detection of change. The learning mode refers to how the model is changed when an adaptation is required. The model can either be retrained from scratch or updated with the most recent observations. The intuition behind the periodic adaptation is to frequently train a new model on the most recent data. This way, a new model can capture new concepts iteratively. Table 8.1 summarizes all performed strategies. All adaptation strategies except for the *switching scheme* are inspired by the taxonomy of adaptive learning systems (Gama et al., 2014).

Tab. 8.1.: Overview on drift handling strategies.

		Learning mode		
		Retraining	Update	Switch
Adaptation	Periodically	•Quarterly Retraining •Yearly Retraining	•Quarterly Update •Yearly Update	
	Triggered	•Retraining	•Update	•Switching Scheme

Regarding the *periodic adaptation strategy*, we test yearly and quarterly adaptation as different strategies. The training data for each model is fixed to two years of observations (sliding window of two years). In case of the yearly retraining strategy,

each model is deployed to make one-step-ahead forecasts for the upcoming year. After all predictions have been computed, a new model is trained on the most recent observations. This means that the initial model is trained on taxi demand data of 2009 and 2010 to compute forecasts for 2011, while the next model is trained on data of 2010 and 2011 to compute forecasts for 2012 and similarly for the following years. The quarterly retraining strategy follows the same procedure. However, a new model is trained on a quarterly basis. In contrast, the incremental update strategy regularly updates the existing model by performing incremental learning on the most recent observations. The incremental update strategies follow the same logic as the retraining strategies: A yearly incremental update strategy computes predictions for the upcoming year. When all predictions are obtained, the model is updated with the most recent observations.

The *triggered adaptation strategy* initiates a model change based on explicit drift detection. Incoming data is monitored on a continuous basis and statistical tests are performed to detect drift. If a change is suspected, an adaptive action is triggered. We again test both retraining as well as updating the prediction model based on this trigger. In accordance with the periodic adaptation strategies, the window of the training data for each model change is set to two years. For instance, a drift detected on 10th of July in 2012 initiates a training of a new model with a training data set containing the observations from 10th of July 2010 up to 10th of July in 2012.

We also propose the novel *switching scheme adaptation strategy* for handling concept drift in real-world data sets. This strategy performs a combination of incremental updates and retraining of prediction models. The idea behind the switching scheme is to take advantage of the individual benefits of a complete retraining and an incremental update strategy. The initial model is kept and is incrementally updated with the most recent observations for a certain period of time. This allows the model to adapt to the most recent concepts. At the same time, the model profits from access to an overall large training set since both the initial training data as well the most recent observations are considered. However, after a certain period of time, updates will not be sufficient to adapt the model to the latest data changes (since the concept is now fundamentally different from the previous) which means that the current model is outdated. Therefore, this requires the retraining of a new model.

Figure 8.2 illustrates the concept: First, the initial training of the model is performed. Afterwards the model computes the next predictions. If a drift is detected at $t < \tau$, the model is incrementally updated based on the observations in λ. The lock in the figure symbolizes that no retraining is allowed during this period. If a drift is detected at $t > \tau$, a new model is trained to replace the existing model and τ is reset.

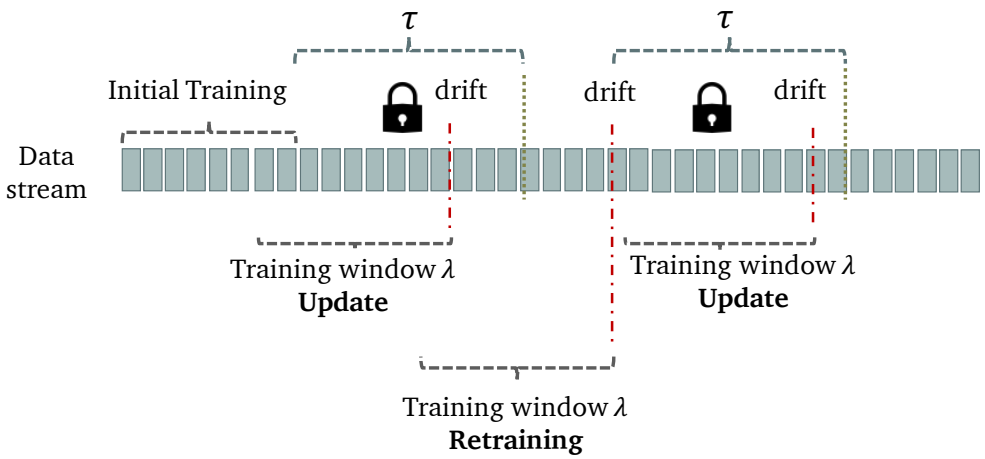


Fig. 8.2.: Explanation of switching scheme.

This procedure is repeated until all forecasts are obtained. For our experiments, we set the training window $\lambda = 2$ for all drift handling strategies. For the switching scheme, we set $\tau = 1$. As a result, the model is incrementally updated if a drift is detected as long as the last retraining does not date back more than one year. In general, τ is a parameter which is specific to the application domain and therefore the selection of the optimal parameter value requires domain knowledge. We suggest to use a value which also reflects a logically connected unit of time (e.g., one year in our case or one month for projects with a shorter time horizon). Alternatively, an optimal parameter could be estimated via grid search on validation data.

8.4.2 Drift Detectors

For the remainder of this work, we use the following four drift detectors which already have been introduced in Section 8.2. While HDDDM and MK are able to process raw data input, i.e. the raw past demand, for drift detection, ADWIN and STEPD require the binary input of a classifiers performance over time. Therefore, we create an additional variable for ADWIN and STEPD which transforms the predictions into a binary variable indicating whether a prediction is correct or not. A single prediction is considered correct if the relative deviation from the actual value is within a threshold of 10%. Otherwise, this prediction is labelled false. Both HDDDM and MK process raw observations instead of classification errors for drift detection. However, the raw demand data in the taxi data set exhibits strong seasonal patterns. Therefore, we apply seasonal differencing on a daily and a weekly basis to remove seasonal effects. Therefore, monotonic trends are still included in the data whereas

seasonal trends are eliminated. This allows the drift detectors to detect incremental change more accurately.

Furthermore, we need to adapt the MK test for drift detection, since it is usually performed only once on past data (Sonali & Kumar, 2013). After a minimum number of n observations are streamed, the initial MK test is performed. In case a monotonic trend is detected, a drift is signaled and the MK test is reset. In case no drift is detected, additional n instances are streamed and the MK test is performed again on all $2 * n$ observations. Depending on whether a drift is detected, the test is reset or more instances are added to the observation window. For the experiments, we set the number of instances to $n = 168$ corresponding to one week of observations. We assume that incremental drift is captured more accurately by forcing the detectors to process more instances, thus reducing the risk to detect short-term effects.

Finally, the evaluation of concept drift handling on real-world data sets is difficult as we do not have any information about the size or duration of drifts or whether drifts are included at all in the data set (Gonçalves et al., 2014). For artificial data sets, in contrast, this information is known in advance and can be used for evaluation. Therefore, real world data is usually not evaluated by analyzing the precision of a drift detection algorithm but rather by monitoring the prediction accuracy of machine learning model in combination with a drift detector (Elwell & Polikar, 2011; Gonçalves et al., 2014). We follow this strategy in the remainder of this work.

8.5 Evaluation

The evaluation is split into two sections. At first, we perform a pre-test with different models on the NYC taxi data set in order to identify the most suitable prediction model. Subsequently, we choose the best forecasting model and apply the adaption strategies described in Section 8.4.

8.5.1 Evaluation of Pre-Test

In order to identify the best prediction model for the given data set, we perform pretests with a group of baseline models (Naive model and ARIMA) as well as a group of complex models (MLP, LSTM, XGBoost).

The *naive model* predicts just that future demand is equal to the present demand: $Y_{t+1} = Y_t$ and is a commonly used baseline (Zhu & Laptev, 2017). Regarding

the *ARIMA* model, we obtain a stationary time series by performing first order differencing as well as seasonal differencing with a lag of 24 and 168 to remove daily and weekly seasonal effects. The Augmented Dickey-Fuller test confirms stationary for the transformed time series. The final parameters for the ARIMA model are chosen in a grid search based on the model with the lowest Akaike's Information criterion and this step leads to a model of order (24, 0, 4).

Regarding the complex models, we apply a *MultiLayer Perceptron (MLP)*, *Long Short-Term Memory networks (LSTM)* and the tree-based *XGBoost* model. The MLP receives as input features the regions and the current weekday as one-hot encoded vector. Furthermore, it receives the demand during the past 24 hours as well as the demand during the same hour on the same weekday in the four past weeks. Additionally, we include cosine and sine features to depict that hours and months are cyclical features to improve prediction performance as suggested in literature (Hernández et al., 2013). We use 128 neurons in the hidden layer with a relu activation function and the network is trained using 50% dropout. XGBoost is trained on the same input data as the MLP. Regarding the LSTM, instead of one hot encoding the taxi zones, we incorporate the past demand by including a multidimensional input array which contains information about the taxi demand in each taxi zone. This way, the LSTM can capture dependencies among neighboring taxi zones.

Each model is trained on the hourly demand ranging from January 1st, 2009 to December 31st, 2010. All models are evaluated based on one-step-ahead forecasts computed for the years 2011 up to June 2018. As evaluation metrics, we apply the root mean squared error (RMSE) as well as the symmetric mean absolute percentage error (SMAPE). Applying two metrics–one absolute (RMSE) and one relative (SMAPE)–allows for a more holistic evaluation of our approach. Table 8.2 summarizes the average RMSE and SMAPE over all years and taxi zones based on the forecasts by the *static models* for all 20 taxi zones and the whole forecasting period.

Tab. 8.2.: Overall evaluation of static models.

Model	**SMAPE**	**RMSE**
Naive	27.512	132.045
ARIMA(24,0,4)	21.087	91.621
MLP	13.009	58.015
LSTM	14.007	64.047
XGBoost	11.354	57.568

Baseline models such as naive and ARIMA provide less accurate predictions than the neural network models and XGBoost. The results of the naive model are as expected

since this model does not have any parameters to learn and might adapt too quickly to unusual demand patterns. ARIMA provides a better forecast compared to the Naive model, but cannot compete with the more complex models.

Interestingly, the LSTM does provide worse prediction results compared to XGBoost and MLP which might be due to the relatively small training data set of two years and a long forecasting range. Especially, the SMAPE result for XGBoost is notable as it is far better than all other models. Presumably, the XGBoost model is especially capable to compute correct predictions during periods of low demand where large deviations severely influence the SMAPE value.

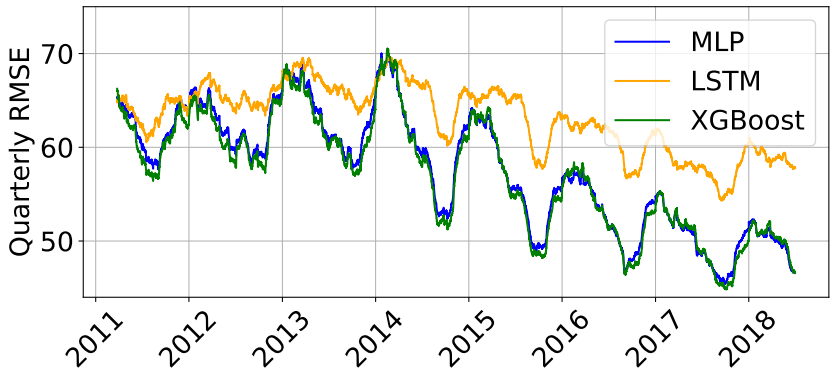


Fig. 8.3.: Quarterly rolling RMSE of static models.

For analyzing the influence of drifts on the prediction performance over time, we compute the rolling quarterly RMSE as depicted in Figure 8.3. Due to space limitations, we only consider the complex models. The RMSE is increasing until the year 2014 and then starts to decrease. This is contradictory to our intuition as we expected the RMSE to increase over time as the static models become outdated. However, the decreasing RMSE suggests an increase in performance over time. To explain this phenomenon, we need to consider the overall demand trend for yellow taxis in NYC (Figure 8.1 on page 148). The decreasing RMSE after 2014 maps well to the decreasing taxi demand after 2014. Due to the quadratic term, the RMSE penalizes more strongly higher differences in forecasts and demand which more often appear within periods of high demand.

The intuition about a decreasing performance over time is confirmed by an analysis of the SMAPE (Figure 8.4). Unlike the RMSE, the SMAPE metric considers the relative error which is independent of the actual demand level. Consequently, the SMAPE is not affected by an overall demand decrease. The results suggest that all static models are unable to capture the incremental change of the demand, resulting in decreasing prediction accuracy. Furthermore, all models exhibit an increase in the

error measures during the winter season. This can probably be explained with more fluctuating taxi demand during winter times due to extreme weather conditions such as blizzards or snow storms.

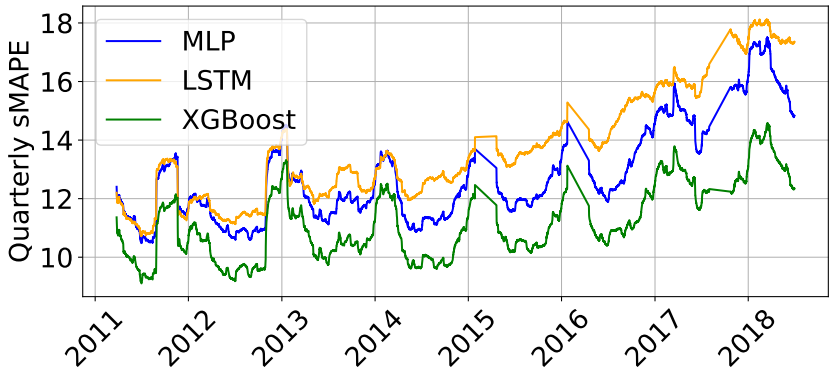


Fig. 8.4.: Quarterly rolling SMAPE of static models.

8.5.2 Evaluation of Adaptation Strategies

This section presents the results of both the *periodic* as well as the *triggered* adaptation strategies introduced in Table 8.1. For evaluation, we consider the overall average RMSE and SMAPE results based on the forecasts for all 20 taxi zones between 2011 and 2018. We report the RMSE for completeness but to assess the ability of the models to adapt to concept drift, the SMAPE measure is primarily considered as discussed in Table 8.3. Due to space limitations, we only report the results for XGBoost in this chapter. However, we have also performed all strategies with the MLP model with similar results.

Tab. 8.3.: Evaluation of periodic adaptation.

Strategy	**SMAPE**	**RMSE**	**#Actions** [1]
Static	*11.354*	*57.568*	*(-/-)*
Quarterly Update	**10.913**	54.430	(30/-)
Quarterly Retraining	10.996	55.906	(-/30)
Yearly Update	11.021	55.288	(7/-)
Yearly Retraining	11.037	56.083	(-/7)

[1] denotes (number of updates/number of retraining)

First, the results of the periodic retraining and update strategies are presented. Table 8.3 summarizes the average performance metrics as well as the number of adaptive actions performed by each strategy. The SMAPE suggests that that all adaptation strategies improve the prediction performance compared to the static

model depicted in the first row. Furthermore, we perform cross-wise Diebold-Mariano-tests (Diebold & Mariano, 2002) and can confirm that all prediction results differ significantly ($\alpha = 0.01$).

Regarding the differences between the adaptation strategies, the periodic update strategy provides better results compared to mere retraining. These findings highlight the effectiveness to update an existing model with recent observations instead of performing a complete retraining. We assume that a model which is updated incrementally better captures the underlying demand pattern since it processes a larger number of observations compared to a newly created model. Furthermore, the increased frequency–from yearly to quarterly–of adaptations improves the prediction performance. A higher frequency of adaptive actions increases the chance to adapt quickly to new concepts.

Tab. 8.4.: Evaluation of triggered adaptation.

Strategy	SMAPE	RMSE	#Actions [1]
ADWIN Retr.	11.036	56.013	(-/36)
ADWIN Upd.	10.946	54.447	(28/-)
ADWIN Sw.	**10.726**	54.582	(27/6)
STEPD Retr.	11.011	55.899	(-/16)
STEPD Upd.	10.921	54.6364	(16/-)
STEPD Sw.	**10.864**	55.218	(11/5)
HDDDM Retr.	11.017	55.942	(-/10)
HDDDM Upd.	10.955	54.972	(10/-)
HDDDM Sw.	**10.947**	55.593	(5/5)
MK Retr.	11.023	55.965	(-/24)
MK Upd.	10.914	54.516	(24/-)
MK Sw.	**10.816**	54.989	(18/6)

[1] denotes (number of updates/number of retraining)

Table 8.4 introduces the results of the triggered adaptation strategies including the retraining (retr.) as well as update (upd.) strategy and the switching scheme (sw.). The best SMAPE results among all triggered strategies are obtained by the ADWIN switching strategy followed by the MK switching strategy. The Diebold-Mariano test also confirms that those strategies provide significantly better performance results compared to the quarterly update strategy. However, both strategies trigger a large number of adaptive actions. In case it is necessary to reduce the amount of adaptive actions, the STEPD switching strategy also provides a competitive SMAPE result with a low number of adaptations. This highlights that not only frequent adaptations improve the prediction performance but also adaptations at the right point in time. Note that the column *#Actions* also serves as an indicator for the computational

burden of each strategy—a higher combined number of update and retraining steps requires also higher computational cost.

Comparing the strategies among each detector, it becomes evident that the switching scheme provides the best results in combination with any detector while the second best results are obtained through incremental updates. These findings demonstrate that the switching scheme effectively leverages the strengths of a frequent retraining and frequent incremental updates independent of the prediction model.

Figure 8.5 illustrates the performance of the best adaptation strategies over all years in the test set. The SMAPE metric in 2011 is rather similar for all depicted strategies, whereas the performance differences increase over time. During the whole forecasting period, there is a distinct gap between the performance of the adaptation strategies and the static model, indicating the effectiveness of the adaptation strategies.

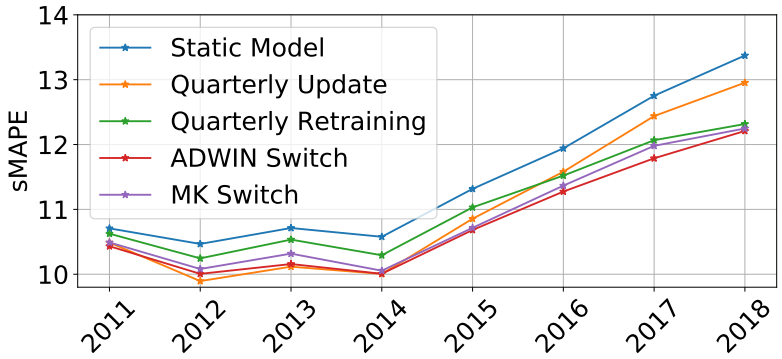


Fig. 8.5.: Yearly average SMAPE of best strategies.

Furthermore, the course of performance of the quarterly update strategy is interesting. While it provides the best performance in the first years, the prediction performance starts to decrease considerably after 2014 and becomes the least performing adaptation strategy. At the same time, the switching scheme does not exhibit such a strong decrease in predictive performance and even the quarterly retraining strategy starts to provide better results after 2016. Presumably, the XGBoost model does not benefit from endless incremental updates but rather needs to be reset at some point by creating a new model. This finding also supports our hypothesis of the underlying working principle (i.e. the need to retrain the model at some point in time) of the switching scheme.

8.5.3 Robustness Check

We perform an additional robustness check of the proposed adaptation strategies on a data set containing flight records (Ikonomovska et al., 2011). The data set contains features such as carrier name, origin and destination airport as well as date information about domestic flights in the US. It is a suitable data set for concept drift evaluation since flight records are influenced by variety of changes over time, e.g., by a rapidly increasing passenger volume over the last years or the 9/11 attacks. The objective is to predict whether a flight will be delayed (similar to Brzezinski and Stefanowski (2014)). We select a subset of the data by focusing only on departures from the busiest airport with most aircraft operations which is O'Hare International Airport in Chicago. Furthermore, only flights in the time frame from 1990 up to 2008 are considered. This limitation still leaves us with data set including approximately 5.7 million flights. As prediction model, we apply an XGBoost classifier.

Tab. 8.5.: Evaluation on flight records data set.

Strategy	**Accuracy**	**MCC**
Static	*0.7209*	*0.4458*
ADWIN Retraining	**0.7498**	0.5023
ADWIN Update	0.7426	0.4856
ADWIN Switching	0.7489	0.4986
HDDDM Training	0.7430	0.4875
HDDDM Update	0.7373	0.4758
HDDDM Switching	**0.7494**	0.5007

Similar to the taxi data set, we use two years of data for retraining or updating of a model. The initial training and the static model are both trained on data from January 1st, 1990 to December 31st, 1991. Subsequently, the different drift handling strategies are applied. Since the testing of the different handling strategies is computationally expensive, we limit our evaluation regarding drift detectors to ADWIN and HDDDM. Table 8.5 depicts the results achieved on the airline data set. Again, drift handling strategies clearly improve prediction performance compared to a static model and the switching scheme provides very competitive results regarding predictive accuracy.

Comparing both the taxi data as well as the airline data set, we can clearly see that concept drift handling, and especially the switching scheme, improves prediction performance. However, we assume that drift in case of the airline data set is less pronounced since delays are less vulnerable to changes compared to an overall demand pattern as in the taxi case. This might also be a reason why the switching scheme performs better on the taxi data set.

8.6 Conclusion

Concept drift is the phenomenon of changing data patterns over time. This work examines the effects of concept drift on the real-world demand forecasting problem of predicting taxi demand in New York City. This work contributes to the body of knowledge on multiple levels: First, we introduce the switching scheme which combines the advantages of retraining and incremental updates for machine learning models in case of incremental concept drift. Second, we benchmark different drift detectors for demand forecasting depicting their advantages and disadvantages. Third, we can clearly demonstrate the effectiveness of drift handling strategies on improving the overall prediction accuracy based on two real-word data sets, the NYC taxi demand and the flight record data set. Static models wear out and cannot guarantee a high predictive performance over time. Fourth, we can show that there are significant differences between the different drift handling strategies. Nevertheless, the difference between using *no* adaptation strategy and *any* adaptation strategy at all is more striking.

Consequently, we strongly encourage researchers and practitioners to incorporate drift handling strategies into their deployed machine learning artifacts. Both the periodic and the triggered adaptation strategies have their specific advantages. The periodic adaptation strategies are easy to understand and implement but might lead to unnecessary adaptations of the underlying machine learning model. The triggered adaptation strategies, in contrast, cause an adaptation of the prediction only in case a change in the data stream is detected. However, those strategies are more complex to implement and the choice of the right parameters is difficult and requires experience. Therefore, the selection of the right strategy does not only depend on the properties of the use case but also on the experience and skills within the organization deploying the model.

The generalizability of our results are subject to certain limitations. Despite these first promising results, our findings are based on two data sets only. In future work, we want to broaden the field of application by analyzing additional real-word data sets. This requires the identification of additional real-world data with incremental concept drift patterns. In addition, artificial data sets might also provide a valuable source for additional evaluation of the switching scheme. Due to the nature of its design, the switching scheme is rather suitable for handling incremental concept drift. Sudden or reoccurring concept drift presumably requires a different approach such as switching between two different prediction models, e.g., one model for

normal situations and one for extreme situations (Baier et al., 2020) or training a prediction model for summer and winter respectively.

This work systematically tests different adaptation strategies for handling incremental concept drift and evaluates the strategies based on their prediction performance in hindsight. However, in real-world applications, it is necessary to know upfront before deployment which strategy is best suited for a specific use case. Therefore, more research investigating the proper matching of drift handling strategies and use cases is required. Furthermore, the effect of differently sized detection windows on the prediction performance needs further research. Lastly, the triggered adaptation strategies implemented in this work are based on the assumption that true labels are received shortly after a prediction is computed. There are many fields of application where this assumption does not hold true which require an adapted handling strategy.

In general, this works shows the importance of including concept drift handling into deployed machine learning artifacts. By implementing efficient drift adaptation strategies, practitioners can create autonomous systems that—if implemented correctly with carefully adjusted alarms—require less supervision and maintenance. However, it needs to be noted that less supervision and increased automation can have negative effects, for instance *automation bias*. As research shows, employees prefer suggestions from automated systems and, over time, start to ignore contradictory information, even if they are valid (Cummings, 2004). Therefore, any automated decision-making system needs to account for this bias in its real-world implementation.

Nonetheless, our proposed artifact will generate a better prediction performance of the underlying machine learning model which in turn lead to improved service offerings or internal efficiency gains. At the same time, if efficient handling strategies are applied, employees will accustom to reliable actions by the machine learning models which results in higher trust and confidence in IS systems powered by machine learning.

Part V

Concept Drift Handling with Limited Label Availability

9 Handling by Model Uncertainty - Uncertainty Drift Detection[1]

9.1 Introduction

Across most industries, machine learning models are deployed to capture the benefits of the ever-increasing amounts of available data. When deploying models, most practitioners assume that future incoming data streams are stationary, i.e., the data generating process does not change over time. However, this assumption does not hold true for the majority of real-world applications (Aggarwal et al., 2003). In the literature, this phenomenon is referred to as *concept drift* or *dataset shift*, which usually leads to a decreasing prediction performance. Even small changes or perturbations in the distribution can cause large errors—which has been shown through, e.g., adversarial examples (Szegedy et al., 2013).

The concept drift community has developed several learning algorithms that are able to adapt incrementally (Shalev-Shwartz, 2011) or detect concept drift and trigger retrainings of a corresponding learning algorithm (Bifet & Gavaldà, 2007; Gama et al., 2004). These techniques usually require full and immediate access to ground-truth labels, which is an unrealistic assumption in most real-world use cases. As an example, let us consider a manufacturing line with a manual end-of-line quality control. By collecting sensor data from all manufacturing stations and combining this information with previously acquired quality assessments (labels) of human experts, a predictive model can be built to replace the manual quality control and thus reduce repetitive and expensive human labour. However, this prediction model is likely exposed to concept drift due to, e.g., modifications in raw materials, machine wear, ageing sensors or changing indoor temperatures due to

[1]This chapter comprises an article that was published as: Baier, L., Schlör, T., Schöffer, J., & Kühl, N. (2021). Detecting Concept Drift With Neural Network Model Uncertainty. *Working Paper*. Note: The abstract has been removed. Tables and figures were reformatted, and newly referenced to fit the structure of the thesis. Chapter and section numbering and respective cross-references were modified. Formatting and reference style was adapted and references were integrated into the overall references section of this thesis.

seasonal changes. A continuous stream of true labels for concept drift detection is not available in this use case—which is why traditional concept drift detection algorithms are not applicable.

To that end, we propose a novel concept drift detection algorithm which detects drifts based on the prediction uncertainty of a neural network at inference time, and we call this method *Uncertainty Drift Detection (UDD)*. Specifically, we derive uncertainty by applying Monte Carlo Dropout (Gal & Ghahramani, 2016). In case of a detected drift, we assume that true labels are available upon request (e.g., provided by domain experts) for retraining of the prediction model. In contrast to most drift detection algorithms, *UDD* can be used for both regression and classification problems. We evaluate *UDD* on two synthetic as well as ten real-world benchmark data sets and show that it outperforms other state-of-the-art drift detection algorithms.

9.2 Background and Related Work

9.2.1 Dataset Shift and Concept Drift

The machine learning and data mining communities use different terms to describe the phenomenon of changing data distributions over time and its impact on machine learning models (Moreno-Torres et al., 2012). *Dataset shift* (Quionero-Candela et al., 2009) is described as a change in the common probability distribution of input data x and corresponding labels y between training (tr) and test time (tst): $P_{tr}(x, y) \neq P_{tst}(x, y)$. This is similar to a common definition of *concept drift* (Gama et al., 2014; Widmer & Kubat, 1996): $P_{t_0}(x, y) \neq P_{t_1}(x, y)$, where t_0 and t_1 are two different points in time with $t_1 > t_0$. Note the difference regarding the indices: Dataset shift focuses on the difference between training and testing environment, whereas concept drift refers to the temporal structure of the data.

Dataset shift and concept drift can be further divided into different subcategories: *Covariate shift* (or *virtual drift* (Gama et al., 2014)) refers to changes in the distribution of the input data x, without affecting the distribution of labels: $P_{tr}(x) \neq P_{tst}(x)$ and $P_{tr}(y|x) = P_{tst}(y|x)$ (Moreno-Torres et al., 2012). *Real concept drift* refers to any changes in $P(y|x)$, independent of whether this change is triggered by $P(x)$ or not.

9.2.2 Handling Concept Drift

While the above terms all describe related topics, research in the area of concept drift not only deals with the detection of distributional changes and their impact on the prediction quality but also focuses on the question of how to adapt and retrain the affected model (Gama et al., 2014).

There are many reasons for changing data. Usually, it is intractable to measure all confounding factors—which is why those factors cannot directly be included in the machine learning model. Often, those factors are considered as "hidden context" of the machine learning models' environment (Tsymbal, 2004). In general, three different categories for detecting concept drift can be distinguished (Lu et al., 2019): First, error rate-based drift detection, which is also the largest group of methods (Lu et al., 2019) and aims at tracking changes in the error rate of a machine learning model. Popular algorithms in this category are the *Drift Detection Method* (DDM) (Gama et al., 2004), *Page-Hinkley* test (Page, 1954), and *ADaptive WINdowing* (ADWIN) (Bifet & Gavaldà, 2007). Note that the error rate-based drift detection necessarily requires access to ground-truth labels. Second, data distribution-based drift detection usually applies some distance function to quantify the similarity between the distributions of a reference batch of data and the current data. Algorithms in this category work on the input data x only and do not require true labels for drift detection. Popular approaches are based on tests for distribution similarity, such as Kolomogorov-Smirnov test (Raab et al., 2020). Third, the multiple hypothesis test category detects drift by combining several methods from the previous two categories.

In many real-world applications, the assumption that all true labels are available is unrealistic (Krawczyk et al., 2017). Furthermore, the acquisition of true labels from experts (e.g., in quality control) is likely expensive. Those limitations have inspired research on handling concept drift under limited label availability. In general, methods can be distinguished based on their (non-)requirement of true labels for either drift detection or for retraining of the corresponding model: The first category of algorithms assumes that true labels are available for both drift detection and retraining, but they are only provided in limited portions at specific points in time. In this category, algorithms based on active learning have been developed, where true labels for selected samples are acquired based on a certain decision criterion (Fan et al., 2004; Žliobaitė et al., 2014). Other approaches under limited label availability apply semi-supervised learning methods with clustering techniques to derive concept clusters which can be investigated for drifts (Masud et al., 2012; Wu et al., 2012). The second category requires no true labels for detection of concept

drifts, but it uses them for retraining of the model in case of a drift. One approach uses confidence scores produced by support vector machines during prediction time and compares those over time by measuring a distance between a reference window and a window of current instances (Lindstrom et al., 2013). If this distance reaches a fixed threshold, an alarm is triggered and the model is retrained using a limited set of current true labels. Other algorithms monitor the ratio of samples within the decision margin of a support vector machine for change detection (Sethi & Kantardzic, 2015). An incremental version of the Kolmogorov-Smirnov test has also been applied in this category (Dos Reis et al., 2016). The third category handles concept drift without any label access, neither for drift detection nor for retraining, e.g., by applying ongoing self-supervised learning to the underlying classifier (Dyer et al., 2014; Sun et al., 2020).

Note that the first category requires some true labels continuously over time in order to be able to detect a drift and trigger corresponding retraining. In contrast, the second category monitors the data stream for drifts based on the input data only and then requires true labels in case a drift has been detected. This is also the category that *UDD* belongs to. The third category can adapt without any true label knowledge. However, this category of algorithms also has the least adaption capabilities due to its limited knowledge of changes.

9.2.3 Uncertainty in Neural Networks

In many applications it is desirable to understand the certainty of a model's prediction. Often times, class probabilites (e.g., outputs of a softmax layer) are erroneously interpreted as a model's confidence. In fact, a model can be uncertain in its predictions even with a high softmax output for a particular class (Gal, 2016). Generally, neural networks are not good at extrapolating to unseen data (Haley & Soloway, 1992). Hence, if some unusual data is introduced to the model, the output of a softmax layer can be misleading—e.g., unjustifiably high. This likely happens in the case of concept drifts.

Generally, existing literature distinguishes two types of uncertainty: *aleatory* and *epistemic* (Der Kiureghian & Ditlevsen, 2009). The former (also called *data uncertainty*) can usually be explained by randomness in the data generation process and, e.g., corresponds to the error term in a regression setting. The latter (*statistical* or *model uncertainty*) usually results from insufficient training data. For classification tasks, uncertainty can be for instance quantified through entropy, variation ratios or mutual information (Hemmer et al., 2020).

One state-of-the-art approach to capture model uncertainty for neural networks is *Monte Carlo Dropout* (MCD) (Gal & Ghahramani, 2016). While dropout at training time has been widely used as a regularization technique to avoid overfitting (Srivastava et al., 2014), the idea of MCD is to introduce randomness in the predictions using dropout at inference time. This allows to deduce uncertainty estimates by performing multiple forward passes of a given data instance through the network and analyzing the resulting empirical distribution over the outputs or parameters.

Another family of methods to quantify predictive uncertainty is called *Deep Ensembles* (Lakshminarayanan et al., 2017). In essence, the authors of the respective paper (Lakshminarayanan et al., 2017) propose to enhance the final layer of a neural network such that the model's output is not just a single prediction but a set of distributional parameters, e.g., the mean and variance for a Gaussian distribution. The corresponding parameters can then be fitted by using the (negative) log-likelihood as loss function. For previously unseen data, the approach suggests then to train an ensemble of several neural networks with different initializations at random. The average of all variance estimates can eventually be interpreted as model uncertainty.

Other recent approaches for quantifying uncertainty in neural networks include variational inference (Blundell et al., 2015), expectation propagation (Hernández-Lobato & Adams, 2015), evidential deep learning (Sensoy et al., 2018), and stochastic gradient Markov Chain Monte Carlo methods (Welling & Teh, 2011), some of which have been applied to areas like active learning (Beluch et al., 2018; Hemmer et al., 2020) and others. A good overview of state-of-the-art methods for quantifying uncertainty, including an empirical comparison regarding their performance under dataset shift, is provided by Ovadia et al. (2019).

9.3 Methodology

When labels are expensive and their availability is limited, popular drift detection algorithms like ADWIN, DDM and Page-Hinkley are not applicable in their original form, as these algorithms detect drifts based on a change in the prediction error rate (and therefore require true labels). As described in Section 9.2.2, there are different scenarios for concept drift handling with limited label availability. In this paper, we develop a novel approach which detects drifts without access to true labels—yet it requires labels for retraining the model. For detecting drifts, we rely on the uncertainty of a (deep) neural network's predictions. Previously, it has been

shown that the uncertainty of a prediction model is correlated with the test error (Kendall & Gal, 2017; Roy et al., 2018). Thus, we argue that model uncertainty can be used as a proxy for the error rate and should therefore be a meaningful indicator of concept drift.

To investigate this hypothesis, we develop the following approach: For each data instance, we measure the uncertainty of the corresponding prediction issued by the neural network. Subsequently, this uncertainty value is used as input for the ADWIN change detection algorithm. We choose ADWIN as it as able to work with any kind of real-valued input and does not require any knowledge regarding the input distribution (Bifet & Gavaldà, 2007). Other drift detection algorithms such as DDM (Gama et al., 2004) or EDDM (Baena-Garcia et al., 2006) are designed for inputs with a Binomial distribution and are therefore not applicable to uncertainty measurements (which can have different distributions by nature).

We call our approach *Uncertainty Drift Detection (UDD)*. By applying *UDD*, we can detect significant changes in the mean uncertainty values over time. If a drift is detected, we require true labels for retraining of the model. Since there are methods for measuring uncertainty in both regression and classification settings, this approach allows to detect concept drifts for both learning tasks—as opposed to most other concept drift detection algorithms, which handle classification tasks only (Krawczyk et al., 2017). Note that *UDD* cannot detect any label shift where $P_{tr}(x) = P_{tst}(x)$ and $P_{tr}(y|x) \neq P_{tst}(y|x)$. However, we assume that in most real-world settings there is no label shift without any changes in the input distribution.

For drift detection without true label availability, input data-based drift detection, such as Kolmogorov-Smirnov (Raab et al., 2020), is generally also appropriate. However, considering solely input data bears the risk of detecting changes in features that may not be important for the prediction model. Specifically, it may occur that input data-based methods detect drifts (including expensive acquisition of new labels) where no retraining is required, because this drift will have little or no impact on the predictions of the model (e.g., virtual drift where $P_{t_0}(x) \neq P_{t_1}(x)$ and $P_{t_0}(y|x) = P_{t_1}(y|x)$). Our uncertainty-based approach, on the other hand, detects only changes in the input data that also have an impact (as reflected by the uncertainty) on the predictions.

For measuring uncertainty and computing predictions, we apply Monte Carlo Dropout (MCD) because it showed the best performance during our experiments. However, note that the proposed method can be easily extended to use other uncertainty estimates (e.g., Deep Ensembles) as well. In practice, MCD applies dropout at inference time with a different filter for each stochastic forward pass through

the network. We denote T the number of stochastic forward passes. Predictions $\widehat{p}(y|x)$ are computed by averaging the predictions for each forward pass T given the samples w_i of model parameters from the dropout distribution and the input data x:

$$\widehat{p}(y|x) = \frac{1}{T}\sum_{i=1}^{T} p_i(y|w_i, x)\,. \tag{9.1}$$

Regression and classification require different methods for determining predictive uncertainty. We choose to evaluate the uncertainty for *classification* tasks based on Shannon's entropy H over all different label classes K:

$$H\left[\widehat{p}(y|x)\right] = -\sum_{k=1}^{K} \widehat{p}(y=k|x) * \log_2 \widehat{p}(y=k|x)\,. \tag{9.2}$$

For *regression* tasks, uncertainty estimates can be obtained by computing the variance of the empirical distribution of the T stochastic forward passes through the network (Gal & Ghahramani, 2016):

$$\widehat{\sigma}^2 = \frac{1}{T}\sum_{i=1}^{T} \left(p_i(y|w_i, x) - \widehat{p}(y|x)\right)^2\,. \tag{9.3}$$

Real-world data streams for concept drift handling are heterogeneous, e.g., in their number of class labels and size (Souza et al., 2020). This variability is also reflected by heterogeneous distributions of the respective uncertainty indicator. Furthermore, due to different approaches for computing uncertainty, this indicator varies significantly in scale and fluctuation between regression and classification problems. Therefore, ADWIN has to be adjusted to each data stream, which can be achieved by setting its sensitivity parameter $\alpha \in (0, 1)$: A change is detected when two sub-windows of a recent window of observations exhibit an absolute difference in means larger than α.

New data instances arrive individually and are predicted at the time of arrival. The obtained uncertainty U_t (either expressed as entropy or variance) from the prediction at time t is used as input for an ADWIN change detector. Once a drift is detected, a retraining of the prediction model is performed. For retraining, *UDD* uses the most recent data instances in addition to the original training data. This way, we can ensure that the model (a) can adapt to new concepts and (b) has enough training data for good generalization. Algorithm 2 on page 170 describes the required steps for *UDD* in a regression (U_t equals variance of prediction $\widehat{\sigma}^2$) or classification setting (U_t equals entropy of prediction H_t).

Algorithm 2 Uncertainty Drift Detection

Input: Trained model M; Data stream $\mathcal{D}$; Training data $\mathcal{D}_{tr}$
Output: Prediction $\widehat{y}_t$ at time t
repeat
  Receive incoming instance x_t
  $\widehat{y}_t, U_t \leftarrow M$.predict($x_t$)
  Add U_t to $ADWIN$
  if $ADWIN$ detects change **then**
    Acquire most recent labels y_{recent}
    M.train($\mathcal{D}_{tr} \cup \mathcal{D}_{recent}$)
  end if
until $\mathcal{D}$ ends

9.4 Experiments

For evaluation purposes, we conduct extensive experiments to compare *UDD* with several competitive benchmark strategies on two synthetic and ten real-world data sets. This stands in contrast to most concept drift literature, where new methods are mainly evaluated on simulated data sets with artificially induced concept drifts. The code for our experiments can be found under https://github.com/anonymous-account-research/uncertainty-drift-detection.

9.4.1 Experimental Setup

Throughout the experiments, for MCD, we set the number of stochastic forward passes $T = 100$ for regression tasks and $T = 50$ for classification tasks. Regarding the deep feed forward network, we vary the structure between three to five hidden layers with relu activation functions depending on the data set. Each hidden layer is followed by a dropout layer with dropout rate 0.1 or 0.2, as it is proposed in the original MCD paper (Gal & Ghahramani, 2016). Details regarding the experimental setup for each data set can be found in Table A.5 in the appendix.

For initial model training, we use the first five percent of a data stream's instances. We perform a parameter optimization for *UDD* by requiring the associated ADWIN algorithm to detect one drift on a given validation data set—this yields a concrete value for the sensitivity parameter α. If no drifts are detected on the validation data with the initial value for α, we assume that no drifts are present in the validation data and α is set to the `scikit-multiflow` (Montiel et al., 2018) default value of 0.002. We use the ten percent of instances following the initial training data as validation data. Every time we detect a drift, we provide the last data instances as

well as corresponding labels equivalent to one percent of the overall data stream's length. The exemplary partitioning of a data stream is depicted in Figure 9.1.

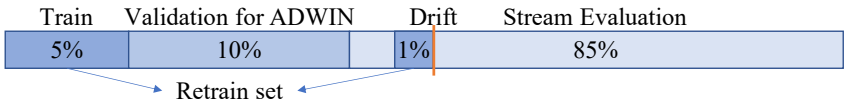


Fig. 9.1.: Partitioning of data stream.

In order to benchmark *UDD*, we compare it against six different strategies within two groups. The first group of strategies handles concept drift with *Limited Label Availability* whereas the second group of strategies allows for *Unlimited Label Availability*.

Limited Label Availability

The first benchmark is a non-adaptive model, *No Retraining (No Retr.)*. This strategy does not test for drifts and the machine learning model is only trained once with the initial training set. The performance of this strategy constitutes a lower-bound benchmark.

The second benchmark is an *Uninformed Retraining (Uninf.)* strategy which randomly draws retraining points out of all possible time stamps included in the respective data stream. To ensure comparability, we set the number of retrainings of this strategy to be equal to the *UDD* approach. This also ensures that the uninformed retraining strategy receives access to the same number of true labels. Otherwise, a strategy with access to more true labels will likely perform better due to larger training set sizes. To get a reliable performance estimate for this strategy, we repeat this experiment five times and average the results.

The third benchmark, *Equal Distribution (Equal D.)*, is similar to the previous benchmark but the retraining points are equally distributed over the course of the data stream.

The *Kolmogorov-Smirnov* test-based drift detector (*KSWIN*) belongs to the category of input data-based drift detection and works by individually investigating each input feature for changes. We optimize its sensitivity parameter α with the same procedure as for *UDD*. This detector is known to produce many false positive concept drift signals, due to multiple hypothesis testing (Raab et al., 2020). Again, we restrict the number of retrainings to be equal to the *UDD* approach. If this strategy detects more drifts, detected drifts are sorted by the order of their p-values and only the top

drifts are considered for retraining. For this strategy, we use the `scikit-multiflow` (Montiel et al., 2018) implementation *KSWIN* (Kolmogorov-Smirnov WINdowing) with the following parameters: $window\_size = 200$, $stat\_size = 100$.

Unlimited Label Availability

The second group of strategies is not restricted with respect to the amount of allowed retrainings. Therefore, they are *not* an appropriate benchmark in a context where true labels are scarce. We still include these strategies since they serve as an upper-bound performance benchmark. This allows us to estimate the performance loss when confronted with a situation where full label availability is infeasible.

The *Kolmogorov-Smirnov* test with *unlimited* retrainings (*KSWIN(unl.)*) benchmark is similar to the previous *KSWIN* strategy but without restricting the number of retrainings. Therefore, all detected drifts trigger a retraining of the prediction model.

The last benchmark is the *ADWIN* change detection algorithm applied to the prediction error rate. This strategy already requires all true labels for the computation of the error rate and therefore for drift detection. Note that all other strategies manage the drift detection without any true labels and then only require labels for retraining. For this method, we use the `scikit-multiflow` (Montiel et al., 2018) implementation with default parameter settings.

9.4.2 Data Sets

For evaluation, we consider two synthetic data sets (Friedman and Mixed) and ten real-world data sets. The **Friedman** regression data set (Friedman, 1991) consists of ten features that are each drawn from a uniform distribution from the interval $[0, 1]$. The first five features are relevant for the prediction task, the remaining five are noise. The **Mixed** classification data set is inspired by Gama et al. (Gama et al., 2004) and contains six features where two features are Boolean and the other four features are drawn from a discrete distribution. Two of the features are noise which do not influence the classification function. By modifying the distribution of some features, we can either induce real or virtual concept drifts (see Section 9.2.1) in both the Friedman and the Mixed data set.

Furthermore, ten real-world data sets—eight classification and two regression tasks—are used for the evaluation of the *UDD* method. The characteristics of the real-world

data sets regarding sample size, number of features, and targets can be found in the appendix in Table A.6. The **Air Quality** data set (De Vito et al., 2008) contains measurements from five metal oxide chemical sensors, a temperature, and a humidity sensor. The learning task is to predict the benzene concentration, which is a proxy for air pollution. The real benzene concentration is measured with a specialized, expensive sensor. Concept drift is present due to seasonal weather changes. The **Bike Sharing** data set (Fanaee-T & Gama, 2014) provides hourly rental data for a bike sharing system between 2011 and 2012 in Washington, D.C., with the objective to predict the hourly demand for bike rentals. Concept drift is again assumed to be present due to seasonal weather changes.

All following classification data sets are taken from the USP Data Stream Repository (Souza et al., 2020): The various **Insects** data sets were gathered by controlled experiments on the use of optical sensors to classify six types of different flying insects (three species with two sexes, respectively). Concept drift is artificially induced by changes in temperature. The **Abrupt** data set contains five sudden changes in temperature, whereas in the Incremental (**Inc**) data set, temperature is slowly increased over time. The Incremental Abrupt (**IncAbr**) data set has three cycles of incremental changes with additional abrupt drifts included as well. In the Incremental Reoccurring (**IncReo**) data set, the temperature increases incrementally within several cycles. The **KDDCUP99** data set contains TCP connection records from a local area network. Features comprise information such as connection duration, protocol type, and transmitted bytes. The learning task is to recognize whether the connection is normal or relates to one of 22 different types of attacks. The **Gas Sensor** data set was collected over 36 months at a gas delivery system. For each recording, one of six gases is diluted in synthetic dry air inside a sensing chamber, and the objective is to identify the respective gas. Both sensor drift (due to aging) and concept drift (due to external alterations) are included in the data. The **Electricity** data set was gathered at the Australian New South Wales Electricity Market. Each record contains information about recent electricity consumption and market prices. The learning task is to predict whether the market price will increase or decrease compared to the last 24 hours. The **Rialto Bridge** Timelapse data set contains images taken by a webcam close to the Rialto Bridge in Venice, Italy during May and June 2016. The objective is to correctly classify nearby buildings with concept drift occurring due to changing weather and lighting conditions.

9.4.3 Performance Metrics

Evaluating concept drift detection on real-world data sets is a challenging endeavor as most real-world data sets do not have specified drift points. Specifically, for most real-world data, it is intractable to measure the accuracy of drift detection itself. Therefore, we perform two different analyses regarding the behaviour of *UDD* in this work. First, we apply *UDD* on two synthetic data sets to specifically evaluate its drift detection capabilities. Second, we perform extensive experiments to investigate its performance on ten real-world data sets.

For synthetic data sets, the real drift points are known, which allows to compute metrics regarding the drift detection capabilities of a drift detector (Bifet, Read, Pfahringer, et al., 2013). In this work, we compute the Mean Time to Detection (MTD), the False Alarm Count (FAC), and the missed detection count (MDC).

In contrast, an appropriate evaluation for real-world data sets is more difficult. However, one can assume that a drift is present when the prediction performance of a static model decreases over time. Since the real drift points are unknown, we evaluate the different strategies based on their prediction performance, as it is common in the concept drift literature (Elwell & Polikar, 2011). For regression tasks, we apply the Root Mean Squared Error (RMSE), and we use the Matthews Correlation Coefficient (MCC) for classification tasks. MCC is a popular metric for classification settings as it can also handle data sets with class imbalance (Chicco & Jurman, 2020).

9.4.4 Analysis on Synthetic Data Sets

To test the capabilities of *UDD*, we analyze its behaviour when applied on two synthetic data sets (Friedman and Mixed). Both data sets contain virtual as well as real concept drifts. Virtual drifts refer to changes in the input data with no effect on the resulting label. Hence, *UDD* should *not* raise an alarm for these drifts as a retraining of the machine learning model in this case is unnecessary. Recall that this kind of analysis is only feasible on synthetic data sets, as we do not have any knowledge regarding the type of concept drift as well as its timing on real-world data sets. On the synthetic data sets, we test *UDD* and *KSWIN(unl.)* as they both do not require true labels for drift detection. The parameters of both approaches are optimized based on a validation set which includes one drift (see Section 9.4.1).

Figure 9.2 shows the trajectory of the predictive uncertainty over the course of the Friedman data set. The uncertainty changes significantly each time a real concept

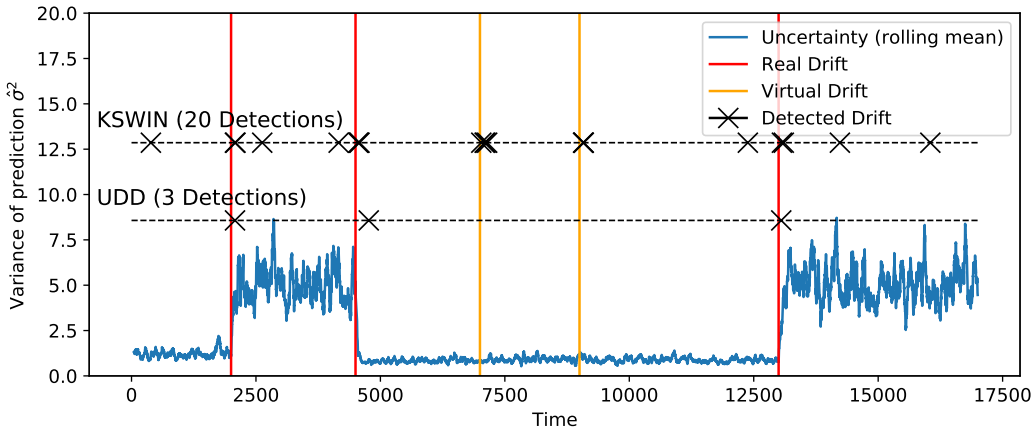


Fig. 9.2.: Behaviour of *UDD* and *KSWIN* on synthetic Friedman data set.

drift occurs. Accordingly, this is also detected by *UDD*. As expected, the two virtual drifts (marked by orange vertical lines in the figure) do not trigger a drift detection. In contrast, the input data-based detection (*KSWIN*) detects also these virtual drifts. Furthermore, note the overall large number (20) of detected drifts by *KSWIN* despite a parameter optimization. This illustrates *KSWIN*'s problem of high reactivity leading to several false-positive drift detections. Figure A.1 in the appendix depicts this analysis for the Mixed data set.

Since the real drift points for the synthetic data sets are known, we compute the mean time to detection (MTD), the false alarm count (FAC), and the missed detection count (MDC) for both strategies in Table 9.1. *UDD* correctly identifies all real concept drifts in both data sets. Furthermore, no false alarms are raised. However, *KSWIN* achieves lower MTD values compared to *UDD* in both data sets, which means that *KSWIN* recognizes concept drifts faster. This can likely be explained by the high sensitivity of *KSWIN* regarding changes. However, this sensitivity also leads to large numbers of false alarms (17 and 11, respectively), as depicted in Table 9.1. Such a behaviour is especially detrimental in scenarios where the acquisition of true labels is expensive. Each time a false alarm is raised, new true labels must be acquired at a high cost—even though a retraining is not required since no real concept drift has occurred.

Tab. 9.1.: Evaluation on synthetic data sets.

Data Set	Drift Detector	MTD	FAC	MDC
Friedman	*UDD*	132.7	0	0
	KSWIN	65.7	17	0
Mixed	*UDD*	247.3	0	0
	KSWIN	50.3	11	0

Tab. 9.2.: RMSE (the lower the better) on *regression* benchmark data sets. Number of retrainings in brackets (the lower the less computationally expensive). *No Retraining* depicts the lower-bound benchmark, while *KSWIN(unl.)* and *ADWIN* represent the upper-bound performance benchmark.

		Limited Label Avail.				Unlimited Label Avail.	
Data Set	**No Retr.**	**Uninf.**	**Equal D.**	**KSWIN**	**UDD**	***KSWIN(unl.)***	***ADWIN***
Air Quality	1.170 (0)	1.383 (14)	1.231 (14)	1.285 (14)	**1.151** (14)	1.304 (19)	1.387 (12)
Bike Sharing	171.47 (0)	170.00 (5)	144.94 (5)	143.88 (5)	**129.93** (5)	120.69 (27)	127.07 (8)

Tab. 9.3.: MCC (the higher the better) on *classification* benchmark data sets. Number of retrainings in brackets (the lower the less computationally expensive). *No Retraining* depicts the lower-bound benchmark, while *KSWIN(unl.)* and *ADWIN* represent the upper-bound performance benchmark.

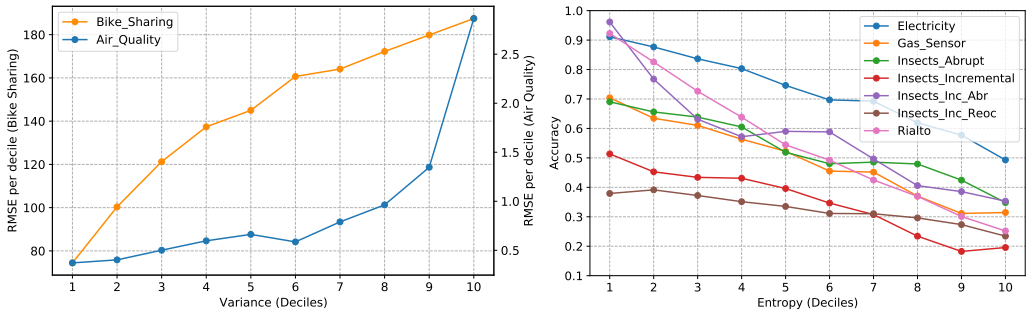
		Limited Label Avail.				Unlimited Label Avail.	
Data Set	**No Retr.**	**Uninf.**	**Equal D.**	**KSWIN**	**UDD**	***KSWIN(unl.)***	***ADWIN***
Insects Abrupt	0.452 (0)	0.468 (9)	0.475 (9)	0.456 (9)	**0.516** (9)	0.521 (192)	0.497 (9)
Insects Inc	0.052 (0)	0.210 (4)	0.211 (4)	0.191 (4)	**0.242** (4)	0.238 (27)	0.251 (3)
Insects IncAbr	0.292 (0)	0.463 (22)	0.483 (22)	0.464 (22)	**0.522** (22)	0.488 (107)	0.516 (23)
Insects IncReo	0.114 (0)	0.190 (10)	0.197 (10)	0.126 (10)	**0.208** (10)	0.218 (149)	0.239 (13)
KDDCUP99	0.663 (0)	0.830 (20)	0.873 (20)	0.772 (20)	**0.964** (20)	0.986 (345)	0.984 (61)
Gas Sensor	0.255 (0)	0.472 (39)	0.469 (39)	0.325 (39)	**0.484** (39)	0.454 (149)	0.480 (49)
Electricity	0.139 (0)	0.372 (13)	0.362 (13)	0.254 (13)	**0.436** (13)	0.511 (269)	0.471 (45)
Rialto Bridge	0.534 (0)	0.558 (14)	0.561 (14)	**0.583** (14)	**0.583** (14)	0.586 (17)	0.600 (116)

9.4.5 Experimental Results

Both *UDD* and *KSWIN* require as input a suitable value for α, which determines their sensitivity regarding concept drift detection. Since the data sets included in this experiment are fundamentally different from each other (e.g., different number of class labels), individual values of α are required for each data set. As described in Section 9.4.1, we determine the respective value for both strategies by performing a test on a validation data set. The parameter values for *UDD* for each data set are depicted in Table A.7 in the appendix.

A summary of the experimental results on all data sets is provided in Table 9.2 for regression data sets (RMSE) and in Table 9.3 for classification (MCC). Furthermore, we provide an additional view on the results by depicting the SMAPE metric in Table A.8 and the F1-score in Table A.9 in the appendix. For the evaluation, we primarily focus on the first five columns of the table which as a group can be characterized by only requiring a limited amount of true labels. This is also illustrated by the values in parentheses which describe how often the corresponding machine learning models are retrained. As explained in Section 9.4.1, *KSWIN(unl.)* and *ADWIN* serve as an upper-bound benchmark due to their requirement of full label availability.

The best strategy with limited label availability per data set is marked in bold. For both regression data sets, *UDD* outperforms the other four strategies. Regarding



(a) Regression **(b)** Classification

Fig. 9.3.: Relationship between deciles of uncertainty and prediction performance.

the classification tasks, *UDD* achieves the best prediction performance on seven out of eight data sets and always outperforms the strategies *No Retraining*, *Uninformed* and *Equal Distribution*. Solely for the Rialto data set, the strategy based on *KSWIN* performs equally well, which might be explained with rather significant changes in individual input features that can be detected well with *KSWIN*. As expected, the *No Retraining* strategy usually performs worst. Interestingly, the *Uninformed* already achieves good prediction performance and sometimes even outperforms the *KSWIN* strategy, especially for the regression tasks. By design, the number of retrainings is equal for all four strategies—*Uninformed*, *Equal Distribution*, *KSWIN*, and *UDD*.

The right two columns in both Table 9.2 and Table 9.3 show the prediction performance of the *KSWIN(unl.)* and *ADWIN* strategy. As expected, these strategies usually outperform all other strategies but also require significantly more true labels for retraining. For the KDDCUP99 data set, the difference in amounts of retrainings for *UDD* compared to *KSWIN(unl.)* is most striking: While *UDD* requires 16 retraining, *KSWIN(unl.)* performs 345 retrainings in total. Yet, the difference in predictive performance is rather small. Also, recall that the *ADWIN* strategy requires *all* true labels for drift detection itself. For the Insects Abrupt, Insects IncAbr, and the Gas Sensor data set, the *UDD* strategy performs even better than *ADWIN*.

We also investigate the average prediction performance for *UDD* based on the level of uncertainty in Figure 9.3. Per data set, we sort instances in deciles, from instances with lowest uncertainty (decile 1) up to instances with highest uncertainty (decile 10) based on entropy H or variance $\widehat{\sigma}^2$, respectively. Subsequently, we compute the average prediction performance per decile. As expected, the RMSE for regression data sets increases with rising uncertainty, as shown in the left plot (a). The right plot (b) shows the classification data sets—decile 1 shows the highest mean accuracy

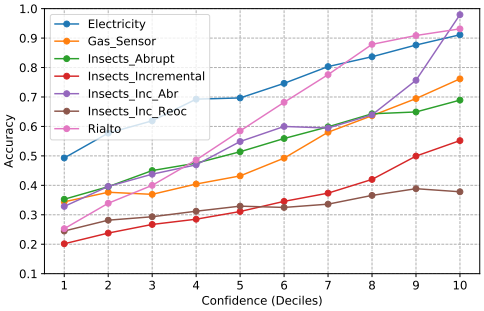


Fig. 9.4.: Relationship between deciles of confidence and accuracy.

and decile 10 the lowest.[2] Thus, Figure 9.3 confirms our assumption that uncertainty represents a proxy for the error metric.

For classification tasks, we additionally analyze the relationship between confidence and accuracy (similar to Lakshminarayanan et al. (2017)). We define confidence $c := \max_k \widehat{p}(y = k|x)$ as the highest predicted probability for one class of a specific data instance. All instances of a data set are sorted into confidence deciles, where decile 10 contains all instances with highest confidence. Figure 9.4 depicts the relationship. As expected, the mean accuracy score increases with larger confidence.

9.5 Conclusion

In this work, we have introduced the *Uncertainty Drift Detection (UDD)* algorithm for concept drift detection. This algorithm does not depend on true labels for detection of concept drift, and—only in case of a detected drift—requires access to a limited set of true labels for retraining of the prediction model. Therefore, this algorithm is especially suitable for drift handling in deployed machine learning settings within real-world environments where the acquisition of true labels is expensive (e.g., quality control). Standard drift detection algorithms such as DDM and ADWIN are not applicable in such settings because they require access to the entire set of true labels. Our approach is based on the uncertainties derived from a deep neural network in combination with Monte Carlo Dropout. Drifts are detected by applying the ADWIN change detector on the stream of uncertainty values over time. In contrast to most existing drift detection algorithms, our approach is able to detect drift in both regression and classification settings. We have performed an extensive evaluation on two synthetic as well as ten real-world concept drift data sets to

[2]The KDDCUP99 data set is not included in Figure 9.3 and 9.4 because deciles cannot be computed due to the highly skewed distribution of entropy and confidence values.

demonstrate the effectiveness of *UDD* for concept drift handling in comparison to other state-of-the-art strategies.

In future work, we aim to improve the *UDD* method by including active learning methods. Including only those instances with high uncertainty in the retraining set rather than all recent instances could further improve the prediction performance.

10 Handling by Outlier Detection - the Two-Step Prediction Method[1]

10.1 Introduction

Due to the explosion of data in recent years, supervised machine learning plays an important role in nearly all fields of business, ranging from marketing to scientific-, health- and security-related applications (Chen et al., 2012). Many companies rely on deployed machine learning models for increasing process efficiency or for introducing innovative services (Schüritz & Satzger, 2016). Besides the mentioned increased availability of data, this growth in popularity can also be explained by a massive increase in computation power in recent years (Jordan & Mitchell, 2015).

However, there are also areas of application for supervised machine learning where computational resources are strictly limited. This especially applies to machine learning models in production systems. For instance, companies might generally restrict the connection of sensible data and machine learning models to the infrastructure of cloud providers due to the fear of loosing data and corresponding intellectual property (Zhang et al., 2010). In other cases, it might be just technically unfeasible to connect specific parts or components to cloud services. In both cases, it is necessary to rely on available local computing resources. Applications on mobile devices are a typical example for this case (Oneto et al., 2015). Therefore, we regard resource considerations as one constraint for motivating this work.

The second constraint relates to a machine learning model in operation which does not receive any ground truth labels for the predictions that are issued (Raykar et al., 2009; Yu et al., 2014). This problem needs to be considered in the context

[1] This chapter comprises an article that was published as: Baier, L., Kühl, N., & Schmitt, J. (2021). Increasing Robustness for Machine Learning Services in Challenging Environments – Limited Resources and No Label Feedback. *Proceedings of SAI Intelligent Systems Conference (IntelliSys) [forthcoming]*. Note: The abstract has been removed. Tables and figures were reformatted, and newly referenced to fit the structure of the thesis. Chapter, section and research question numbering and respective cross-references were modified. Formatting and reference style was adapted and references were integrated into the overall references section of this thesis.

of data streams because basically all models which are deployed in productive information systems receive input data as data streams and continuously issue predictions over months or even years for those very data instances. Unfortunately, data streams usually evolve over time leading to changes in the underlying data patterns (Žliobaitė, 2010) which require the adequate adaption of a prediction model (Baier, Kühl, et al., 2019). However, usual adaptation strategies for model updates (Tsymbal, 2004) cannot be applied because no feedback with regard to the prediction performance is received (Gama et al., 2014). Both constraints are expressed in the central research question of the work at hand:

Research Question F
How to increase robustness for supervised machine learning services without label feedback and limited infrastructure resources?

We introduce a novel prediction method artifact ("machine learning service") which addresses both mentioned constraints: limited computational sources and no availability of true labels after deployment. To achieve this, our artifact combines an outlier detection with a robust machine learning model. We assess available outlier detection models as well as prediction models with regard to their suitability based on several criteria. Subsequently, we select appropriate models for our prediction method. Our suggested method is evaluated based on a use case from a large global automotive supplier. We show that we can increase prediction performance significantly with our method compared to normal prediction models while at the same time restricting necessary memory and computation time requirements. By the introduction of the method, this work aims at contributing to increase the acceptance of machine learning solutions for real-world applications. So far, many companies are still skeptical about trusting deployed machine learning solutions for automated decision-making in production environments (Zhou, 2017).

The remainder of this work is structured as follows: The upcoming Section 10.2 presents foundations on which we base our research. Section 10.3 introduces the artifact requirements, while Section 10.4 gives an overview of different options to fulfill these specified requirements. Section 10.5 describes the evaluation of our approach with an industry use case. Section 10.6 discusses our results, describes implications and outlines future research.

10.2 Foundations

To lay the necessary foundations for the remainder of this work, we first briefly introduce machine learning followed by an overview on concept drift and outlier detection.

10.2.1 Machine Learning

Traditionally, machine learning approaches are divided into supervised, unsupervised as well as reinforcement learning (Alsheikh et al., 2014). Supervised machine learning depends on labeled examples in the training data. In contrast, unsupervised machine learning aims at detecting unknown relationships and patterns in the data. In reinforcement learning, an agent learns to interact with its environment by receiving specific rewards. We focus on supervised approaches, as most real-word applications of machine learning are of supervised nature (Jordan & Mitchell, 2015; Kotsiantis, 2007).

However, many machine learning models need to be operated in environments with limited resources. For instance, the training process of machine learning models with gradient-descent optimization has been analyzed to improve computational efficiency (Gupta et al., 2015). Other approaches in literature aim to reduce the computational requirements for calculating the actual prediction in operation, e.g., for the application of support vector machines for activity recognition (Anguita et al., 2012) or for computer vision tasks in cars (Anguita et al., 2007). Specialized distributed approaches have been developed for the training and deployment of machine learning on network edges (Wang et al., 2018). A lot of research in this context has been dedicated to the application of machine learning in wireless sensor networks (An et al., 2018; Hu & Hao, 2012) where computations have to be performed on small independent and distributed sensors (Alsheikh et al., 2014). Additional challenges arise when machine learning models are deployed on local and isolated computing units such as mobile devices. Challenges for mobile devices are a high power consumption as well as limited computational power and memory storage (Oneto et al., 2015).

10.2.2 Concept Drift

Companies usually apply machine learning models to make predictions for specific services on a stream of unseen incoming data. However, data tends to evolve and change over time. Therefore, predictions issued by models which have been trained on past data may become less accurate or opportunities for performance improvements might be missed (Žliobaitė et al., 2016). In computer science, the challenge of changing data streams and its implications for machine learning are described with the term concept drift (Widmer & Kubat, 1996). A concept $p(X, y)$ is the joint probability distribution over a set of input features X and the target variable y. The definition of concept drift refers to a change of a concept between two time points t_0 and t_1 with $p_{t0}(X, y) \neq p_{t1}(X, y)$ (Gama et al., 2014).

An example for an application with concept drift over time is a machine learning service which monitors the output quality in a chemical production process and predicts corresponding failures (Žliobaitė et al., 2016). Sensors connected to the machine will generate the necessary data input. However, sensors wear out over time leading to different data measurements. The previously deployed machine learning model is not prepared for this drift as this data pattern has not been included in the training data set. Thus, correct quality predictions are difficult to make in the long run (Kadlec & Gabrys, 2011). Detecting concept drift usually relies on a continuous evaluation of the prediction performance over time (Gama et al., 2014). Less popular approaches focus on the computation of drift detection features on the input data (Cavalcante et al., 2016). Examples for concept drift handling can be found in various domains such as mobility Baier et al., 2020 or fraud detection Dal Pozzolo et al., 2017.

10.2.3 Outlier Detection

Outliers are described as data instances that appear to be inconsistent with the remaining data instances in the same sample (Barnett & Lewis, 1994). They are also referred to as abnormalities or anomalies. The detection of outliers is an important task in many different application scenarios. These range from fraud detection to the detection of unauthorized intruders in computer networks up to fault diagnosis in complex machine parts such as detecting aircraft engine rotation defects (Hodge & Austin, 2004).

Different outlier detection methods are usually divided into unsupervised and supervised methods based on the availability of true labels with regard to the outlierness

of data instances (Aggarwal, 2015). In real-world settings, the detection of outliers typically brings along many challenges (Hodge & Austin, 2004; Zimek et al., 2014). For instance, data contains natural noise which is similar to actual outliers and therefore it is difficult to distinguish between noise and outliers. Furthermore, true outlier labels for data instances are rarely available and therefore complicate the proper validation of different outlier detection methods (Chandola et al., 2007). This fact is one of the reasons for the popularity of unsupervised approaches. Of those, especially distance- or neighborhood-based techniques have gained a lot of popularity because they are relatively parameter-free and easy to adapt (Orair et al., 2010). However, the computational complexity during application is tremendous. In contrast, statistical approaches allow for easier computation due to their model-based structure, but they rely on an assumption regarding the underlying data distribution which often does not hold true (Chandola et al., 2007). There are numerous reviews of existing, mostly unsupervised, outlier detection methods (e.g., Chandola et al. (2007) and Hodge and Austin (2004)) which we use as input for the development of our prediction method.

10.3 Problem Definition and Requirements

In this work, we consider a machine learning application with two specific problems. First, no ground truth labels during deployment can be acquired and therefore monitoring the prediction performance is impossible. This can happen for instance when models are deployed to enhance the functionality of a small entity which is integrated into a larger system (e.g., a large machine consisting of several different components). Due to the embedding into the larger system, necessary sensors cannot be installed and no ground truth labels can be acquired. Still, also in this case, machine learning models are exposed to concept drift which needs to be addressed. These facts are summarized in problem 1 (P1).

P1: *No ground truth labels are available after deployment of the machine learning model.*

Second, the necessary computations for the execution of the prediction method need to be performed on a local computing unit. Neither the integration of a more powerful computing unit nor the connection of the prediction to cloud services are feasible solutions. Therefore, only limited resources are available which is depicted in problem 2 (P2), a typical problem in real-world applications of machine learning (Malikopoulos et al., 2007; Vong et al., 2006; Xie et al., 2018).

P2: *The prediction method needs to run on a local computing unit.*

Based on the problem characteristics (P1, P2), we derive the following design requirements (R) for the prediction method.

R1: The developed method must show robust behaviour to concept drift and continuously deliver acceptable prediction performance. Ground truth labels for the predictions issued by the method in operation are not accessible.

R2: The prediction method needs to operate with limited storage space for saving necessary data (e.g., model weights and other parameters) as the storage space on the local computing unit is limited (memory requirement).

R3: The computational complexity of the prediction method is constrained by the computing power of the local computing unit. This requirement refers to the amount of operations that need to be carried out for a prediction during operation and is therefore closely related to time constraints for the computation of a prediction.

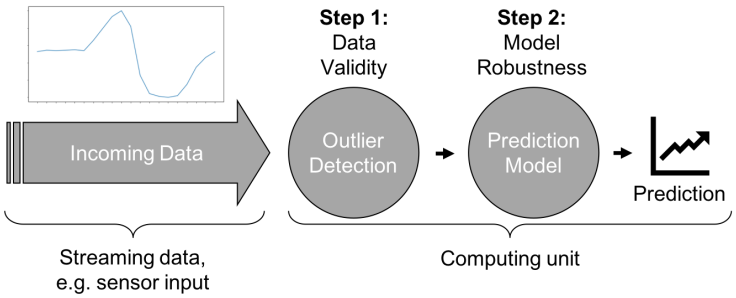


Fig. 10.1.: Operating principle of the prediction method.

The design of the prediction method is depicted in Figure 10.1. The individual components are designed to ensure that the overall method aligns to the requirements (R1-R3). At first, streaming data (sensor measurements) is transferred to a computing unit. As described above, no ground truth labels for any of the predictions can be acquired. Therefore, it is impossible to continuously adapt the model to data changes. Instead, we suggest to control the input data for the prediction model (step 1). An outlier detection is implemented before the prediction model which ensures the proper validity of the new incoming data. Data validity implies that new data is similar to the data that the prediction model has been trained on in the training process. Data instances that exhibit unusual patterns compared to the training data will be filtered out and no prediction for those data instances will be issued. In this case, the whole system will rely on the previous prediction. With regard to time series, this behavior is acceptable because data instances are highly auto-correlated, e.g., we do not expect any discontinuities in the prediction target. In general, it

is preferable to receive a prediction for every new data instance. However, it is better to receive no prediction than an entirely false prediction. Otherwise, the control unit of the whole system will adapt its behavior based on this false prediction and trigger corresponding actions which might have negative effects. Due to this dependency, it is better to accept the limitation that predictions are only issued for normal data instances. We prefer to skip adaptations of the whole system at some time points instead of performing a false adaptation based on a false prediction with unforeseeable consequences. In case the prediction method is constantly detecting outliers, an alert will be created.

The second step in the prediction method is the application of the prediction model (step 2). In case a data instance is not marked as an outlier, the relevant data is transmitted to the prediction model. One evaluation criteria for the prediction model needs to be the achieved prediction performance. Additionally, it is crucial that the prediction model computes robust predictions which means that the prediction performance does not change significantly over time.

According to R2 and R3, potential methods for step 1 and step 2 respectively need to have low computational as well as memory requirements so that these can be deployed on the local computing unit.

10.4 Design Options

In the following chapter, we introduce the different available methods which can be implemented to build the two-step prediction method.

10.4.1 Step 1: Data Validity

The first step in the prediction method (Figure 10.1) needs to ensure the data validity of the incoming data instances (R1). This means that the new incoming data in operation must be similar to the training data. The objective of outlier detection is the identification of data instances that are different to other data instances. Therefore, outlier detection is a suitable approach to guarantee validity of new data instances. Since no labelled data regarding the outlierness is available, we rely on unsupervised models in the following. Unsupervised outlier detection models can be differentiated into the following groups (Aggarwal, 2015): Extreme value models, Clustering models, Distance-based models, Density-based models, Probabilistic models and Information-theoretic models.

Considering these groups of algorithms, we need to make a selection that is suitable for the overall prediction method. Distance-based and density-based models require the storage of all initial training instances for a proper functionality in practice. This means that the whole initial training dataset needs to be stored on the computing unit. Due to the memory and computational requirements (c.f. design requirement R2 and R3 in Section 10.3), these groups of models are not feasible for the overall approach. However, we will select one representative of those models as a benchmark for the other selected models. Information-theoretic models inherently only allow to indirectly compare the outlierness of different data instances which makes the application difficult and also computationally expensive. Aggarwal (2015) suggests that other models should be preferred if they can be applied. Consequently, we constrain the selection of suitable outlier detection models to the groups of clustering as well as distance-based, probabilistic and extreme values models. The individual models will be evaluated in detail in the next section.

10.4.2 Step 2: Model Robustness

The second step in the proposed approach is the identification of a robust machine learning model. It is desirable that the chosen prediction algorithm is robust to changes in the input data and keeps the same prediction performance over time (R1). This can be interpreted as a second safety measure in addition to the outlier detection. It is crucial that the machine learning model is robust and issues ongoing reliable predictions. For the selection of the prediction models, we choose the available prediction model out of the following groups in scikit-learn (Pedregosa et al., 2011): Generalized Linear Models, Kernel Ridge Regression, Support Vector Regression, Stochastic Gradient Descent, Neural Network Regression, Ensemble methods.

10.5 Evaluation

We evaluate the overall prediction method with data from a large global automotive supplier. The objective is to optimize the performance of an integrated engine part with a small computing unit within a vehicle. It is necessary to measure the flow quantity of a liquid [mm^3] as exactly as possible for an overall good driving performance of the engine. The estimation is crucial to optimally align the behaviour of other components in the vehicle. However, from a technical perspective, it is difficult to install a sensor in the affected engine component for

the direct measurement of the liquid's quantity. Furthermore, such a sensor is very costly and economically not feasible. Therefore, surrounding sensors in other parts of the vehicle are used as input to estimate the flow quantity. These consist of a high frequency pressure sensor as well as a sensor in a feed pump which measures the number of revolutions per minute. We apply a supervised machine learning approach in order to predict the flow quantity.

After the engine component is built into the vehicle during production, it is impossible to measure the flow quantity (no ground truth labels). However, flow quantity measurements can be derived on a specialized test bench which is used for research and development activities. This gives us the opportunity to train and evaluate the proposed prediction method. The test bench allows to simulate the real-life usage of the corresponding vehicle. The final prediction method needs to exhibit good performance but also at the same time high robustness to outliers. Furthermore, model updates can only be performed when the vehicle is brought to the garage for maintenance work. Therefore, a robust prediction model which does not require regular adaptations needs to be implemented.

Restrictions with regard to the available infrastructure for computing the predictions within the vehicle also need to be taken into account. Due to space limitations as well as shock resistance requirements, the built-in computing unit is rather small. This leads to two additional challenges: First, the computing power of the computing unit is limited. Therefore, the computation of the prediction itself should not be very complex since otherwise the computing unit will take too long to issue a prediction. The same limitation applies to the outlier detection. However, this does not restrict the initial training of the model which can be performed outside of the local computing unit, e.g., on a large cluster. Second, there is only a limited amount of storage space available within the computing unit. This restricts not only the size of the prediction model but also leads to limitations for the outlier detection, e.g., it is impossible to store a large data set of acceptable data instances on the computing unit.

The data provided by the company is split into two data sets. Data set 1 consists of data instances which are the result of a systematic testing of different operation points which are occurring in real driving behavior of the vehicle. This data set is therefore used as training and validation set in order to optimize the chosen prediction method. In total, it consists of 9,992 data instances. Data set 2 consists of a random sample of data instances with different operating modes and contains 13,354 data instances in total. This is used as a test set to examine the performance of the chosen prediction method.

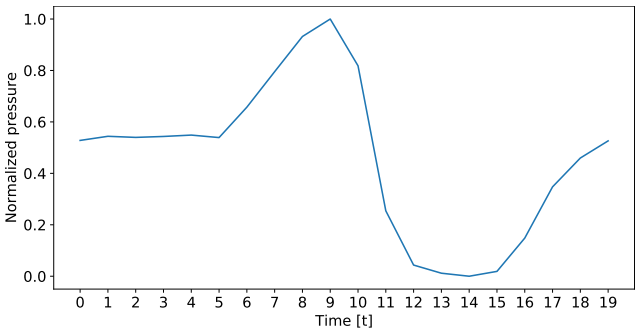


Fig. 10.2.: Normalized pressure trajectory.

Each data instance has 21 variables. One variable is the number of revolutions of the feed pump. Additionally, there are 20 variables which are related to the trajectory of the pressure values. A normalized pressure trajectory is depicted in Figure 10.2. Every pressure trajectory is related to exactly one quantity flow which is the prediction target.

10.5.1 Evaluation of Data Validity (Step 1)

The first step in the proposed prediction method is a dedicated outlier detection approach. As explained in Section 10.4, we focus on the following classes of outlier detection algorithms: Extreme values, clustering, distance-based and probabilistic models. All outlier detection models require the definition of an outlier threshold in order to identify outliers. In this work, the thresholds are derived from the behavior of the outlier detection models on the training set. Thresholds are defined by considering the 95%-quantile of the outlier distance measure on the training set instances. This way, we also define some of the data instances in the training set as outliers. We do not have any information regarding the real outlierness of the data instances in the training set but we somehow need to define a threshold to identify outliers in operation of the general prediction approach. Due to the potentially critical impact of outliers on the behavior of the prediction model, we prefer to falsely classify a few normal data instances as outliers compared to the probability of missing real outliers.

For the *extreme values model*, we compute the data distribution over all data instances in the training set for each input feature respectively. Subsequently, we derive the thresholds for each input feature based on the training data. However, we need to adjust the quantile value q for each input feature since an application of $q = 0.95$ leads to overly strict thresholds as $0.95^{21} = 0.341$. Therefore, the quantile for

each input feature is updated to $q = e^{\frac{log_e 0.95}{21}} = 0.997$. This value of q is evenly distributed to low and high values for each input feature. All data instances which have a measurement which is either higher or lower than one of the thresholds are considered outliers.

From the *clustering models*, we apply the k-means algorithms with euclidean distance on the training data in order to identify suitable clusters (Kanungo et al., 2002). Parameter k is determined by applying the elbow criteria which leads to the selection of $k = 3$. We then use this parameter to apply a clustering on the training data with three clusters. Subsequently, for each of the clusters, the distance to the corresponding cluster centroid for every data instance in the training set is computed. We order these distances by size and determine the 95%-quantile of these distances per cluster as a threshold.

K-Nearest-Neighbors (KNN) (Aggarwal, 2015) is chosen as representative from the *distance-based models* since it is the most common used algorithm from this group. We choose the value of k based on the size of the training set N. Therefore, k is computed as follows: $k = \sqrt{N} = \sqrt{9992} \approx 100$. For each data instance in the training set, the euclidean distance to its k^{th} neighbor is computed. Those distances are ordered by size and the corresponding 95%-quantile is derived as threshold.

With regard to the *probabilistic models*, we compute the Mahalanobis Distance (MD) (De Maesschalck et al., 2000) which is basically defined by the covariance matrix of the entire training data set. The MD between a data instance and the mean value of all data instances can be understood as a metric for the outlierness of a data instance. In comparison to the Euclidean Distance, the MD also takes into account the covariance structure of the data and thereby normalizes the influence of each input feature on the overall distance computation. If a specific threshold for the MD is fixed in a two-dimensional space, we basically determine an ellipse around normal data instances which allows to differentiate outliers from non-outliers. The threshold value is based on the 95% - quantile of the MD in the training set. We call this approach confidence ellipse in the following.

As a fifth outlier detection method, we combine clustering with probabilistic models. Instead of fitting just one ellipse around the data instances in the training set, a k-means clustering with $k = 3$ is performed. Afterwards, an ellipse is estimated around each of the identified clusters. The threshold is computed based on the training data. This approach is called clustering and confidence ellipses in the following.

To evaluate the five outlier detection methods, we have defined two distinct evaluation criteria: The first criterion is the amount of memory on the computing unit which is necessary to enable the outlier detection method (R2). We approximate the memory requirements by deriving how much *Floating Point Numbers (FPN)* need to stored. The second criterion is the necessary computation time in operation (R3). This refers to the calculations which need to be carried out in order to determine the outlierness of a new data point. For the remainder of this work, we differentiate into *Simple Operations (SO)* and *Complex Operations (CO)*. Simple operations refer to mathematical operations such as multiplications, additions or root calculations. On the contrary, complex operations mean the computation of more sophisticated functions such as $tanh$ or $logistic$. The first two criteria refer to the limited computing resources in our use case.

Unfortunately, it is impossible to evaluate the general performance of the various outlier detection models at this point. In general, evaluation of outlier models are based on some kind of ground truth. However, neither the company nor we possess any ground truth labels with regard to the outlier detection and it is impossible to automatically derive the labels from available data. One solution is the manual labelling by a domain expert, however in our case with over 23,000 data instances, this approach seems unfeasible. Apart from this approach, there exist no general evaluation approach for outlier detection (Zimek et al., 2012). Therefore, we decide to evaluate the performance of the outlier detection models by comparing the overall prediction performance of various combinations of outlier detction and prediction models. This is reflected in the results of section 10.5.3.

Tab. 10.1.: Evaluation of different outlier detection approaches.

Requirements / Outlier detection	Memory	Computation time in operation
Extreme values	~ 42 FPN	SO: ~ 80 CO: 0
Clustering	~ 50 – 200 FPN	SO: ~ 100 – 300 CO: 0
K-nearest neighbors	~ 200.000 FPN	SO: ~ 200.000 – 500.000 CO: 0
Confidence ellipse	~ 400 FPN	SO: ~ 1.000 CO: 0
Clustering & confidence ellipses	~ 1000 – 4000 FPN	SO: ~ 1.100 – 1.300 CO: 0

The preliminary evaluation of the chosen outlier detection approaches is depicted in Table 10.1. The first row shows the evaluation of the extreme values method (Aggarwal, 2015). With regard to the storage space requirement, it is necessary to save exactly two values per input feature (one lower and one upper threshold). This results in $2 * 21 = 42\ FPN$. The computation time in operation is determined by comparing 21 values of a newly arriving data instance with the given maximum and minimum thresholds resulting in 42 boolean expressions. Subsequently, the results of each of these 42 boolean expressions need to be evaluated which results in 84 simple operations in total. For both, memory as well as computational requirements, we have indicated approximate values since small changes in the parameter space also lead to changes in these numbers. The memory requirements for kNN are depicted in third row of the table. Since it is necessary to store the entire training set in memory, we need to have enough storage space for $9,992 * 21 = 209,832\ FPN$. Regarding the computation time, for every data instance, it is necessary to compute one subtraction as well as one square computation per input feature and one square root operation per data instance: $9,992 * (21 * 2 + 1) \approx 430,000$.

The difference in memory requirements between the outlier detection methods and kNN is significant. Even the most complex method (clustering with confidence ellipses) requires only around one percent of the storage space of kNN. Among the four remaining outlier detection methods, confidence ellipse as well as clustering with confidence ellipses require the most memory and computation time.

Furthermore, we perform a sensitivity analysis of the outlier detection approach with the creation of artificial outliers. Several types of artificial outliers are created by shifting the minimum and the maximum value of the pressure trajectory as well as by adding random noise and by introducing a phase shift. An example for an artificial outlier with phase shift is depicted in Figure 10.3. This was evaluated manually on single data instances in order to further assess the functionality of the outlier detection model.

The left figure shows an original pressure trajectory (in blue) and one which is shifted by six time units (in orange). The right figure shows the corresponding development of the Mahalanobis distance (MD) with the amount of phase shift on the x-axis. The red horizontal line marks the outlier threshold for the MD. The MD rises significantly with the introduced phase shift. It is interesting to note that the MD decreases again with a phase shift of 18 or 19. The shifted pressure trajectory at this point is very similar again to the original trajectory since a phase shift of 20 refers to the exact same trajectory.

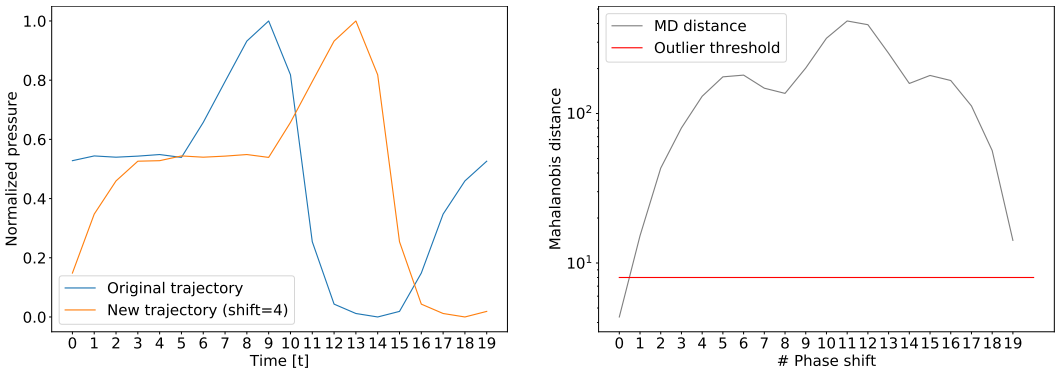


Fig. 10.3.: Sensitivity of the Mahalanobis distance regarding a phase shift.

10.5.2 Evaluation of Model Robustness (Step 2)

With the outlier detection of step 1 successfully evaluated, we now regard the evaluation of the prediction model. In order to make a first selection of suitable prediction algorithms, we conducted pretests with the groups of algorithm that we introduced in Section 10.4. For every group, we implemented one prediction model with the standard parameter configuration in scikit-learn (Pedregosa et al., 2011). Due to the large number of available options for the generalized linear models, we implemented two prediction models, namely linear regression and polynomial regression (degree $= 2$). A random train-test split with 30% test data is performed on the training data (data set 1) for the model evaluation. We apply the Mean Absolute Percentage Error (MAPE) as evaluation metric because it is the common metric for engineers involved in the use case and provides meaningful and interpretable results (Armstrong & Collopy, 1992). Furthermore, MAPE is independent of scale and therefore comparable among different data sets.

Tab. 10.2.: Results of pretest with various prediction models.

Prediction Model	MAPE
Polynomial Regression	3.07%
Neural Network Regression	6.02%
Support Vector Regression	10.17%
Linear Regression	11.05%
Stochastic Gradient Descent Regression	12.80%
AdaBoost Regression	14.17%
Kernel Ridge Regression	15.67%

The results are depicted in Table 10.2. There are large performance differences between the three best performing prediction models with polynomial regression

outperforming the other models by far. However, Neural Network (NN) and Support Vector Regression (SVR) usually increase their performance significantly after parameter optimization. Therefore, we include polynomial regression, NN and SVR for the next steps of a more detailed investigation.

Subsequently, we perform a grid search (Bergstra & Yoshua, 2012) with a 3-fold-cross-validation (Golub et al., 1979) on the three prediction models. Since the polynomial regression does not possess any parameters except the degree for the polynomial transformation, we also evaluate the lambda parameter for the ridge shrinkage method on the polynomial regression. The grid search is performed on the entire training set (data set 1). Best parameter combinations for each prediction model are shown in Table 10.3.

Tab. 10.3.: Parameter results of grid search.

NN parameters		
Par.	**Range**	**Best**
Layers	(2,), (4,), ... (40,), (10,5,), (20,10), (40,20)	**(32,)**
Activation function	logistic, tanh, relu	**tanh**
Alpha	0.00001, 0.0001, ..., 10, 100	**0.0001**

SVR parameters		
Par.	**Range**	**Best**
Kernel	rbf, poly, linear, sigmoid	**rbf**
C	1, 10, ..., 100.000	**10.000**
Gamma	0.0001, 0.001, ..., 10	**0.1**
Epsilon	2,3,...,7	**3**

Polynomial ridge regression parameters		
Par.	**Range**	**Best**
Lambda	0.00001, 0.00002, ..., 0.01, 0.02, ... 1	**0.002**
Degree	2, 3	**2**

For the evaluation of the different prediction models, several criteria need to be considered. Similar to the evaluation of the outlier detection methods, it is necessary to consider the memory requirements (R2) as well as the computation time in operation (R3). Additionally, prediction performance (R1) is measured by applying the prediction models on the test set (data set 2). Furthermore, the robustness of the prediction model (R1) is an important evaluation criterion. We measure robustness by randomly splitting the test set ($N = 13,354$) in 10 batches of equal size and by computing the prediction performance on each of those batches. Subsequently, the range of the MAPE values between the different batches is assessed.

For each of the prediction model classes, we implement two instantiations with different parameter settings. One instantiation refers to a simple model (fewer parameters), whereas the other instantiation represents are more complex model (more parameters). This approach allows to test the sensitivity between model simplicity and prediction performance (Adler & Clark, 1991). As an example, we implement a simple neural network with 4 neurons in the hidden layer and a more complex one with 32 neurons which was the best number of neurons during grid search. In total, $3 \cdot 2 = 6$ different prediction models are evaluated. Similar to the evaluation of the outlier detection models, we give approximate values for the

memory and computation time requirements since those values heavily depend on the parameter choice. However, the values in the table provide a pretty good indication of the differences between various prediction models.

Tab. 10.4.: Evaluation of different prediction models.

Requirements / Prediction model	Memory	Computation time in operation	Prediction performance (MAPE)	Robustness (10 folds)
NN (1 hidden with n_1 neurons)	~ 100 – 400 FPN	SO: ~ 200 – 800 CO: ~ 4 – 20	4.51%	[4.23% - 4.77%]
NN (1 hidden with n_2 neurons)	~ 300 – 1.500 FPN	SO: ~ 600 – 3.000 CO: ~ 15 – 60	5.05%	[4.51% - 6.07%]
Simple SVR (rbf-kernel)	~ 2.000 – 6.000 FPN	SO: ~ 4.000 – 12.000 CO: ~ 100 – 300	3.54%	[3.36% - 3.78%]
Complex SVR (rbf-kernel)	~ 10.000 – 40.000 FPN	SO: ~ 20.000 – 80.000 CO: ~ 500 – 2.000	3.67%	[3.47% - 4.00%]
Polynomial regression with ridge (deg = 2)	~ 50 – 150 FPN	SO: ~ 150 – 450 CO: 0	6.00%	[5.30% - 7.03%]
Polynomial regression (deg = 2)	~ 250 FPN	SO: ~ 750 CO: 0	5.70%	[4.90% - 6.37%]

The results are depicted in table Table 10.4. The simple neural network consists of only 4 neurons in the hidden layer and the input vector has 21 features. Therefore, we need to save $21 \cdot 4 = 84$ weights and $1 \cdot 4 = 4$ biases between the input and the hidden layer as well as 4 weights and 1 bias between hidden layer and the output layer. With regard to the computation time, it is necessary to perform one multiplication and one addition per weight as well as one addition per bias leading to $((21 + 1) \cdot 4) \cdot 2 + (4 + 1) \cdot 2 = 186\ SO$. In each of the neurons, one $tanh$ needs to be evaluated resulting in 4 CO. The MAPE of 4.51% is indicated in the third column and the range of the MAPE values among the different batches is shown in the last column. The differences in memory requirements and computation time between the prediction models are remarkable. Compared to the simple neural network, the complex SVR is represented by 1.448 support vectors which results in $1.448 * 22 + 1 = 31.857\ FPN$ to store. With regard to the computational complexity, due to the chosen rbf-kernel, it is necessary to compute 1.448 exponential functions (CO) for every new data instance.

Compared to the prediction results in the pretest (c.f. Table 10.2 on data set 1 on page 194), the prediction performance of the different prediction models seems to be discouraging. This is especially true since a parameter optimization has been carried out. Presumably, the test data (data set 2) which is used to compute the MAPE values is more diverse and complex to predict than the training set. This impression is also confirmed by the company which describes data set 2 as randomly sampled whereas instances for data set 1 were systematically selected.

Considering the performance differences between groups of algorithms, SVR now clearly outperforms its counterparts whereas Polynomial Regression which was the best prediction model in the pretest performs worst. Probably, this behavior can be explained by the fact that SVR contains more parameters which can be optimized in a parameter optimization. Interestingly, the performance difference between the simple and the complex model per prediction model class are rather small. This indicates that also models with few parameters might be powerful enough to generalize well in this use case. Regarding the robustness, the MAPE range for a simple SVR is only 0.4% which indicates a rather robust prediction model. In contrast, the performance of the polynomial regression fluctuates significantly with 1.7%.

10.5.3 Evaluation of Overall Prediction Method

As we evaluated the two main steps of the proposed method in an isolated way, we now evaluate the performance of the overall approach with outlier detection and a subsequent prediction model on the test set (data set 2). Table 10.5 displays the MAPE values for each prediction model (in rows) in combination with each of the chosen outlier detection models (in columns). The first row in the table indicates the number of outliers which are identified in the test set by each of the models. Clustering with confidence ellipse detects only 322 outliers, whereas kNN flags nearly 1,300 instances as outliers. It is important to keep those numbers in mind for the overall evaluation of different combinations. An approach which just marks nearly all data instances as outliers will not be applied in practice.

Each prediction model in the table is described by three rows. The first row depicts the prediction performance (MAPE) when no outliers are removed from the test data. The second row refers to the prediction performance when the outliers identified by the detection models are removed from the data set. The third row is introduced to illustrate the effectiveness of the outlier detection models: In this case, we randomly remove the same amount of data instances as indicated by the outlier detection models and compute the MAPE based on the remaining data instances. To illustrate this, we will consider the complex Neural Network in combination the extreme values approach. The cell “All" is computed based on all $13,354$ data instances, the cell “OutlierDet" is based on $13,354 - 877 = 12,477$ data instances and the cell “Random" is also based on $12,477$ instances but which are randomly selected from the entire test set.

Tab. 10.5.: Performance of outlier detection in combination with prediction model (MAPE).

		Extreme values	Clustering	kNN	Ellipse	Clustering & ellipses
	# outliers	877	409	1294	330	322
Simple NN	**All**	0.0451	0.0451	0.0451	0.0451	0.0451
	OutlierDet	0.0375	0.0393	0.0419	0.0394	0.0393
	Random	0.0447	0.0449	0.0450	0.0449	0.0449
Complex NN	**All**	**0.0505**	0.0505	0.0505	0.0505	0.0505
	OutlierDet	**0.0306**	0.0312	0.0336	0.0313	0.0312
	Random	**0.0503**	0.0506	0.0510	0.0506	0.0506
Simple SVR	**All**	0.0354	0.0354	0.0354	0.0354	0.0354
	OutlierDet	0.0315	0.0323	0.0346	0.0324	0.0321
	Random	0.0352	0.0353	0.0354	0.0354	0.0354
Complex SVR	**All**	0.0367	0.0367	0.0367	0.0367	0.0367
	OutlierDet	0.0309	0.0312	0.0335	0.0313	0.0312
	Random	0.0366	0.0368	0.0369	0.0368	0.0368
Ridge Reg	**All**	0.0600	0.0600	0.0600	0.0600	0.0600
	OutlierDet	0.0336	0.0352	0.0378	0.0353	0.0352
	Random	0.0592	0.0593	0.0598	0.0593	0.0593
PolyReg	**All**	0.0570	0.0570	0.0570	0.0570	0.0570
	OutlierDet	0.0344	0.0359	0.0383	0.0361	0.0360
	Random	0.0567	0.0570	0.0576	0.0569	0.0569

Considering the overall performance, all models improve their performance significantly when the outlier detection is introduced, e.g., the MAPE of the complex neural network improves from 5.05% to 3.06% in combination with the extreme values approach. As expected, the MAPE for randomly selected data instances remains on a similar level (5.03%).

Table 10.6 shows the 5 combinations of outlier detection and prediction model with the lowest MAPE. Extreme values and clustering with confidence ellipses seem to be promising approaches for the outlier detection. Complex neural network and complex support vector regression perform best among all prediction models. With all this information at hand (Table 10.1, Table 10.4, Table 10.5), an informed choice about the best possible combination can be made. In case decision-makers focus on the lowest overall MAPE, extreme values in combination with complex neural network should be chosen. In case, it is important that not too many data instances are considered outliers, clustering with confidence ellipses with for instance complex support vector regression should be considered. Additionally, one can also consider

the memory as well as computation requirements of each combination for the decision making process.

Tab. 10.6.: 5 combinations with the lowest MAPE.

Rank	Outlier detection model	Prediction model	MAPE
1.	Extreme values	Complex NN	0.0306
2.	Extreme values	Complex SVR	0.0309
3.	Clustering	Complex NN	0.0312
4.	Clustering & ellipses	Complex SVR	0.0312
5.	Clustering & ellipses	Complex NN	0.0312

Figure 10.4 shows the distribution of MAPE (y-axis) with the liquid quantity (target) on the x axis based on the computation of complex neural network with clustering and confidence ellipses. Data instances which are marked in red are detected as outliers, whereas data instances marked in blue are considered normal data. The left plot shows that data instances with a high error are also detected as outliers. We can exactly identify those data instances as outliers which also lead to high errors during application of the prediction model. This clearly underlines the performance of our prediction method. The right plot zooms in on the middle of the left plot. The algorithm detects some outliers which have a faulty prediction, however there also seem to be some normal data instances (with a low error) which are detected as outliers. These data instances are especially located at the far right side of the plot.

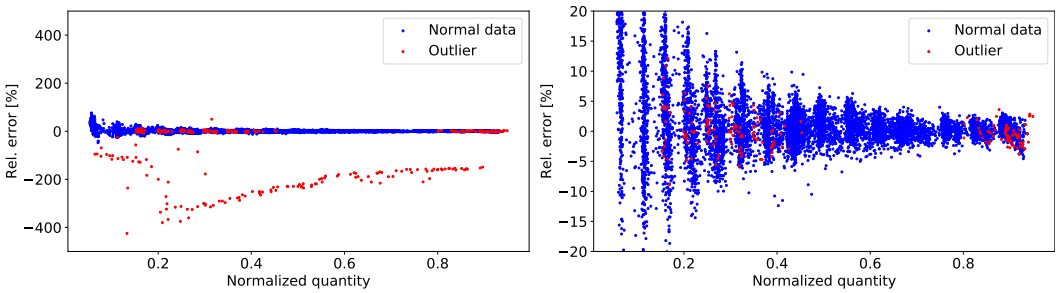


Fig. 10.4.: Error plot with complex neural network.

In total, the outlier detection method only detects 2.4% ($322/13354 = 0.024$) of data instances as outliers in the test set. The prediction model will not issue a prediction in this case because this data instance has been marked as outlier beforehand. Other parts of the vehicle can therefore not adapt their behavior because they do not receive the necessary feedback. In this case, the system has to rely on previous predictions and adjust accordingly. This solution is certainly not optimal, but definitely better

than adapting on a false prediction which might lead to devastating results. Usually, sudden changes from one measurement to another are not observed in this system and therefore change happens incrementally. This also allows to skip some required predictions in between.

However, in case of real concept drift in the system which results in a persistent new concept over time, the proposed prediction method will not be of use. Due to the missing label feedback, the system will not be able to adapt to the new concept and will therefore continuously classify new data instances as outliers. We can monitor the percentage of outliers compared to all incoming data instances as a metric for the sanity of the system. If an increase in the percentage of detected outliers is observed, the driver will be notified with a request to bring the vehicle to the garage where model updates can be performed.

10.6 Conclusion

This work introduces a method for designing robust machine learning models in local environments without label feedback. Traditional concept drift adaption mechanisms (Gama et al., 2014) cannot be applied in this case because they rely on feedback from true labels. We propose a prediction method (machine learning service) combining a dedicated outlier detection with a subsequent prediction model. Predictions are only issued for data instances similar to the training data. Dissimilar data instances are marked as outliers and no prediction is computed in this case reducing the probability of receiving erroneous predictions. Infrastructure resources such as limited computing power and storage space are also considered. We evaluate our approach by applying it on a data set from a large global automotive supplier. We show that the prediction performance based on our prediction approach is increased significantly compared to an approach without outlier detection and at the same time available memory and computation resources are sparsely applied.

Our contribution is twofold: First, we introduce a prediction method which combines an outlier detection in combination with a robust prediction model to ensure the proper functionality in local areas of application without label feedback. Second, we develop a set of requirements for the evaluation of outlier detection and prediction models for similar settings. For each requirement, we derive and develop corresponding metrics and then apply these methods on our use case to demonstrate the feasibility of the method.

Regarding the managerial implications, this work can be used as a guideline for practitioners on how they can implement machine learning solutions in settings with similar constraints. The developed method allows to assess and evaluate a variety of different requirements relevant for prediction approaches implemented in practice. Thereby, this work contributes to ensuring the correctness of machine learning results in industry settings.

Our approach certainly has limitations. The prediction method discards data instances classified as outliers and does not provide a prediction for those. In case of many subsequent outliers, the system will therefore not receive any prediction and will not be able to adapt appropriately. Additionally, we evaluated the approach only with one data set. Even though this data set represents a realistic driving scenario, it is still created on a test bench. Therefore, we cannot evaluate the prediction method during unexpected situations that can arise in reality. Furthermore, it is difficult to assess the performance of the outlier detection method since there are no true labels with regard to the outlierness of data instances. In future work, we aim at performing a field study with a vehicle participating in normal road traffic. Additionally, the proposed approach should be evaluated in a different context in order to prove its effectiveness.

The prediction method presented in this work can help to increase the acceptance of machine learning in real-world contexts with the aim to effectively deploy models in productive environments. So far, many companies are still skeptical about taking automated decisions based on deployed machine learning solutions. This is due to a fear of extreme decisions based on unusual input data. This work shows how existing methods can be combined in order to increase the reliability of machine learning solutions and therefore aims at increasing the practical relevance of this powerful technique.

Part VI

Finale

11 Conclusion

The application and deployment of machine learning within information systems offers huge potential for innovative products and services. Therefore, this thesis analyzes the choices and steps required for introducing machine learning in an IS context as well as the associated risks and challenges. In particular, the deployment of such solutions still remains a difficult endeavor. In this context, especially changing data—also called *concept drift*—and its impact on prediction quality is a significant challenge preventing a more widespread use of this technology. Therefore, this thesis aims to develop novel concept drift handling strategies for information systems relying on machine learning. Overall, the thesis especially focuses on the challenges of information systems in their real-world environments. We address this research objective by answering five interrelated research questions.

11.1 Summary and Contributions

An increasing share of information systems applies machine learning to enhance their functionality. Thereby, information systems can help in many different contexts such as improving internal processes or creating entire new service offerings. Despite the growing capabilities of machine learning, its introduction is also associated with a set of challenges possibly affecting the benefits of information systems. To mitigate these challenges, this thesis contributes on two levels: First, on a more general level, we provide knowledge for the setup phase of such systems by describing necessary choices for the application of machine learning as well as related challenges during deployment. Furthermore, we also introduce a framework characterizing different options for implementing concept drift handling solutions. Second, on a more detailed level, we provide methods for specific problems in handling concept drift.

The D&M information systems success model (Delone & McLean, 2003) specifies various factors influencing the net benefits of information systems. In the context of machine learning-based information systems, especially *information quality* and *system quality* are important factors. This is due to the probabilistic nature of machine learning where predictions and recommendations are often only correct to

a certain extent. *Information quality* can be measured by considering the accuracy and relevance of the recommendations or decisions provided by the information system. In the context of machine learning-based information systems, these metrics can be directly associated with the prediction quality of the underlying machine learning model. Concept drift—if not addressed properly—can have a severe impact on the accuracy of machine learning predictions as shown in many examples in this thesis. Furthermore, *system quality* is an important success factor and can be determined in terms of reliability and functionality of an information system. By implementing appropriate solutions such as the methods introduced in Chapter 7-10, users can increase the robustness and reliability of their information systems continuously.

The contributions in this thesis can also be considered from a life cycle perspective on machine learning-based information systems (Duarte & Costa, 2012). First, the implementation process of such systems can be improved by taking the introduced choices and challenges related to machine learning into account. Second, especially maintenance activities to ensure the proper functionality over time can be facilitated. By implementing appropriate detection methods (e.g., Uncertainty Drift Detection), it is easier to identify the need for maintenance in specific parts of the information system. When also implementing appropriate adaptation strategies (e.g., Error Intersection Approach), machine learning-based information systems will require less maintenance overall due to their self-adapting capabilities.

Within this thesis, we address five research questions with different research methods ranging from literature reviews to interview studies combined with a qualitative content analysis as well as a set of technical experiments. In the following, we summarize, synthesize and discuss the contributions according to the research questions introduced in Chapter 1, starting with the first research question.

Research Question 1 (RQ1)
Which choices are relevant in the application of supervised machine learning in IS research?

We address the first research question by developing the supervised machine learning reportcard which allows to comprehensively document a machine learning endeavor in IS research. The reportcard is divided into three phases: model initiation, performance estimation and model deployment. By applying the reportcard to a set of high-quality IS articles, we reveal that current IS research documentation can be improved. For instance, only a few research articles justify the chosen performance

metric and compare the performance of their models with appropriate benchmarks. The analyzed articles also provide little evidence regarding the choices during the model deployment phase. Various reasons can be considered to explain this phenomenon. In some cases, researchers might only want to prove the feasibility of an approach utilizing supervised machine learning. If a project focuses on this, it is not necessary to perform and therefore also describe related deployment activities. Nevertheless, only a minority of all papers at least describe ideas about a possible model deployment and corresponding challenges such as concept drift. This observation stands in contrast to the objective of IS to produce final, implementable results with large implications for practice (Gholami et al., 2016). Therefore, it is also possible that knowledge regarding deployment and the challenges associated with it is limited. The detailed research results are presented in Chapter 3.

The application of machine learning in IS is complex and requires a multitude of choices to be made. However, it seems that individual choices are associated with different levels of difficulty with the deployment step being especially complex. Therefore, we aim to better understand the necessary choices in the model deployment phase and derive the prevailing challenges during this phase in practical settings.

Research Question 2 (RQ2)
Which challenges arise regarding the deployment of supervised machine learning models?

We explore and answer the second research question from a practice perspective in Chapter 4. To this end, we perform semi-structured interviews with 11 machine learning practitioners from different domains. Based on our analysis, we can identify six overarching clusters of challenges: *data structure*, *implementation*, *infrastructure*, *governance*, *customer relation* and *economic implications*. Regarding deployment, concept drift and data drift as well as the speed of new incoming instances are the main challenges. Furthermore, the implementation of ongoing data and results validation methods is difficult. Lastly, there is no standardized deployment infrastructure available, which often prevents scalability of deployed solutions.

The results show that there exist also a number of other challenges hindering more widespread use of machine learning technology. The results reveal a wide set of non-technical challenges such as appropriate expectation management as well as results and customer communication. Furthermore, difficult data access and data management processes can additionally complicate deployment projects.

The results derived in Chapter 4 indicate that deploying machine learning models is still a difficult process. In this context, data changes over time are one of the main challenges since they can have a large impact on the correct operation of deployed models. Therefore, we decide to better understand and analyze the impact of concept drift on machine learning models as well as to investigate possible solutions for this problem.

> **Research Question 3 (RQ3)**
> What are typical challenges for the application of concept drift handling algorithms?

For answering RQ3, we apply different research methods—a technical experiment and a literature review—which allows us to derive a broad overview on concept drift handling implementation challenges. In Chapter 5, we report on a real-world process mining use case. Specifically, we use machine learning techniques to improve a procurement process by predicting the product delivery time, which allows for optimizing production processes and, therefore, is associated with substantial cost savings. This use case depicts the challenge of choosing the appropriate concept drift handling strategy, as different strategies lead to different improvements in prediction performance. Furthermore, this chapter also illustrates the difficulties of deciding on the right number of data instances as well as which specific data instances to select for retraining of the prediction model. Therefore, we propose and evaluate three different selection strategies.

Additionally, this chapter provides more information about the nature of real-world concept drifts. Our analysis shows that over the years, more and more process steps of the procurement process have been automated (e.g., automated booking for incoming products). This, in turn, decreases the average delivery time and thereby might explain the decreasing prediction performance over time. However, due to the IT structure within the company, the information about increasing automation cannot be included in the prediction model. This is a very illustrative example of a phenomenon called *hidden context* in concept drift research. It refers to a feature which is strongly influencing the prediction quality but is impossible to include as an input feature.

While Chapter 5 depicts challenges of concept drift handling from a technical perspective, the subsequent Chapter 6 investigates the problem from a more conceptual perspective. Based on literature, we derive a framework describing different choices during the implementation and operation of a machine learning service confronted

with concept drift. The different categories can be divided into three parts: 1) setup decisions, 2) algorithmic decisions, and 3) decisions during operation of machine learning services. Based on our framework, we can show that during the setup of such a service, different design characteristics and choices need to be considered, e.g., what type of concept drift to expect. Furthermore, it is crucial to respect the characteristics during operation. Depending on the use case, true labels might not be available for concept drift detection or adaptation.

After its creation, we utilize the framework to classify existing solutions from literature and create a heatmap illustrating different research approaches. This analysis reveals several challenging areas with little research so far: Only few concept drift approaches for unstructured data have been introduced even though applications with data like images or text gain more importance. Furthermore, we identify a deficient amount of use cases working with regression tasks or analyzing the problem of limited availability of true labels. These observations are also confirmed by other research papers in the field. Therefore, our work regarding research questions 4 and 5 provides specific solutions for these topics.

Research Question 4 (RQ4)
How can concept drift in regression problems be handled?

In Chapter 7 and Chapter 8, we introduce two novel methods for detecting and handling concept drift in regression setups with a special focus on time series forecasting. Handling concept drift in regression tasks is special compared to classification as predictions have continuous values and prediction errors are also distributed differently. Therefore, standard drift detection algorithms are often not applicable in this setup.

Chapter 7 introduces the Error Intersection Approach (EIA), a method for handling sudden concept drift in time series forecasting. The general idea of EIA is to switch between a simple and a complex prediction model for drift handling. The complex model is able to accurately capture the general components of a time series and, therefore, produces precise forecasts during periods with regular patterns. However, in times of sudden change, the simple model is applied. It quickly adapts to current changes and is able to generate correct predictions during periods of unusual observations. The simple model is largely influenced by the autocorrelation properties of a time series. Therefore, it is reasonable to assume that EIA performs better if the autocorrelation with recent observations is higher. A switch between the two models is triggered by a comparison of the error rates. If the error curves of the

two models intersect, a change between models is initiated. Therefore, the method always chooses the model which recently showed better prediction performance. In summary, EIA is able to detect drift (the moment when the error curves intersect indicates a drift) but also to adapt to new environments (by switching between the two machine learning models).

We can show that EIA improves prediction performance significantly in a technical experiment for predicting taxi ride demand in New York City. By analyzing days with the highest prediction performance improvements, we can reveal that EIA is especially helpful during days with weather irregularities or on national holidays. However, we also identify other days where it is impossible to derive the drift cause. In that case, more local events such as demonstrations might trigger concept drift. This analysis also illustrates that EIA cannot only be applied for improving prediction performance but also for analyzing past decisions of deployed models. This can reveal which additional variables should be included in the modeling process.

In Chapter 8, we introduce the Switching Scheme as an alternative way for handling concept drift in regression problems. In contrast to the previous method, it focuses on handling slower, incremental concept drifts. To that end, it combines the adaptation techniques of retraining from scratch and incremental updates for concept drift adaptation. In the first phase after deployment, the Switching Scheme relies on incremental updates for adaptation. Thus, the resulting model has access to recent data instances during the update step as well as the initial training data during the first training. Eventually, this model has access to an overall larger data set for training compared to a single retraining, which increases its ability to generalize. However, after a certain period of time, incremental updates are no longer sufficient for adapting to novel concepts. In this case, the Switching Scheme changes its adaptation method to retraining from scratch, which allows a stronger adaptation to the current concepts.

The Switching Scheme is evaluated in a technical experiment on a data set regarding taxi ride demand in New York City and allows for significant performance improvements. Furthermore, we also test its ability to generalize on classification problems by applying it on a flight records data set for predicting flight delays.

Research Question 5 (RQ5)
How can concept drift be handled in machine learning settings with limited availability of true labels?

Lastly, in Chapter 9 and Chapter 10, we introduce novel methods for handling concept drift in deployment settings where no true labels are available for predictions issued during operation. Many products or services that organizations want to enhance by applying machine learning share this property of limited label availability. A good example is quality control, where a company—after quality inspection by a machine learning model—does not want to control every single part with a human expert. However, most concept drift handling algorithms require true labels to work properly. Therefore, novel algorithms are required for this special case.

In this context, Chapter 9 introduces Uncertainty Drift Detection (UDD), a novel approach for detecting concept drift without the necessity to acquire true labels. To this end, we use the uncertainty of a prediction model for drift detection. We can show that UDD outperforms input data-based drift detection because those methods detect any change in the input, irrespective of its effect on the machine learning model. In contrast, UDD only detects drift when the machine learning model is also affected by the data changes. For UDD, we apply a neural network for computing predictions and derive uncertainty based on Monte Carlo Dropout. In a technical experiment, we first evaluate UDD on two synthetic data sets where we can clearly demonstrate its ability to differentiate between virtual and real concept drifts. Second, we show the superior prediction performance of UDD compared to relevant drift detection benchmarks based on a set of 8 classification and 2 regression data sets.

While UDD follows the traditional approach of first detecting drift and then adapting to it, Chapter 10 introduces the Two-Step Prediction Method which takes a different perspective. Its objective is to provide a robust machine learning model for challenging environments. *Challenging* refers to the fact that no ground truth labels are available, neither for concept drift detection nor for adaptation. Furthermore, the Two-Step Prediction Method reduces the amount of required computational resources. This is often necessary for situations where computations need to be performed on a local computing unit. This limitation not only refers to the amount of computing power but also to the amount of memory storage available.

Due to the lack of true labels, an adaptation of the prediction model is not possible. Therefore, we shift the objective of the Two-Step Prediction Method from concept drift adaptation to preventing false predictions. To this end, we develop an approach consisting of an outlier detection as well as a robust prediction method. The general idea is that the outlier detection serves as a filter which prevents that possibly wrong predictions are computed for unusual data instances. The Two-Step Prediction Method is evaluated in a technical experiment with industry data from a

large German OEM. Based on this method, we can demonstrate an increased prediction performance while at the same time significantly reducing the computational requirements.

This thesis describes various challenges for the application of machine learning-based information systems with a special focus on the problem of changing data over time. One limitation of current concept drift algorithms is the lack of evaluation with real-word data. Therefore, a strong focus of this thesis lies on the evaluation of the introduced methods with real-world data sets. We perform a variety of experiments based on publicly available real-world data sets such as predicting taxi ride demand in New York City, predicting delays of flights, detecting network intruders, and predicting sensor values. Furthermore, we also utilize proprietary data sets where we improve engine operations in a vehicle as well as optimize procurement processes. By applying these methods to such a wide set of applications, we show their feasibility and also illustrate the possible performance improvement.

The four introduced methods answering research question 4 and 5 (Chapter 7-10) can be differentiated based on their characteristics regarding concept drift handling. As described in Section 2.3.2, concept drift handling can be divided into two steps: 1) concept drift detection and 2) concept drift adaptation. Therefore, we classify the introduced methods regarding those two steps. The resulting overview is presented in Table 11.1, which reveals that we both contribute to concept drift detection as well as adaptation.[1]

Tab. 11.1.: Classification of different methods regarding detection and adaptation.

	Concept drift detection	**Concept drift adaptation**
Error Intersection Approach	Detection by monitoring the intersection of error curves	Adaptation by switching between simple and complex model
Switching Scheme	*N/A* (detection with established drift detection methods)	Switching between incremental updates and retraining from scratch
Uncertainty Drift Detection	Detection by applying ADWIN on neural network uncertainty	*N/A* (adaptation by retraining)
Two-Step Prediction Method	*N/A* (detection possible by monitoring the frequency of outliers)	*N/A* (no adaptation)

[1] If the contribution of a method does not cover one of the two steps, we mark the respective cell with *N/A*.

The Error Intersection Approach can be considered as both a detection approach as well as an adaptation method. The Switching Scheme provides a novel way for the adaptation of prediction models in case of concept drift. For the detection of concept drift, however, it relies on existing methods. In contrast, Uncertainty Drift Detection introduces a novel way for drift detection and relies on established methods for adaptation. Due to the different nature of the objective, the Two-Step Prediction Method can only be partially classified in Table 11.1. Drift detection is possible by considering the frequency of detected outliers. However, we could not evaluate this capability in our experiment. Furthermore, concept drift adaptation is impossible due to the absence of true labels.

Regarding the positioning of our research in the overall field of concept drift research, this thesis deals with two important areas where current research still lacks sufficiently performing solutions: First, handling concept drift in regression problems and second, concept drift handling with limited availability of true labels during operation. Consequently, our contributions can also be categorized based on the two dimensions, label availability and machine learning task. This categorization is depicted in Figure 11.1.

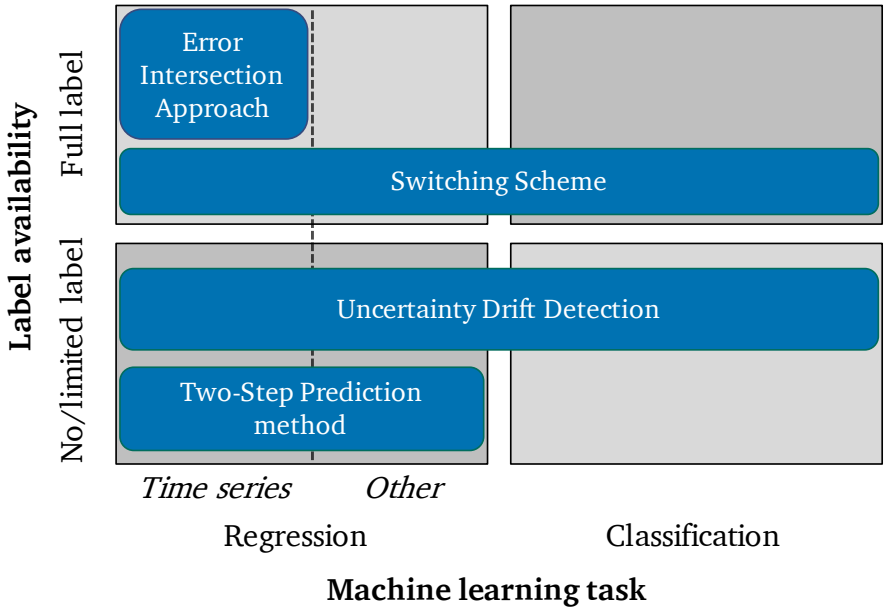


Fig. 11.1.: Overview of methodological concept drift contributions.

The *Error Intersection Approach* requires full availability of true labels for computing the error rate of the utilized machine learning models. For improvement of prediction performance, the Error Intersection Approach exploits the autocorrelation property of time series problems. The *Switching Scheme* also requires true labels during

operation since it is built on established drift detection algorithms. However, it is applicable in a broader context because it combines the more fundamental principles of incremental updates and retraining which can be performed for both classification and regression tasks.

In contrast to the previous two methods, the *Uncertainty Drift Detection* works with a limited amount of true labels since this information is not required for drift detection. Only in case of retraining, a small share of true labels needs to be provided. Since prediction uncertainty can be computed for both classification and regression, this method is also applicable to both tasks. Lastly, the *Two-Step Prediction Method* does not require any true labels at all during deployment. Based on the optimized underlying machine learning models, it is suitable for regression tasks only.

11.2 Practical Implications

In the following section, we reflect on the practical implications of our research. Many companies and organizations consider machine learning as a promising new technology to gain competitive advantages. This observation has triggered countless implementation projects to test the capabilities of machine learning in various use cases. However, only a minority of these projects has been planned carefully and fulfills the necessary requirements for an appropriate deployment of the resulting machine learning models (Alla & Adari, 2021). This thesis contains a multitude of examples showing how a "static" deployed model—which is not regularly adapted—completely loses its validity over time with significant drops of prediction performance.

In this context, it is crucial to consider the economic value of keeping a good prediction performance over time. In today's web applications, a single deployed machine learning model can optimize the experience for a large set of users. Therefore, a slight performance increase can have far-reaching implications. If a model is deployed for improving millions of purchase transactions, a small increase in prediction performance can already imply an increase in revenue by millions. In other scenarios such as quality control, the replacement of repetitive manual tasks might only be feasible if the machine learning model is able to guarantee a stable prediction performance above a certain threshold. This threshold can also be given externally, e.g., by quality requirements defined in a customer contract. Therefore, appropriate concept drift warning and adaption methods need to be implemented. Lastly, in use cases such as optimizing complex supply chains, various machine learning models

deployed by different providers might react and adapt to each other. Poor-quality predictions by a single model can trigger a chain reaction of model failures with consequences being much more dramatic (e.g., empty shelves in supermarkets) compared to lost additional revenue.

Based on this observation, it is critical that design choices regarding the management of models after deployment in general and concept drift in particular are considered from the beginning of any machine learning project. However, our research in the context of the Supervised Machine Learning Reportcard (Chapter 3) reveals that widely used process models such as CRISP-DM (Wirth & Hipp, 2000) or the Microsoft Team Data Science Process (Microsoft, 2020) provide little guidance for the management of deployed solutions. Yet, we identify a variety of significant challenges that only occur after deployment (e.g., concept drift) based on interviews with machine learning practitioners (Chapter 4). Nowadays, these deployment challenges are often neglected in many implementation projects, which is why most projects never go beyond the prototype phase (Alla & Adari, 2021). Therefore, it seems like a major design flaw of process models structuring machine learning projects to not put more emphasis on the necessary steps after the deployment of a model. Practitioners should carefully consider the identified challenges in Chapter 4 to get a realistic view on required steps after deployment. A few researchers are already working on adapting existing process models for this problem, e.g., Žliobaitė et al. (2016) suggest an adaptation of CRISP-DM with additional steps focusing on monitoring, online evaluation and change detection. However, more research is required to support machine learning users in implementing appropriate solutions. In commercial solutions, related tasks regarding the appropriate management of deployed machine learning models are often described with the term *MLOps* or machine learning operations (Renggli et al., 2021).

By introducing novel methods for handling concept drift, this thesis provides solutions for relevant practical applications such as regression tasks or scenarios where the access to true labels is limited. In general, the methods presented in this thesis can either be directly applied by practitioners or can offer inspiration how to build drift detection strategies for their specific use cases. Interestingly, some of the methods can also be combined to create novel artifacts (e.g., using the Uncertainty Drift Detection for detection and Switching Scheme for adaptation). Irrespective of the actual solution, we want to emphasize one important observation that we made during our research: It is significantly better to have a very simple strategy for concept drift handling in place than to have no adaptation strategy at all. This statement especially applies to less experienced practitioners who find some of the suggested solutions difficult to implement due to the required parameter op-

timizations. The use case of predicting taxi demand is a good illustration of this phenomenon (Chapter 8). With a *simple* strategy such as yearly retraining, prediction performance can already be increased significantly compared to a static, *no adaptation* strategy. Implementing the *sophisticated* Switching Scheme additionally improves the prediction performance. However, the performance difference between *sophisticated* and *simple* strategy is smaller compared to the difference between *simple* strategy and *no adaptation* strategy at all.

The examples included in this thesis illustrate various types of concept drifts as well as their effect on prediction performance in a broad range of application domains, such as mobility, IT, manufacturing or automotive. Therefore, these examples clearly show that the effect of concept drift is not limited to specific domains but rather represents a general challenge of machine learning in any kind of application. This finding should raise the awareness of machine learning practitioners to consider concept drift and its implication in their specific application domain. Professional providers of machine learning technology have already identified this need and are developing appropriate tools, at least for monitoring deployed machine learning models. For instance, Amazon, IBM and Microsoft (Amazon, 2021; IBM, 2021; Microsoft, 2021) have recently introduced specific tools regarding model monitoring in their cloud solutions for machine learning.

11.3 Limitations and Future Research

Despite its contributions, this thesis has certain limitations that at the same time open avenues for future research. The limitations of each individual contribution are already discussed in the respective chapters. Therefore, this section focuses more on the general challenges and limitations that we encountered during our work on handling concept drift. We believe that there are many interesting research results already available. However, we have identified the following limitations as interesting opportunities for future research.

First, the concept drift community is still lacking a larger set of *real-world data sets* for evaluation and benchmarking of different methods. Currently, most of the evaluation is performed with a few simple, simulated data sets which do not accurately represent the complexity in practical settings. On the contrary, data sets associated with real-world problems often have the problem that the exact timing of concept drifts is unknown. Therefore, we argue for the provision of more real-world data sets where detailed information about the given concept drifts is included.

This could be achieved by an in-depth analysis of real use cases or by performing experiments in the field where drift can be induced by changing specific features on purpose (e.g., Souza et al. (2020)). For instance, such data sets could be produced on test benches for specific machine parts by artificially changing the environment (e.g., by changing temperature levels) in the context of predictive maintenance solutions. Additionally, most concept drift handling algorithms are tested with well-formatted structured data only. However, a lot of the recent progress in machine learning and especially deep neural networks deals with *unstructured data* such as image, audio or text data. Specific algorithms and evaluation data sets might be required to properly document progress in this area.

Second, the *availability of true labels* remains a challenge for the deployment of concept drift solutions. This especially applies to use cases dealing with unstructured data (e.g., images) where true labels often can only be provided by domain experts. We introduce some novel methods for the problem of missing true labels in this thesis, but especially concept drift adaptation remains a difficult problem in this context. Inspiration for possible solutions might be found in the literature regarding *semi-supervised learning*, a machine learning approach to improve models by considering both labeled and unlabeled data instances (Zhou & Belkin, 2014). Furthermore, there are first attempts of researchers proposing unsupervised adaptation of machine learning models (Perdomo et al., 2020). It remains to be seen how such approaches perform when confronted with severe concept drifts associated with significant changes in data distributions.

Third, the vast majority of concept drift research focuses on the supervised machine learning paradigm. Nevertheless, *unsupervised machine learning* techniques such as clustering or frequent itemset mining also play an important role for many applications in organizations. For instance, a customer segmentation based on clustering algorithms might evolve over time due to shifting preferences. This example highlights the need for a larger number of proper detection and adaptation algorithms for unsupervised problems as well. This also applies to *reinforcement learning*, the third large group of machine learning algorithms. Despite an increased general research focus on reinforcement learning, only few researchers analyze this group of algorithms with respect to the effects of concept drift. Due to the iterative nature of reinforcement learning, where agents receive different rewards based on their decisions over time, algorithms in this group might be more suitable to adapt to change. However, it remains an interesting research objective to investigate how those agents can quickly adapt in case of sudden shocks and changes.

Fourth, concept drift handling algorithms need to be more *transparent* in their decisions regarding detection and adaptation. Especially if fully automated decision systems with self-adapting capabilities are deployed, organizations will demand a detailed understanding of the system's behavior. Therefore, additional methods for the *explanation of concept drifts* are needed. This does not only refer to the timing and extent of concept drift but probably most importantly to the underlying reasons. Such explanations are especially important when the decision system is not only evaluated regarding its prediction performance but also against other metrics. For instance, legislation in different countries now requires that automated decisions are explainable and do not discriminate against specific sub-groups of a population. Therefore, more research is required regarding how to guarantee that machine learning models deployed in concept drifting environments continuously comply with fairness requirements.

Fifth, fully automated decision-making still requires a lot of development effort for many tasks and might not even be desirable in specific situations. Often, successful collaboration between automated systems and human experts results in overall better task performance (Huang & Rust, 2018). Thus, concept drift detection and adaptation provide fruitful research topics for implementing and designing appropriate *human-computer collaboration* (e.g., Vössing et al. (2019)). For instance, a change detection algorithm with low sensitivity could be implemented, triggering an alert for domain experts in case of slight changes. The expert can then perform additional analyses and decide if and how to adapt the corresponding machine learning model. Subsequently, the decision of the expert can also be integrated as feedback into the drift detection system to improve detection accuracy, similar to human-in-the-loop systems in supervised machine learning (Holzinger, 2016). Besides better prediction performance, such collaboration can also increase employees' *trust* in these systems as they get more accustomed to algorithmic suggestions and decisions.

Sixth, setting up concept drift solutions will probably remain a manual process for the foreseeable future. Therefore, a *framework* providing guidance for the selection of an appropriate handling strategy represents a worthwhile research objective. As input, characteristics of the data stream such as size, speed as well as application domain could be given. Based on this information, the framework might suggest appropriate components for implementing a holistic drift handling strategy.

The growing capabilities of machine learning technology will surely increase the share of deployed models in practice. However, it is crucial that appropriate monitoring and adaptation tools be implemented. Otherwise, many deployment projects

will be completed with disappointing results, potentially leading to another period of AI winter where this technology is going to be viewed much more critically (Floridi, 2020). We believe that our research can help to avoid such a scenario and increase robustness and performance of deployed machine learning-based information systems. In this way, we hope that this thesis provides a contribution to help this promising technology to reach its full potential.

Bibliography

Abbasi, A., Albrecht, C., Vance, A., & Hansen, J. (2012). MetaFraud: A meta-learning framework for detecting financial fraud. *MIS Quarterly*, *36*(4), 1293–1327 (cit. on p. 19).

Abdullah, H., Qasem, A., Mohammed, N., & Emad, M. (2011). A comparison study between data mining tools over some classification methods. *International Journal of Advanced Computer Science and Applications, Special Issue on Artificial Intelligence* (cit. on p. 52).

Acito, F., & Khatri, V. (2014). Business analytics: Why now and what next? *Business Horizons*, *57*(5), 565–570 (cit. on p. 73).

Adam, M., Toutaoui, J., Pfeuffer, N., & Hinz, O. (2019). Investment decisions with robo-advisors: The role of anthropomorphism and personalized anchors in recommendations. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 19).

Adler, P. S., & Clark, K. B. (1991). Behind the learning curve: A sketch of the learning process. *Management Science*, *37*(3), 267–281 (cit. on p. 195).

Ågerfalk, P. J. (2020). Artificial intelligence as digital agency. *European Journal of Information Systems*, *29*(1), 1–8 (cit. on p. 3).

Aggarwal, C. C. (2006). On biased reservoir sampling in the presence of stream evolution. *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 607–618 (cit. on p. 22).

Aggarwal, C. C. (2015). *Data Mining. The Textbook*. Springer International Publishing. (Cit. on pp. 185, 187, 188, 191, 193).

Aggarwal, C. C., Philip, S. Y., Han, J., & Wang, J. (2003). A framework for clustering evolving data streams. *Proceedings of the International Conference on Very Large Data Bases (VLDB)* (cit. on pp. 23, 93, 107, 127, 163).

Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In C. C. Aggarwal & C. Zhai (Eds.), *Mining text data* (pp. 163–222). Springer US. (Cit. on p. 52).

Agrawal, S., & Goyal, N. (2012). Analysis of Thompson sampling for the multi-armed bandit problem. *Proceedings of the Annual Conference on Learning Theory*, 39.1–39.26 (cit. on p. 25).

Agrawal, V., Panigrahi, B. K., & Subbarao, P. M. V. (2018). Increasing reliability of fault detection systems for industrial applications. *IEEE Intelligent Systems*, *33*(3), 28–39 (cit. on p. 112).

Akhtar, Z., Ahmed, A., Erdem, C. E., & Foresti, G. L. (2015). Adaptive facial recognition under ageing effect. In A. Rattani, F. Roli, & E. Granger (Eds.), *Adaptive Biometric Systems: Recent Advances and Challenges* (pp. 97–117). Springer International Publishing. (Cit. on p. 38).

Alippi, C., & Roveri, M. (2008). Just-in-time adaptive classifiers—Part I: Detecting nonstationary changes. *IEEE Transactions on Neural Networks*, *19*(7), 1145–1153 (cit. on pp. 30, 32).

Alla, S., & Adari, S. K. (2021). What Is MLOps? *Beginning MLOps with MLFlow: Deploy models in AWS SageMaker, Google Cloud, and Microsoft Azure* (pp. 79–124). Apress. (Cit. on pp. 214, 215).

Alsheibani, S., Cheung, Y., & Messom, C. (2018). Artificial intelligence adoption: AI-readiness at firm-level. *Proceedings of the Pacific Asia Conference on Information Systems (PACIS)*, 37 (cit. on p. 18).

Alsheikh, M. A., Lin, S., Niyato, D., & Tan, H. (2014). Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys Tutorials*, *16*(4), 1996–2018 (cit. on pp. 130, 183).

Amazon. (2021). Amazon SageMaker Model Monitor. Retrieved January 15, 2021, from https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor.html. (Cit. on p. 216)

Ambati, S. (2019). The power of open source AI. Retrieved January 14, 2021, from https://www.forbes.com/sites/insights-intelai/2019/05/22/the-power-of-open-source-ai/?sh=35f12bb26300. (Cit. on p. 3)

Amrit, C., Wijnhoven, F., & Beckers, D. (2015). Information waste on the World Wide Web and combating the clutter. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 61).

An, X., Zhou, X., Lü, X., Lin, F., & Yang, L. (2018). Sample selected extreme learning machine based intrusion detection in fog computing and MEC. *Wireless Communications and Mobile Computing*, *2018* (cit. on p. 183).

Anand, S. S., & Büchner, A. G. (1998). *Decision support using data mining*. Financial Times Management. (Cit. on p. 46).

Anderson, M., & Anderson, S. L. (2015). Toward ensuring ethical behavior from autonomous systems: A case-supported principle-based paradigm. *Industrial Robot: An International Journal*, *42*(4), 324–331 (cit. on p. 71).

Andrews, D. W. (1993). Tests for parameter instability and structural change with unknown change point. *Econometrica: Journal of the Econometric Society*, *61*(4), 821–856 (cit. on p. 33).

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In J. Bravo, R. Hervás, & M. Rodríguez (Eds.), *Ambient assisted living and home care* (pp. 216–223). Springer Berlin Heidelberg. (Cit. on p. 183).

Anguita, D., Ghio, A., Pischiutta, S., & Ridella, S. (2007). A hardware-friendly support vector machine for embedded automotive applications. *International Joint Conference on Neural Networks (IJCNN)*, 1360–1364 (cit. on p. 183).

Antoch, J., Hanousek, J., Horváth, L., Hušková, M., & Wang, S. (2019). Structural breaks in panel data: Large number of panels and short length time series. *Econometric Reviews*, *38*(7), 828–855 (cit. on p. 28).

Armstrong, J. S., & Collopy, F. (1992). Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, *8*(1), 69–80 (cit. on p. 194).

Aue, A., & Horváth, L. (2013). Structural breaks in time series. *Journal of Time Series Analysis*, *34*(1), 1–16 (cit. on p. 28).

Babcock, B., Datar, M., & Motwani, R. (2001). Sampling from a moving window over streaming data. *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (cit. on p. 23).

Bach, S. H., & Maloof, M. A. (2008). Paired learners for concept drift. *Proceedings of IEEE International Conference on Data Mining (ICDM)*, 23–32 (cit. on pp. 130, 135).

Baena-Garcia, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., & Morales-Bueno, R. (2006). Early drift detection method. *Fourth International Workshop on Knowledge Discovery from Data Streams*, *6*, 77–86 (cit. on pp. 129, 168).

Baesens, B., Bapna, R., Marsden, J. R., Vanthienen, J., & Zhao, J. L. (2014). Transformational issues of big data and analytics in networked business. *MIS Quarterly*, *38*(2), 629–631 (cit. on pp. 70, 72, 73).

Bahja, M. (2018). Identifying patient experience from online resources via sentiment analysis and topic modelling approaches. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 20).

Bai, J. (1994). Least squares estimation of a shift in linear processes. *Journal of Time Series Analysis*, *15*(5), 453–472 (cit. on p. 33).

Bai, J., & Perron, P. (1998). Estimating and testing linear models with multiple structural changes. *Econometrica*, *66*(1), 47–78 (cit. on p. 33).

Bai, J., & Perron, P. (2003). Computation and analysis of multiple structural change models. *Journal of Applied Econometrics*, *18*(1), 1–22 (cit. on p. 28).

Baier, L., Hofmann, M., Kühl, N., Mohr, M., & Satzger, G. (2020). Handling concept drifts in regression problems–the error intersection approach. *International Conference on Wirtschaftsinformatik* (cit. on pp. 160, 184).

Baier, L., Jöhren, F., & Seebacher, S. (2019). Challenges in the deployment and operation of machine learning in practice. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on pp. 91, 143).

Baier, L., Kühl, N., & Satzger, G. (2019). How to cope with change?-preserving validity of predictive services over time. *Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS)* (cit. on pp. 54, 64, 72, 91, 96, 127, 147, 182).

Banko, M., & Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics - ACL '01* (cit. on p. 49).

Barbon Junior, S., Tavares, G. M., da Costa, V. G. T., Ceravolo, P., & Damiani, E. (2018). A framework for human-in-the-loop monitoring of concept-drift detection in event log stream. *Companion Proceedings of the The Web Conference*, 319–326 (cit. on p. 38).

Barnett, V., & Lewis, T. (1994). Outliers in Statistical Data, John Wiley. *New York* (cit. on p. 184).

Barocas, S., Hardt, M., & Narayanan, A. (2017). Fairness in machine learning. *NIPS Tutorial* (cit. on pp. 6, 50).

Barreno, M., Nelson, B., Joseph, A. D., & Tygar, J. D. (2010). The security of machine learning. *Machine Learning*, *81*(2), 121–148 (cit. on p. 143).

Baumann, A., Lessmann, S., Coussement, K., & Bock, K. D. (2015). Maximize what matters: Predicting customer churn with decision-centric ensemble selection. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 19).

Bawack, R. E., Fosso Wamba, S., & Carillo, K. (2019). Artificial intelligence in practice: Implications for IS research. *Proceedings of the Americas Conference on Information Systems (AMCIS)* (cit. on p. 4).

Beluch, W. H., Genewein, T., Nürnberger, A., & Köhler, J. M. (2018). The power of ensembles for active learning in image classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 167).

Benlian, A., Klumpe, J., & Hinz, O. (2020). Mitigating the intrusive effects of smart home assistants by using anthropomorphic design features: A multimethod investigation. *Information Systems Journal*, *30*(6), 1010–1042 (cit. on p. 19).

Berente, N., Gu, B., Recker, J., & Santhanam, R. (2019). Call for papers MISQ special issue on managing AI. *MIS Quarterly* (cit. on pp. 4, 17, 18).

Bergstra, J., & Yoshua, B. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281–305 (cit. on p. 195).

Beyazit, E., Alagurajah, J., & Wu, X. (2019). Online learning from data streams with varying feature spaces. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 3232–3239 (cit. on pp. 24, 26).

Bifet, A. (2017). Classifier concept drift detection and the illusion of progress. In L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, & J. M. Zurada (Eds.), *Artificial Intelligence and Soft Computing* (pp. 715–725). Springer International Publishing. (Cit. on pp. 35, 36).

Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. *Proceedings of the 2007 SIAM International Conference on Data Mining*, 443–448 (cit. on pp. 30, 31, 94, 129, 145, 163, 165, 168).

Bifet, A., Gavaldà, R., Holmes, G., & Pfahringer, B. (2018). *Machine learning for data streams: With practical examples in MOA*. MIT Press. (Cit. on pp. 21, 22).

Bifet, A., Holmes, G., Pfahringer, B., & Frank, E. (2010). Fast perceptron decision tree learning from evolving data streams. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 299–310 (cit. on p. 36).

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 139–148 (cit. on pp. 36, 37).

Bifet, A., Read, J., Pfahringer, B., Holmes, G., & Žliobaitė, I. (2013). CD-MOA: Change detection framework for massive online analysis. *International Symposium on Intelligent Data Analysis* (cit. on p. 174).

Bifet, A., Read, J., Žliobaitė, I., Pfahringer, B., & Holmes, G. (2013). Pitfalls in benchmarking data stream classification and how to avoid them. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 465–479 (cit. on p. 36).

Black, M., & Hickey, R. (2002). Classification of customer call data in the presence of concept drift and noise. In D. Bustard, W. Liu, & R. Sterritt (Eds.), *Software 2002: Computing in an imperfect world* (pp. 74–87). Springer Berlin Heidelberg. (Cit. on p. 112).

Black, M., & Hickey, R. (2004). Detecting and adapting to concept drift in bioinformatics. *Knowledge Exploration in Life Science Informatics*, 161–168 (cit. on pp. 112, 114, 118).

Blanc, S. M. (2016). *Bias-Variance Aware Integration of Judgmental Forecasts and Statistical Models* (Doctoral dissertation). Karlsruher Institut für Technologie (KIT). (Cit. on p. 52).

Blenk, A., Kalmbach, P., Kellerer, W., & Schmid, S. (2017). O'zapft is: Tap your network algorithm's big data! *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, 19–24 (cit. on p. 70).

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. *International Conference on Machine Learning (ICML)* (cit. on p. 167).

Bose, R. P. C., Van Der Aalst, W., Žliobaitė, I., & Pechenizkiy, M. (2014). Dealing with concept drifts in process mining. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(1), 154–171 (cit. on pp. 95, 112, 119).

Boutaba, R., Salahuddin, M., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., & Caicedo, O. (2018). A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. *Journal of Internet Services and Applications*, *9*(1), 16 (cit. on pp. 67, 69, 72, 73).

Box, G. E. P., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, *26*(2), 211–243 (cit. on p. 100).

Brahma, A., Chatterjee, S., & Seal, K. C. (2020). Understanding cardiovascular disease progression behavior from patient cohort data using markov chain model. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 20).

Bretschneider, U., & Peters, R. (2016). Detecting cyberbullying in online communities. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 62).

Brodley, C. E., Rebbapragada, U., Small, K., & Wallace, B. (2012). Challenges and opportunities in applied machine learning. *AI Magazine*, *33*(1), 11–24 (cit. on pp. 69, 70).

Brodley, C. E., & Smyth, P. (1995). The process of applying machine learning algorithms. *Proceedings of the ICML workshop on Applying Machine Learning in Practice* (cit. on p. 46).

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (cit. on p. 3).

Brynjolfsson, E., & Mitchell, T. (2017). What can machine learning do? Workforce implications. *Science*, *358*(6370), 1530–1534 (cit. on p. 3).

Brzezinski, D., & Stefanowski, J. (2014). Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, *265*, 50–67 (cit. on p. 158).

Bughin, J., Hazan, E., Ramaswamy, S., Henke, N., Trench, M., Dahlstroem, P., Allas, T., & Chui, M. (2017). Artificial intelligence, the next digital frontier? *McKinsey and Company Global Institute*, *47* (cit. on pp. 3, 4, 6).

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., et al. (2013). API design for machine learning software: Experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238* (cit. on p. 143).

Buxmann, P., Hess, T., & Thatcher, J. B. (2021). AI-based information systems. *Business & Information Systems Engineering*, *63*(1), 1–4 (cit. on p. 7).

Cabena, P., Hadjinian, P., Stadler, R., Verhees, J., & Zanasi, A. (1998). *Discovering data mining: from concept to implementation*. Prentice-Hall, Inc. (Cit. on p. 46).

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, *300*, 70–79 (cit. on pp. 5, 67, 71).

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *International Conference on Machine Learning (ICML)* (cit. on p. 51).

Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. d. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, *137*, 106024 (cit. on p. 3).

Cattral, R., Oppacher, F., & Deugo, D. (2002). Evolutionary data mining with automatic rule generalization. *Recent Advances in Computers, Computing and Communications*, *1*(1), 296–300 (cit. on p. 37).

Cavalcante, R. C., Minku, L. L., & Oliveira, A. L. I. (2016). Fedd: Feature extraction for explicit concept drift detection in time series. *International Joint Conference on Neural Networks (IJCNN)*, 740–747 (cit. on pp. 9, 35, 129, 184).

Cawley, G. C., & Talbot, N. L. C. (2010). On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research*, *11* (cit. on p. 53).

Ceci, M., Corizzo, R., Malerba, D., & Rashkovska, A. (2019). Spatial autocorrelation and entropy for renewable energy forecasting. *Data Mining and Knowledge Discovery*, *33*(3), 698–729 (cit. on p. 38).

Chandola, V., Banerjee, A., & Kumar, V. (2007). Outlier detection: A survey. *ACM Computing Surveys* (cit. on p. 185).

Chatterjee, S., Byun, J., Dutta, K., Pedersen, R. U., Pottathil, A., & Xie, H. (2018). Designing an Internet-of-Things (IoT) and sensor-based in-home monitoring system for assisting diabetes patients: Iterative learning from two case studies. *European Journal of Information Systems*, *27*(6), 670–685 (cit. on p. 20).

Chatterjee, S., Saeedfar, P., Tofangchi, S., & Kolbe, L. M. (2018). Intelligent road maintenance: A machine learning approach for surface defect detection. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on pp. 20, 63).

Chawla, N. V. (2010). Data mining for imbalanced datasets: An overview. In O. Maimon & L. Rokach (Eds.), *Data Mining and Knowledge Discovery Handbook* (pp. 875–886). Springer US. (Cit. on p. 51).

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357 (cit. on p. 51).

Chen, H., Chiang, R., & Storey, V. (2012). Business intelligence and analytics: From big data to big impact. *MIS Quarterly*, *36*(4), 1165–1188 (cit. on pp. 67, 91, 107, 127, 181).

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794 (cit. on p. 141).

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 1–13 (cit. on p. 174).

Ching, P. G. J. (2018). In AI we trust: Perceived value of advanced artificial intelligence in services. *PhD thesis, The University of Queensland* (cit. on p. 91).

Chollet, F. (2018). *Deep Learning with Python*. Manning Publications. (Cit. on p. 61).

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regressions. *Econometrica*, *28*(3), 591–605 (cit. on p. 32).

Cios, K. J., Teresinska, A., Konieczna, S., Potocka, J., & Sharma, S. (2000). Diagnosing myocardial perfusion from PECT bull's-eye maps-A knowledge discovery approach. *IEEE Engineering in Medicine and Biology Magazine*, *19*(4), 17–25 (cit. on p. 46).

Coiera, E., Ammenwerth, E., Georgiou, A., & Magrabi, F. (2018). Does health informatics have a replication crisis? *Journal of the American Medical Informatics Association*, *25*(8), 963–968 (cit. on p. 43).

Coussement, K., Lessmann, S., & Verstraeten, G. (2017). A comparative analysis of data preparation algorithms for customer churn prediction: A case study in the telecommunication industry. *Decision Support Systems*, *95*, 27–36 (cit. on p. 19).

Cramer, J., & Krueger, A. B. (2016). Disruptive change in the taxi business: The case of Uber. *American Economic Review*, *106*(5), 177–182 (cit. on pp. 132, 149).

Cui, G., Wong, M. L., & Wan, X. (2012). Cost-sensitive learning via priority sampling to improve the return on marketing and CRM investment. *Journal of Management Information Systems*, *29*(1), 341–374 (cit. on p. 63).

Cummings, M. (2004). Automation bias in intelligent time critical decision support systems. *AIAA Intelligent Systems Technical Conference*, 6313 (cit. on p. 160).

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2017). Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(8) (cit. on pp. 38, 184).

Dasu, T., Krishnan, S., Venkatasubramanian, S., & Yi, K. (2006). An information-theoretic approach to detecting changes in multi-dimensional data streams. *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications* (cit. on pp. 30, 31).

Davenport, T. H. (2006). Competing on analytics. *Harvard Business Review*, *84*(1), 98–107 (cit. on pp. 67, 91, 107).

Davenport, T. H. (2018). From analytics to artificial intelligence. *Journal of Business Analytics*, *1*(2), 73–80 (cit. on p. 3).

De Caigny, A., Coussement, K., & de Bock, K. W. (2018). A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. *European Journal of Operational Research*, *269*(2), 760–772 (cit. on p. 19).

De Gooijer, J. G., & Hyndman, R. J. (2006). 25 years of time series forecasting. *International Journal of Forecasting*, *22*(3), 443–473 (cit. on p. 146).

De Maesschalck, R., Jouan-Rimbaud, D., & Massart, D. (2000). The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, *50*(1), 1–18 (cit. on p. 191).

De Vito, S., Massera, E., Piga, M., Martinotto, L., & Di Francia, G. (2008). On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*, *129*(2), 750–757 (cit. on p. 173).

Deeplearning.AI. (2020). The batch, issue 83, november 18th 2020. Retrieved January 14, 2021, from https://blog.deeplearning.ai/blog/the-batch-bias-in-surprising-places-retail-models-adjust-to-covid-faster-transformers-ai-patents-explode?. (Cit. on p. 7)

Delany, S. J., Cunningham, P., & Tsymbal, A. (2006). A comparison of ensemble and case-base maintenance techniques for handling concept drift in spam filtering. *Proceedings of FLAIRS Conference*, 340–345 (cit. on pp. 110, 112, 118).

Delone, W. H., & McLean, E. R. (2003). The DeLone and McLean model of information systems success: A ten-year update. *Journal of Management Information Systems*, *19*(4), 9–30 (cit. on pp. 5, 205).

Demetis, D., & Lee, A. S. (2018). When humans using the IT artifact becomes it using the human artifact. *Journal of the Association for Information Systems*, *19*(10), 929–952 (cit. on p. 18).

Demšar, J., & Bosnić, Z. (2018). Detecting concept drift in data streams using model explanation. *Expert Systems with Applications*, *92*, 546–559 (cit. on p. 39).

Demšar, J., Bosnić, Z., & Kononenko, I. (2014). Visualization and concept drift detection using explanations of incremental models. *Informatica*, *38*(4) (cit. on p. 38).

Der Kiureghian, A., & Ditlevsen, O. (2009). Aleatory or epistemic? Does it matter? *Structural Safety* (cit. on p. 166).

Dhar, V., Geva, T., Oestreicher-Singer, G., & Sundararajan, A. (2014). Prediction in economic networks. *Information Systems Research*, *25*(2), 264–284 (cit. on p. 50).

Di Francescomarino, C., Dumas, M., Maggi, F. M., & Teinemaa, I. (2017). Clustering-based predictive process monitoring. *IEEE Transactions on Services Computing*, *12*(6), 896–909 (cit. on p. 94).

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, *20*(1), 134–144 (cit. on pp. 138, 156).

Dietterich, T. G., Domingos, P., Getoor, L., Muggleton, S., & Tadepalli, P. (2008). Structured machine learning: The next ten years. *Machine Learning*, *73*(1), 3–23 (cit. on pp. 71, 72).

Ding, A. W., Li, S., & Chatterjee, P. (2015). Learning user real-time intent for optimal dynamic web page transformation. *Information Systems Research*, *26*(2), 339–359 (cit. on pp. 19, 61).

Ding, Y., & Li, X. (2005). Time weight collaborative filtering. *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, 485–492 (cit. on p. 112).

Dinges, V., Urmetzer, F., Martinez, V., Zaki, M., & Neely, A. (2015). The future of servitization: Technologies that will make a difference. *Cambridge Service Alliance Executive Briefing Paper* (cit. on pp. 67, 107).

Ditzler, G., Roveri, M., Alippi, C., & Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, *10*(4), 12–25 (cit. on pp. 34, 35, 129).

Ditzler, G., & Polikar, R. (2011). Hellinger distance based drift detection for nonstationary environments. *2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, 41–48 (cit. on p. 145).

Dodge, J., Gururangan, S., Card, D., Schwartz, R., & Smith, N. A. (2019). Show your work: Improved reporting of experimental results. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (cit. on pp. 47, 49).

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78–87 (cit. on pp. 56, 71, 73).

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 71–80 (cit. on p. 23).

Domingos, P., & Hulten, G. (2001). Catching up with the data: Research issues in mining data streams. *Workshop on Research Issues in Data Mining and Knowledge Discovery* (cit. on p. 21).

Dong, W., Liao, S., & Zhang, Z. (2018). Leveraging financial social media data for corporate fraud detection. *Journal of Management Information Systems*, *35*(2), 461–487 (cit. on p. 19).

Dorner, V., & Alpers, G. W. (2017). Detecting panic potential in social media tweets. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 61).

Dos Reis, D. M., Flach, P., Matwin, S., & Batista, G. (2016). Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (cit. on p. 166).

Duarte, A. I. M., & Costa, C. J. (2012). Information systems: Life cycle and success. *Proceedings of the Workshop on Information Systems and Design of Communication*, 25–30 (cit. on pp. 6, 206).

Dufour, J.-M. (1982). Generalized chow tests for structural change: A coordinate-free approach. *International Economic Review*, *23*(3), 565–575 (cit. on p. 32).

Dumas, M., La Rosa, M., Mendling, J., Reijers, H. A., et al. (2013). *Fundamentals of business process management*. Springer. (Cit. on p. 94).

Dunning, T., & Friedman, E. (2017). *Machine learning logistics*. O'Reilly Media, Inc. (Cit. on pp. 130, 135).

Dyck, J. (2018). Machine learning for engineering. *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, 422–427 (cit. on pp. 72, 73).

Dyer, K. B., Capo, R., & Polikar, R. (2014). Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(1), 12–26 (cit. on p. 166).

Ekanayake, J., Tappolet, J., Gall, H. C., & Bernstein, A. (2009). Tracking concept drift of software projects using defect prediction quality. *Proceedings of 6th IEEE International Working Conference on Mining Software Repositories*, 51–60 (cit. on p. 112).

Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, *22*(10), 1517–1531 (cit. on pp. 36, 152, 174).

Enders, T., Wolff, C., & Satzger, G. (2020). Knowing what to share: Selective revealing in open data. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 18).

Esswein, M., Mayer, J. H., Stoffel, S., & Quick, R. (2019). Predictive analytics—A modern crystal ball? Answers from a cash flow case study. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on pp. 144, 146).

Evermann, J., Rehse, J.-R., & Fettke, P. (2016). A deep learning approach for predicting process behaviour at runtime. *International Conference on Business Process Management*, 327–338 (cit. on p. 19).

Fan, W., Huang, Y.-a., Wang, H., & Yu, P. S. (2004). Active mining of data streams. *SIAM SDM* (cit. on p. 165).

Fanaee-T, H., & Gama, J. (2014). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, *2*(2), 113–127 (cit. on p. 173).

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, *39*(11), 27–34 (cit. on p. 46).

Fdez-Riverola, F., Iglesias, E. L., Díaz, F., Méndez, J. R., & Corchado, J. M. (2007). Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications*, *33*(1), 36–48 (cit. on pp. 110, 112, 120).

Ferguson, A. L. (2017). Machine learning and data science in soft materials engineering. *Journal of Physics: Condensed Matter*, *30*(4) (cit. on pp. 71, 73).

Fernald, J. G., Hall, R. E., Stock, J. H., & Watson, M. W. (2017). *The disappointing recovery of output after 2009* (tech. rep.). National Bureau of Economic Research. (Cit. on p. 130).

Ferrucci, D., Levas, A., Bagchi, S., Gondek, D., & Mueller, E. T. (2013). Watson: Beyond jeopardy! *Artificial Intelligence*, *199*, 93–105 (cit. on p. 3).

Feuerriegel, S., & Fehrer, R. (2016). Improving Decision Analytics with Deep Learning: the Case of Financial Disclosures. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 45).

Feuerriegel, S., Riedlinger, S., & Neumann, D. (2014). Predictive Analytics For Electricity Prices Using Feed-Ins from Renewables. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 61).

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in Neural Information Processing Systems*, 2962–2970 (cit. on p. 54).

Fisher, R. A. (1936). The use of multiple measures in taxonomic problems. *Annals of Eugenics*, *7*(2), 179–188 (cit. on pp. 57, 58).

Floridi, L. (2020). AI and its new winter: From myths to realities. *Philosophy & Technology*, *33*(1), 1–3 (cit. on p. 219).

Forman, G. (2002). Incremental machine learning to reduce biochemistry lab costs in the search for drug discovery. *BIOKDD*, 33–36 (cit. on p. 112).

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, *19*(1), 1–67 (cit. on p. 172).

Friedman, J. H. (1997). On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery*, *1*(1), 55–77 (cit. on p. 53).

Fromm, H., Habryn, F., & Satzger, G. (2012). Service analytics: Leveraging data across enterprise boundaries for competitive advantage. In U. Bäumer, P. Kreutter, & W. Messner (Eds.), *Globalization of professional services: Innovative strategies, successful processes, inspired talent management, and first-hand experiences* (pp. 139–149). Springer Berlin Heidelberg. (Cit. on p. 109).

Fu, L.-M. (2003). *Neural networks in computer intelligence*. Tata McGraw-Hill Education. (Cit. on p. 44).

Gaber, M. M. (2012). Advances in data stream mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), 79–85 (cit. on p. 23).

Gaber, M. M., Zaslavsky, A., & Krishnaswamy, S. (2005). Mining data streams: A review. *SIGMOD Rec.*, *34*(2), 18–26 (cit. on pp. 21–23).

Gago, P., Silva, Á., & Santos, M. F. (2007). Adaptive decision support for intensive care. *Portuguese Conference on Artificial Intelligence*, 415–425 (cit. on p. 112).

Gal, Y. (2016). *Uncertainty in deep learning* (Doctoral dissertation). University of Cambridge. (Cit. on p. 166).

Gal, Y., & Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *International Conference on Machine Learning (ICML)* (cit. on pp. 11, 164, 167, 169, 170).

Gama, J. (2012). A survey on learning from data streams: Current and future trends. *Progress in Artificial Intelligence*, *1*(1), 45–55 (cit. on pp. 21–23).

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. *Brazilian symposium on artificial intelligence* (cit. on pp. 30, 38, 54, 64, 163, 165, 168, 172).

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, *46*(4), 1–37 (cit. on pp. 4, 7, 9, 27–29, 34–36, 72, 92–94, 108, 110, 112, 115, 118, 122, 128, 135, 145, 147, 149, 164, 165, 182, 184, 200).

Gao, J., Fan, W., Han, J., & Yu, P. S. (2007). A general framework for mining concept-drifting data streams with skewed distributions. *Proceedings of the 2007 SIAM International Conference on Data Mining*, 3–14 (cit. on p. 27).

García-Laencina, P., Figueiras-Vidal, A., & Sancho-Gomez, J.-L. (2008). Machine learning techniques for solving classification problems with missing input data. *Proceedings of the 12th World Multi-Conference on Systems, Cybernetics and Informatics* (cit. on p. 69).

Geva, T., & Oestreicher-Singer, G. (2013). Do customers speak their minds? Using forums and search for predicting sales. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on pp. 61, 63, 64).

Gholami, R., Watson, R. T., Molla, A., Hasan, H., & Bjorn-Andersen, N. (2016). Information systems solutions for environmental sustainability: How can we do more? *Journal of the Association for Information Systems*, *17*(8), 521 (cit. on pp. 64, 207).

Giacomini, R., & Rossi, B. (2009). Detecting and predicting forecast breakdowns. *Review of Economic Studies*, *76*(2), 669–705 (cit. on p. 111).

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 80–89 (cit. on p. 141).

Gimpel, H., Kleindienst, D., & Waldmann, D. (2018). The disclosure of private data: measuring the privacy paradox in digital services. *Electronic Markets*, *28*(4), 475–490 (cit. on p. 46).

Glynn, J., Perera, N., & Verma, R. (2007). Unit root tests and structural breaks: A survey with applications. *Revista de Métodos Cuantitativos para la Economia y la Empresa*, *3* (cit. on p. 146).

Golub, G. H., Heath, M., & Wahba, G. (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, *21*(2), 215–223 (cit. on pp. 53, 195).

Gonçalves, P. M., De Carvalho Santos, S. G., Barros, R. S., & Vieira, D. C. (2014). A comparative study on concept drift detectors. *Expert Systems with Applications*, *41*(18) (cit. on pp. 36, 37, 152).

Gong, J., Abhishek, V., & Li, B. (2018). Examining the impact of keyword ambiguity on search advertising performance: A topic model approach. *MIS Quarterly*, *42*(3) (cit. on p. 45).

Goutte, C., & Gaussier, E. (2005). A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. *European Conference on Information Retrieval*, 345–359 (cit. on p. 52).

Graepel, T., Lauter, K., & Naehrig, M. (2012). ML confidential: Machine learning on encrypted data. *International Conference on Information Security and Cryptology*, 1–21 (cit. on p. 70).

Guajardo, J. A., Weber, R., & Miranda, J. (2010). A model updating strategy for predicting time series with seasonal patterns. *Applied Soft Computing Journal*, *10*(1), 276–283 (cit. on p. 128).

Günther, W. A., Mehrizi, M. H. R., Huysman, M., & Feldberg, F. (2017). Debating big data: A literature review on realizing value from big data. *The Journal of Strategic Information Systems*, *26*(3), 191–209 (cit. on p. 18).

Guo, J., Zhang, W., Fan, W., & Li, W. (2018). Combining geographical and social influences with deep learning for personalized point-of-interest recommendation. *Journal of Management Information Systems*, *35*(4), 1121–1153 (cit. on p. 19).

Gupta, S., Agrawal, A., Gopalakrishnan, K., & Narayanan, P. (2015). Deep learning with limited numerical precision. *International Conference on Machine Learning*, 1737–1746 (cit. on p. 183).

Haldrup, N., Kruse, R., Teräsvirta, T., Varneskov, R. T., et al. (2013). Unit roots, nonlinearities and structural breaks. *Handbook of Research Methods and Applications in Empirical Finance. Cheltenham: Edward Elgar*, 61–72 (cit. on pp. 33, 145).

Haley, P. J., & Soloway, D. (1992). Extrapolation limitations of multilayer feedforward neural networks. *International Joint Conference on Neural Networks* (cit. on p. 166).

Hall, M., National, H., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, *11*(1), 10–18 (cit. on p. 51).

Han, H., Otto, C., Liu, X., & Jain, A. K. (2015). Demographic estimation from face images: Human vs. machine performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(6), 1148–1161 (cit. on p. 45).

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques*. Morgan Kaufmann, San Francisco, CA. (Cit. on pp. 109, 118).

Han, X., Wang, L., & Huang, H. (2017). Deep investment behavior profiling by recurrent neural network in P2P lending. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on pp. 63, 64).

Hansen, B. E. (2001). The new econometrics of structural change: Dating breaks in US labour productivity. *Journal of Economic Perspectives*, *15*(4), 117–128 (cit. on pp. 32, 33).

Harries, M., & Horn, K. (1995). Detecting concept drift in financial time series prediction using symbolic machine learning. *AI Conference* (cit. on pp. 111, 112, 120).

Harries, M. (1999). *Splice-2 comparative evaluation: Electricity pricing* (Technical Report). University of New South Wales. (Cit. on p. 37).

Harries, M. B., Sammut, C., & Horn, K. (1998). Extracting hidden context. *Machine Learning*, *32*(2), 101–126 (cit. on p. 112).

Harrington, P. (2012). *Machine learning in action*. Manning Publications Co. (Cit. on pp. 9, 35).

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Overview of supervised learning. *The Elements of Statistical Learning* (pp. 9–41). Springer. (Cit. on p. 51).

Hazelwood, K., Bird, S., Brooks, D., Chintala, S., Diril, U., Dzhulgakov, D., Fawzy, M., Jia, B., Jia, Y., Kalro, A., et al. (2018). Applied machine learning at facebook: A datacenter infrastructure perspective. *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 620–629 (cit. on pp. 67, 72).

He, H., & Ma, Y. (2013). *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons. (Cit. on p. 50).

Heaven, W. D. (2020). Our weird behavior during the pandemic is messing with AI models. Retrieved January 4, 2021, from https://www.technologyreview.com/2020/05/11/1001563/covid-pandemic-broken-ai-machine-learning-amazon-retail-fraud-humans-in-the-loop/. (Cit. on p. 7)

Heilig, L., Hofer, J., Lessmann, S., & Voc, S. (2016). Data-driven product returns prediction: A cloud-based ensemble selection approach. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 45).

Heinrich, K., Zschech, P., Skouti, T., Griebenow, J., & Riechert, S. (2019). Demystifying the black box: A classification scheme for interpretation and visualization of deep intelligent systems. *Proceedings of the Americas Conference on Information Systems (AMCIS)* (cit. on p. 20).

Heit, J., Liu, J., & Shah, M. (2016). An architecture for the deployment of statistical models for the big data era. *IEEE International Conference on Big Data (Big Data)*, 1377–1384 (cit. on pp. 72, 73).

Helfferich, C. (2011). *Die Qualität qualitativer Daten*. Springer. (Cit. on p. 74).

Hemmer, P., Kühl, N., & Schöffer, J. (2020). DEAL: Deep evidential active learning for image classification. *IEEE International Conference On Machine Learning And Applications* (cit. on pp. 166, 167).

Hennig-Thurau, T., Walsh, G., & Schrader, U. (2004). VHB-JOURQUAL: Ein Ranking von betriebswirtschaftlich-relevanten Zeitschriften auf der Grundlage von Experten-urteilen. *Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung*, *56*(6), 520–545 (cit. on p. 59).

Hernández, L., Baladron, C., Aguiar, J. M., Carro, B., Sanchez-Esguevillas, A., Lloret, J., Chinarro, D., Gomez-Sanz, J. J., & Cook, D. (2013). A multi-agent system architecture for smart grid management and forecasting of energy demand in virtual power plants. *IEEE Communications Magazine*, *51*(1), 106–113 (cit. on pp. 136, 153).

Hernández-Lobato, J. M., & Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. *International Conference on Machine Learning (ICML)* (cit. on p. 167).

Hinder, F., & Hammer, B. (2020). Counterfactual explanations of concept drift. *arXiv preprint arXiv:2006.12822* (cit. on p. 39).

Hirt, R., Kühl, N., & Satzger, G. (2017). An end-to-end process model for supervised machine learning classification: from problem to deployment in information systems. *Proceedings of the International Conference on Design Science Research in Information Systems and Technology* (cit. on pp. 48, 107).

Ho, T. K., & Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(3), 289–300 (cit. on p. 51).

Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, *22*(2), 85–126 (cit. on pp. 184, 185).

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*(301), 13–30 (cit. on p. 23).

Hoens, T. R., Polikar, R., & Chawla, N. V. (2012). Learning from streaming data with concept drift and imbalance: An overview. *Progress in Artificial Intelligence*, *1*(1), 89–101 (cit. on p. 37).

Hoffman, M., Bach, F., & Blei, D. (2010). Online learning for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, *23*, 856–864 (cit. on p. 26).

Holstein, K., Wortman Vaughan, J., Daumé III, H., Dudik, M., & Wallach, H. (2019). Improving fairness in machine learning systems: What do industry practitioners need? *Proceedings of the CHI Conference on Human Factors in Computing Systems* (cit. on p. 4).

Holzinger, A. (2016). Interactive machine learning for health informatics: When do we need the human-in-the-loop? *Brain Informatics*, *3*(2), 119–131 (cit. on pp. 18, 218).

Hu, F., & Hao, Q. (2012). *Intelligent sensor networks: The integration of sensor networks, signal processing and machine learning*. CRC Press. (Cit. on p. 183).

Hu, H., Kantardzic, M., & Lyu, L. (2018). Detecting different types of concept drifts with ensemble framework. *IEEE International Conference on Machine Learning and Applications*, 344–350 (cit. on p. 35).

Hu, H., Kantardzic, M., & Sethi, T. S. (2020). No free lunch theorem for concept drift detection in streaming data classification: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *10*(2) (cit. on pp. 10, 32, 37).

Huang, J., Boh, W. F., & Goh, K. H. (2017). A temporal study of the effects of online opinions: Information sources matter. *Journal of Management Information Systems*, *34*(4), 1169–1202 (cit. on p. 59).

Huang, M.-H., & Rust, R. T. (2018). Artificial intelligence in service. *Journal of Service Research*, *21*(2), 155–172 (cit. on p. 218).

Huang, S., & Dong, Y. (2007). An active learning system for mining time-changing data streams. *Intelligent Data Analysis*, *11*(4), 401–419 (cit. on pp. 112, 121).

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (cit. on pp. 23, 37).

Hutson, M. (2018). Artificial intelligence faces reproducibility crisis. *Science*, *359*(6377), 725–726 (cit. on pp. 8, 43, 47).

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts. (Cit. on p. 136).

IBM. (2021). IBM Watson OpenScale. Retrieved January 15, 2021, from https://www.ibm.com/cloud/watson-openscale. (Cit. on p. 216)

Ikonomovska, E., Gama, J., & Džeroski, S. (2011). Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, *23*(1), 128–168 (cit. on pp. 37, 144, 158).

Ivannikov, A., Pechenizkiy, M., Bakker, J., Leino, T., Jegoroff, M., Kärkkäinen, T., & Äyrämö, S. (2009). Online mass flow prediction in cfb boilers with explicit detection of sudden concept drift. *SIGKDD Explor. Newsl.*, *11*(2) (cit. on pp. 110, 112, 114, 118, 120, 129).

Ivanov, A., & Sharman, R. (2018). Impact of User-Generated Internet Content on Hospital Reputational Dynamics. *Journal of Management Information Systems*, *35*(4), 1277–1300 (cit. on p. 59).

Iwashita, A. S., & Papa, J. P. (2019). An overview on concept drift learning. *IEEE Access*, *7*, 1532–1547 (cit. on p. 35).

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning*. Springer. (Cit. on pp. 53, 63).

Janai, J., Güney, F., Behl, A., Geiger, A., et al. (2020). Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, *12*(1–3), 1–308 (cit. on p. 3).

Jaworski, M. (2018). Regression function and noise variance tracking methods for data streams with concept drift. *International Journal of Applied Mathematics and Computer Science*, *28*(3), 559–567 (cit. on pp. 9, 35).

Jöhnk, J., Weißert, M., & Wyrtki, K. (2020). Ready or not, AI Comes—An interview study of organizational AI readiness factors. *Business & Information Systems Engineering* (cit. on pp. 3, 5, 8, 18).

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260 (cit. on pp. 4, 19, 45, 67, 71, 91, 109, 127, 181, 183).

Kadian, A., Singh, V., & Bhattacherjee, A. (2018). Detecting clickbait using user emotions and behaviors on social media. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 19).

Kadlec, P., & Gabrys, B. (2011). Local learning-based adaptive soft sensor for catalyst activation prediction. *AIChE Journal*, *57*(5), 1288–1301 (cit. on pp. 108, 112, 119, 144, 184).

Kahlen, M., Ketter, W., Lee, T., & Gupta, A. (2017). Optimal prepositioning and fleet sizing to maximize profits for one-way transportation companies. *Proceedings of International Conference on Information Systems (ICIS)* (cit. on pp. 131, 146).

Kambatla, K., Kollias, G., Kumar, V., & Grama, A. (2014). Trends in big data analytics. *Journal of Parallel and Distributed Computing*, *74*(7) (cit. on p. 109).

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *24*(7), 881–892 (cit. on p. 191).

Keim, D., Kohlhammer, J., Ellis, G., & Mansmann, F. (2010). *Mastering the information age solving problems with visual analytics*. Eurographics Association. (Cit. on p. 114).

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? *Proceedings of the Conference on Neural Information Processing Systems* (cit. on p. 168).

Kendall, M. G. (1948). *Rank correlation methods.* Griffin. (Cit. on p. 33).

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., & Ghédira, K. (2018). Discussion and review on evolving data streams and concept drift adapting. *Evolving systems*, *9*(1), 1–23 (cit. on p. 34).

Kifer, D., Ben-David, S., & Gehrke, J. (2004). Detecting change in data streams. *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, *4*, 180–191 (cit. on p. 24).

Klinkenberg, R. (2005). Meta-learning, model selection, and example selection in machine learning domains with concept drift. *FGML* (cit. on pp. 112, 120).

Knight, W. (2020). Even the best AI models are no match for the coronavirus. Retrieved December 12, 2020, from https://www.wired.com/story/best-ai-models-no-match-coronavirus/. (Cit. on p. 7)

Kocheturov, A., Pardalos, P. M., & Karakitsiou, A. (2019). Massive datasets and machine learning for computational biomedicine: Trends and challenges. *Annals of Operations Research*, *276*(1), 5–34 (cit. on pp. 70, 71).

Koehn, D., Lessmann, S., & Schaal, M. (2020). Predicting online shopping behaviour from clickstream data using deep learning. *Expert Systems with Applications*, *150* (cit. on p. 19).

Koopman, P., & Wagner, M. (2017). Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine*, *9*(1), 90–96 (cit. on pp. 72, 73).

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. *Informatica*, *31*, 249–268 (cit. on pp. 51, 52, 56, 109, 183).

Kozlovskiy, I., Sodenkamp, M. A., Hopf, K., & Staake, T. (2016). Energy informatics for environmental, economic and societal sustainability: A case of the large-scale detection of households with old heating systems. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 63).

Krawczyk, B. (2017). Active and adaptive ensemble learning for online activity recognition from data streams. *Knowledge-Based Systems*, *138*, 69–78 (cit. on p. 112).

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, *37*, 132–156 (cit. on pp. 10, 21, 35–37, 120, 165, 168).

Kremser, T., Radszuwill, S., Schweizer, A., & Steffek, B. (2019). How do large stakes influence bitcoin performance? Evidence from the Mt. Gox liquidation case. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on pp. 33, 145).

Krippendorff, K. (2004). *Content analysis: An introduction to its methodology (second edition)*. Sage Publications. (Cit. on pp. 74, 75).

Kubat, M., Holte, R. C., & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, *30*(2), 195–215 (cit. on p. 114).

Kühl, N., Goutier, M., Hirt, R., & Satzger, G. (2019). Machine learning in artificial intelligence: Towards a common understanding. *Proceedings of the 52nd Hawaii International Conference on System Sciences* (cit. on pp. 17, 19, 91).

Kukar, M. (2003). Drifting concepts as hidden factors in clinical studies. In M. Dojat, E. T. Keravnou, & P. Barahona (Eds.), *Artificial Intelligence in Medicine* (pp. 355–364). Springer Berlin Heidelberg. (Cit. on p. 112).

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86 (cit. on p. 31).

Kuncheva, L., & Žliobaite, I. (2009). On the window size for classification in changing environments. *Intelligent Data Analysis*, *13*(6), 861–872 (cit. on p. 128).

Kuncheva, L. I. (2004). Classifier ensembles for changing environments. In F. Roli, J. Kittler, & T. Windeatt (Eds.), *Multiple Classifier Systems*. Springer Berlin Heidelberg. (Cit. on p. 35).

Kurgan, L., & Musilek, P. (2006). A survey of knowledge discovery and data mining process models. *The Knowledge Engineering Review*, *21*(1), 1 (cit. on p. 47).

Kurlej, B., & Wozniak, M. (2011). Active learning approach to concept drift problem. *Logic Journal of IGPL*, *20*(3) (cit. on pp. 112, 120).

Kurlej, B., & Woźniak, M. (2011). Learning curve in concept drift while using active learning paradigm. In A. Bouchachia (Ed.), *Adaptive and intelligent systems* (pp. 98–106). Springer Berlin Heidelberg. (Cit. on p. 112).

Kuusisto, F., Dutra, I., Elezaby, M., Mendonça, E. A., Shavlik, J., & Burnside, E. S. (2015). Leveraging expert knowledge to improve machine-learned decision support systems. *AMIA Summits on Translational Science Proceedings* (cit. on p. 122).

Laghmari, K., Marsala, C., & Ramdani, M. (2018). An adapted incremental graded multi-label classification model for recommendation systems. *Progress in Artificial Intelligence*, *7*(1), 15–29 (cit. on p. 112).

Lai, T. L. (1995). Sequential changepoint detection in quality control and dynamical systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, *57*(4), 613–644 (cit. on p. 33).

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Proceedings of the Conference on Neural Information Processing Systems* (cit. on pp. 167, 178).

Laptev, N., Yosinski, J., Li, L. E., & Smyl, S. (2017). Time-series extreme event forecasting with neural networks at uber. *International Conference on Machine Learning* (cit. on p. 146).

Lash, M. T., & Zhao, K. (2016). Early predictions of movie success: The who, what, and when of profitability. *Journal of Management Information Systems*, *33*(3), 874–903 (cit. on p. 63).

Lebanon, G., & Zhao, Y. (2008). Local likelihood modeling of temporal text streams. *International Conference on Machine Learning*, 552–559 (cit. on pp. 112, 113).

Lee, P. (2016). Learning from Tay's introduction. Retrieved December 12, 2020, from https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/#sm.00000gjdpwwcfcus11t6oo6dw79gw. (Cit. on p. 7)

Lennerholt, C., van Laere, J., & Söderström, E. (2019). Data access and data quality challenges of self-service business intelligence. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 143).

Leung, M. K., Delong, A., Alipanahi, B., & Frey, B. J. (2016). Machine learning in genomic medicine: A review of computational problems and data sets. *Proceedings of the IEEE*, *104*(1), 176–197 (cit. on p. 73).

Li, L., Goethals, F., Giangreco, A., & Baesens, B. (2013). Using social network data to predict technology acceptance. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 63).

Li, W., Chen, H., & Nunamaker Jr, J. F. (2016). Identifying and profiling key sellers in cyber carding community: AZSecure text mining system. *Journal of Management Information Systems*, *33*(4), 1059–1086 (cit. on p. 45).

Liao, L., Patterson, D. J., Fox, D., & Kautz, H. (2007). Learning and inferring transportation routines. *Artificial Intelligence*, *171*(5-6), 311–331 (cit. on p. 112).

Liao, S., Zhou, L., Di, X., Yuan, B., & Xiong, J. (2018). Large-scale short-term urban taxi demand forecasting using deep learning. *Asia and South Pacific Design Automation Conference (ASP-DAC)*, *22*, 428–433 (cit. on pp. 136, 141, 146, 148).

Lin, J., Keogh, E., Lonardi, S., & Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* (cit. on p. 23).

Lindstrom, P., Mac Namee, B., & Delany, S. J. (2013). Drift detection using uncertainty distribution divergence. *Evolving Systems*, *4*(1), 13–25 (cit. on pp. 4, 10, 36, 37, 166).

Lipton, Z. C. (2018). The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, *16*(3), 31–57 (cit. on pp. 8, 20).

Lopes, N., & Ribeiro, B. (2017). Novel trends in scaling up machine learning algorithms. *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 632–636 (cit. on pp. 70–72).

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346–2363 (cit. on pp. 4, 9, 10, 30, 35–38, 165).

Luo, J., Pronobis, A., Caputo, B., & Jensfelt, P. (2007). Incremental learning for place recognition in dynamic environments. *IEEE/RSJ International Conference*, 721–728 (cit. on p. 112).

Lüttenberg, H., Bartelheimer, C., & Beverungen, D. (2018). Designing predictive maintenance for agricultural machines. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on pp. 5, 20).

Maaradji, A., Dumas, M., La Rosa, M., & Ostovar, A. (2017). Detecting sudden and gradual drifts in business processes from execution traces. *IEEE Transactions on Knowledge and Data Engineering*, *29*(10), 2140–2154 (cit. on p. 95).

Maass, W., Parsons, J., Purao, S., Storey, V. C., & Woo, C. (2018). Data-driven meets theory-driven research in the era of big data: Opportunities and challenges for information systems research. *Journal of the Association for Information Systems*, *19*(12), 1 (cit. on p. 18).

Machado, N. L., & Ruiz, D. D. (2017). Customer: A novel customer churn prediction method based on mobile application usage. *International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2146–2151 (cit. on p. 38).

Maedche, A., Legner, C., Benlian, A., Berger, B., Gimpel, H., Hess, T., Hinz, O., Morana, S., & Söllner, M. (2019). AI-based digital assistants. *Business & Information Systems Engineering*, *61*(4), 535–544 (cit. on pp. 5, 18).

Malikopoulos, A. A., Papalambros, P. Y., & Assanis, D. N. (2007). A learning algorithm for optimal internal combustion engine calibration in real time. *ASME International Design Engineering Technical Conferences*, 91–100 (cit. on p. 185).

Malle, B., Kieseberg, P., & Holzinger, A. (2017). Do not disturb? Classifier behavior on perturbed datasets. *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, 155–173 (cit. on p. 71).

Manning, C. D., & Schütze, H. (2000). *Foundations of Natural Language Processing*. MIT Press. (Cit. on p. 51).

Manoj Kumar, M. V., Thomas, L., & Annappa, B. (2015). Capturing the sudden concept drift in process mining. *CEUR Workshop Proceedings* (cit. on pp. 94, 95).

Márquez-Chamorro, A. E., Resinas, M., & Ruiz-Cortes, A. (2017). Predictive monitoring of business processes: A survey. *IEEE Transactions on Services Computing*, *11*(6), 962–977 (cit. on p. 94).

Martens, D., & Provost, F. (2014). Explaining data-driven document classifications. *MIS Quarterly*, *38*(1), 73–100 (cit. on p. 45).

Masud, M. M., Chen, Q., Khan, L., Aggarwal, C. C., Gao, J., Han, J., Srivastava, A., & Oza, N. C. (2013). Classification and adaptive novel class detection of feature-evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, *25*(7), 1484–1497 (cit. on p. 24).

Masud, M. M., Woolam, C., Gao, J., Khan, L., Han, J., et al. (2012). Facing the reality of data stream classification: Coping with scarcity of labeled data. *Knowledge and Information Systems*, *33*(1), 213–244 (cit. on p. 165).

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A proposal for the Dartmouth summer research project on artificial intelligence, August 31, 1955. *AI Magazine*, *27*(4), 12–12 (cit. on p. 17).

Mera, C., Orozco-Alzate, M., & Branch, J. (2019). Incremental learning of concept drift in multiple instance learning for industrial visual inspection. *Computers in Industry*, *109*, 153–164 (cit. on p. 38).

Meyer, G., Adomavicius, G., Johnson, P. E., Elidrisi, M., Rush, W. A., Sperl-Hillen, J. M., & O'Connor, P. J. (2014). A machine learning approach to improving dynamic decision making. *Information Systems Research*, *25*(2), 239–263 (cit. on p. 18).

Microsoft. (2020). Microsoft Team Data Science Process Documentation. Retrieved August 5, 2020, from https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/overview. (Cit. on pp. 46, 57, 215)

Microsoft. (2021). Collect data from models in production. Retrieved January 15, 2021, from https://docs.microsoft.com/en-us/azure/machine-learning/how-to-enable-data-collection. (Cit. on p. 216)

Minku, L. L., White, A. P., & Yao, X. (2009). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, *22*(5), 730–742 (cit. on p. 35).

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I. D., & Gebru, T. (2019). Model cards for model reporting. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 220–229 (cit. on p. 47).

Mittal, V., & Kashyap, I. (2018). An overview of real world applications with concept drifting data streams. *Proceedings of the International Conference on Internet of Things and Connected Technologies (ICIoTCT)* (cit. on p. 147).

Mohamad, S., Bouchachia, A., & Sayed-Mouchaweh, M. (2016). A bi-criteria active learning algorithm for dynamic data streams. *IEEE Transactions on Neural Networks and Learning Systems* (cit. on p. 112).

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2013). Foundations of machine learning. *Journal of Chemical Information and Modeling*, *53*(9), 1689–1699 (cit. on pp. 44, 45).

Mongan, J., Moy, L., & Kahn Jr, C. E. (2020). Checklist for Artificial Intelligence in Medical Imaging (CLAIM): A guide for authors and reviewers. *Radiology: Artificial Intelligence*, *2*(2) (cit. on pp. 47, 49).

Monroe, D. (2018). Chips for artificial intelligence. *Communincations of the ACM*, *61*(4) (cit. on p. 3).

Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, *19*(1) (cit. on pp. 99, 170, 172).

Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., & Damas, L. (2013). Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, *14*(3), 1393–1402 (cit. on p. 146).

Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., & Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recognition*, *45*(1), 521–530 (cit. on pp. 27, 164).

Moroshko, E., Vaits, N., & Crammer, K. (2015). Second-order non-stationary online learning for regression. *Journal of Machine Learning Research*, *16*, 1481–1517 (cit. on p. 25).

Mourão, F., Rocha, L., Araújo, R., Couto, T., Gonçalves, M., & Meira, W. (2008). Understanding temporal aspects in document classification. *Proceedings of the International Conference on Web Search and Data Mining* (cit. on p. 112).

Muthukrishnan, S. (2005). *Data streams: Algorithms and applications*. Now Publishers Inc. (Cit. on p. 23).

Nascimento, A. M., Meirelles, F. d. S., da Cunha, M. A. V., Scornavacca, E., & de Melo, V. V. (2018). A literature analysis of research on artificial intelligence in management information system (MIS). *Proceedings of the American Conference on Information Systems (AMCIS)* (cit. on pp. 17, 19).

Neyman, J. (1934). On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, *97*(4), 558–625 (cit. on p. 53).

Nguyen, H.-L., Woon, Y.-K., & Ng, W.-K. (2015). A survey on data stream clustering and classification. *Knowledge and Information Systems*, *45*(3), 535–569 (cit. on p. 37).

Nielsen, B., & Whitby, A. (2015). A joint chow test for structural instability. *Econometrics*, *3*(1), 156–186 (cit. on p. 32).

Nishida, K., & Yamauchi, K. (2007). Detecting concept drift using statistical testing. *International conference on discovery science*, 264–269 (cit. on p. 145).

Nunes, I., & Jannach, D. (2017). A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction*, *27*(3-5), 393–444 (cit. on p. 73).

Oh, C., & Sheng, O. (2011). Investigating predictive power of stock micro blog sentiment in forecasting future stock price directional movement. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 61).

Öksüz, N., Shcherbatyi, I., Kowatsch, T., & Maass, W. (2018). A data-analytical system to predict therapy success for obese children. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 20).

Olbrich, S., Frank, U., Gregor, S., Niederman, F., & Rowe, F. (2017). On the merits and limits of replication and negation for IS research. *AIS Transactions on Replication Research*, *3*(1), 1 (cit. on p. 43).

Olorisade, B. K., Brereton, P., & Andras, P. (2017). Reproducibility in machine learning-based studies: An example of text mining. *Reproducibility in Machine Learning Workshop, International Conference on Machine Learning* (cit. on p. 47).

Oneto, L., Ghio, A., Ridella, S., & Anguita, D. (2015). Learning resource-aware classifiers for mobile devices: From regularization to energy efficiency. *Neurocomputing*, *169*, 225–235 (cit. on pp. 128, 181, 183).

OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., ... Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. https://arxiv.org/abs/1912.06680 (cit. on p. 3)

Oquendo, M. A., Baca-Garcia, E., Artes-Rodriguez, A., Perez-Cruz, F., Galfalvy, H. C., Blasco-Fontecilla, H., Madigan, D., & Duan, N. (2012). Machine learning and data mining: strategies for hypothesis generation. *Molecular psychiatry*, *17*(10), 956 (cit. on p. 50).

Orair, G. H., Teixeira, C. H., Meira Jr, W., Wang, Y., & Parthasarathy, S. (2010). Distance-based outlier detection: Consolidation and renewed bearing. *Proceedings of the VLDB Endowment*, *3*(1-2), 1469–1480 (cit. on p. 185).

Oroszi, F., & Ruhland, J. (2010). An early warning system for hospital acquired pneumonia. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 45).

Ostovar, A., Leemans, S. J., & Rosa, M. L. (2020). Robust drift characterization from event streams of business processes. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *14*(3), 1–57 (cit. on p. 95).

Ostrom, A. L., Parasuraman, A., Bowen, D. E., Patrício, L., & Voss, C. A. (2015). Service Research Priorities in a Rapidly Changing Context. *Journal of Service Research*, *18*(2), 127–159 (cit. on p. 91).

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., et al. (2019). Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *Proceedings of the Conference on Neural Information Processing Systems* (cit. on p. 167).

Pacheco, F., Exposito, E., Gineste, M., Baudoin, C., & Aguilar, J. (2019). Towards the deployment of machine learning solutions in network traffic classification: A systematic survey. *IEEE Communications Surveys Tutorials*, *21*(2), 1988–2014 (cit. on p. 8).

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, *41*(1/2), 100–115 (cit. on pp. 30, 94, 129, 165).

Pant, G., & Srinivasan, P. (2010). Predicting web page status. *Information Systems Research*, *21*(2), 345–364 (cit. on pp. 45, 61).

Pappas, I. O., Mikalef, P., Giannakos, M. N., Krogstie, J., & Lekakos, G. (2018). Big data and business analytics ecosystems: Paving the way towards digital transformation and sustainable societies. *Information Systems and e-Business Management*, *16*(3), 479–491 (cit. on p. 18).

Parker, C. (2012). Unexpected challenges in large scale machine learning. *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications* (cit. on p. 72).

Pawling, A., Chawla, N. V., & Madey, G. (2007). Anomaly detection in a mobile communication network. *Computational and Mathematical Organization Theory*, *13*(4), 407–422 (cit. on pp. 110, 112, 120).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830 (cit. on pp. 51, 58, 102, 188, 194).

Peng, R. D. (2011). Reproducible research in computational science. *Science*, *334*(6060), 1226–1227 (cit. on p. 46).

Perdomo, J. C., Zrnic, T., Mendler-Dünner, C., & Hardt, M. (2020). Performative prediction. *International Conference on Machine Learning* (cit. on p. 217).

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710 (cit. on p. 26).

Perron, P. (1989). The great crash, the oil price shock, and the unit root hypothesis. *Econometrica*, *57*(6), 1361–1401 (cit. on pp. 33, 130).

Perry, W. L. (2013). *Predictive policing: The role of crime forecasting in law enforcement operations*. Rand Corporation. (Cit. on p. 111).

Pfahringer, B., Holmes, G., & Kirkby, R. (2007). New options for Hoeffding trees. In M. A. Orgun & J. Thornton (Eds.), *Ai 2007: Advances in artificial intelligence*. Springer Berlin Heidelberg. (Cit. on p. 97).

Pineau, J. (2020). The machine learning reproducibility checklist. Retrieved August 5, 2020, from http://www.cs.mcgill.ca/%7B~%7Djpineau/ReproducibilityChecklist-v2.0.pdf. (Cit. on p. 47)

Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2017). Data management challenges in production machine learning. *Proceedings of the ACM International Conference on Management of Data*, 1723–1726 (cit. on pp. 71, 72).

Popovič, A., Hackney, R., Tassabehji, R., & Castelli, M. (2018). The impact of big data analytics on firms' high value business performance. *Information Systems Frontiers*, *20*(2), 209–222 (cit. on p. 18).

Powers, D. (2011). Evaluation: From precision, recall and F-Measure To ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63 (cit. on p. 52).

Prado, M. D., Su, J., Saeed, R., Keller, L., Vallez, N., Anderson, A., Gregg, D., Benini, L., Llewellynn, T., Ouerhani, N., Dahyot, R., & Pazos, N. (2020). Bonseyes AI pipeline—Bringing AI to you: End-to-end integration of data, algorithms, and deployment Tools. *ACM Transactions on Internet Things*, *1*(4) (cit. on p. 3).

Ptaszynski, M., Lempa, P., Masui, F., Kimura, Y., Rzepka, R., Araki, K., Wroczynski, M., & Leliwa, G. (2019). Brute-force sentence pattern extortion from harmful messages for cyberbullying detection. *Journal of the Association for Information Systems*, *20*(8), 4 (cit. on p. 19).

Pumplun, L., Tauchert, C., & Heidt, M. (2019). A new organizational chassis for artificial intelligence-exploring organizational readiness factors. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on pp. 4, 18).

Qahtan, A. A., Alharbi, B., Wang, S., & Zhang, X. (2015). A PCA-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 935–944 (cit. on pp. 30, 31).

Qiao, N. (2019). A systematic review on machine learning in sellar region diseases: quality and reporting items. *Endocrine connections*, *8*(7), 952–960 (cit. on pp. 47, 50).

Quanrud, K., & Khashabi, D. (2015). Online learning with adversarial delays. *Proceedings of the Conference on Neural Information Processing Systems*, 1270–1278 (cit. on p. 25).

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2009). *Dataset shift in machine learning*. The MIT Press. (Cit. on pp. 27, 164).

Raab, C., Heusinger, M., & Schleif, F.-M. (2020). Reactive soft prototype computing for concept drift streams. *Neurocomputing*, *416*, 340–351 (cit. on pp. 30, 31, 165, 168, 171).

Rädsch, T., Eckhardt, S., Leiser, F., Pandl, K. D., Thiebes, S., & Sunyaev, A. (2021). What your radiologist might be missing: Using machine learning to identify mislabeled instances of X-ray images. *Proceedings of the 54th Hawaii International Conference on System Sciences (HICSS)* (cit. on p. 20).

Rahman, M. M., & Davis, D. N. (2013). Addressing the class imbalance problem in medical datasets. *International Journal of Machine Learning and Computing*, *3*(2), 224–228 (cit. on p. 51).

Ramamurthy, S., & Bhatnagar, R. (2007). Tracking recurrent concept drift in streaming data using ensemble classifiers. *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 404–409 (cit. on pp. 29, 37).

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., & Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, *239*, 39–57 (cit. on pp. 21, 23).

Rätsch, G. (2004). A brief introduction into machine learning. *Chaos Communication Congress* (cit. on p. 45).

Raykar, V. C., Yu, S., Zhao, L. H., Jerebko, A., Florin, C., Valadez, G. H., Bogoni, L., & Moy, L. (2009). Supervised learning from multiple experts: Whom to trust when everyone lies a bit. *International Conference on Machine Learning*, 889–896 (cit. on p. 181).

Raza, H., Prasad, G., & Li, Y. (2015). EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments. *Pattern Recognition*, *48*(3), 659–669 (cit. on pp. 30, 32).

Renggli, C., Rimanic, L., Gürel, N. M., Karlaš, B., Wu, W., & Zhang, C. (2021). A data quality-driven view of MLOps. *arXiv preprint arXiv:2102.07750* (cit. on p. 215).

Riekert, M., Leukel, J., & Klein, A. (2016). Online media sentiment: Understanding machine learning-based classifiers. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 63).

Riekert, M., Premm, M., Klein, A., Lyubomir Kirilov, Kenngott, H., Apitz, M., Wagner, M., & Ternes, L. (2017). Predicting the duration of surgeries to improve process efficiency in hospitals. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 61).

Rock, D. (2020). How managers can enable AI talent in organizations. *MIT Sloan Management Review*. https://sloanreview.mit.edu/article/how-managers-can-enable-ai-talent-in-organizations/ (cit. on p. 3)

Rosemann, M., & vom Brocke, J. (2015). The six core elements of business process management. In J. vom Brocke & M. Rosemann (Eds.), *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems* (pp. 105–122). Springer Berlin Heidelberg. (Cit. on p. 92).

Roughgarden, T., & Schrijvers, O. (2017). Online prediction with selfish experts. *Advances in Neural Information Processing Systems*, 1300–1310 (cit. on p. 26).

Roy, A. G., Conjeti, S., Navab, N., & Wachinger, C. (2018). Inherent brain segmentation quality control from fully ConvNet Monte Carlo sampling. In A. F. Frangi, J. A. Schnabel, C. Davatzikos, C. Alberola-López, & G. Fichtinger (Eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI*. Springer International Publishing. (Cit. on p. 168).

Ruano-Ordas, D., Fdez-Riverola, F., & Mendez, J. R. (2018). Concept drift in e-mail datasets: An empirical study with practical implications. *Information Sciences*, *428*, 120–135 (cit. on p. 38).

Rudin, C., & Wagstaff, K. (2014). Machine learning for science and society. *Machine Learning*, *95*(1), 1–9 (cit. on pp. 67, 68, 73, 87).

Rupp, M., Gobre, V., Vazquez-mayagoitia, A., Tkatchenko, A., & Lilienfeld, O. A. V. (2013). Machine learning of molecular electronic properties in chemical compound space. *New Journal of Physics*, *15*, 1–9 (cit. on p. 49).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252 (cit. on p. 3).

Russell, S., Dewey, D., & Tegmark, M. (2015). Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, *36*(4), 105–114 (cit. on pp. 108, 117).

Rutkowski, L., Jaworski, M., & Duda, P. (2020). Basic concepts of data stream mining. *Stream Data Mining: Algorithms and Their Probabilistic Properties* (pp. 13–33). Springer International Publishing. (Cit. on pp. 21, 23, 24).

Rzepka, C., & Berger, B. (2018). User interaction with AI-enabled systems: A systematic review of IS research. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 19).

Saadallah, A., Moreira-Matias, L., Sousa, R., Khiari, J., Jenelius, E., & Gama, J. (2020). BRIGHT—Drift-aware demand predictions for taxi networks. *IEEE Transactions on Knowledge and Data Engineering*, *32*(2), 234–245 (cit. on p. 38).

Saidulu, D., & Sasikala, R. (2017). Machine learning and statistical approaches for big data: Issues, challenges and research directions. *International Journal of Applied Engineering Research*, *12*(21), 11691–11699 (cit. on pp. 70–72).

Sarwate, A. D., & Chaudhuri, K. (2013). Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *IEEE Signal Processing Magazine*, *30*(5), 86–94 (cit. on pp. 70, 71, 73).

Schelter, S., Biessmann, F., Januschowski, T., Salinas, D., Seufert, S., & Szarvas, G. (2018). On challenges in machine learning model management. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* (cit. on pp. 4, 6, 8).

Schlimmer, J. C., & Granger, R. H. (1986). Incremental learning from noisy data. *Machine learning*, *1*(3), 317–354 (cit. on p. 37).

Schooler, J. W. (2014). Metascience could rescue the 'replication crisis'. *Nature*, *515*(7525), 9 (cit. on p. 43).

Schüritz, R., & Satzger, G. (2016). Patterns of data-infused business model innovation. *Proceedings of the IEEE Conference on Business Informatics (CBI)* (cit. on pp. 5, 67, 91, 107, 113, 127, 181).

Schüritz, R., Seebacher, S., Satzger, G., & Schwarz, L. (2017). Datatization as the next frontier of servitization: Understanding the challenges for transforming organizations. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on pp. 4, 143).

Schwaiger, J., Lang, M., Johannsen, F., & Leist, S. (2017). What does the customer want to tell us? An automated classification approach for social media posts at small and medium-sized enterprises. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 64).

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, 2503–2511 (cit. on p. 71).

Seeber, I., Bittner, E., Briggs, R. O., de Vreede, T., de Vreede, G.-J., Elkins, A., Maier, R., Merz, A. B., Oeste-Reiß, S., Randrup, N., Schwabe, G., & Söllner, M. (2020). Machines as teammates: A research agenda on AI in team collaboration. *Information & Management*, *57*(2) (cit. on p. 18).

Seeliger, A., Nolle, T., & Mühlhäuser, M. (2017). Detecting concept drift in processes using graph metrics on process graphs. *Proceedings of the Conference on Subject-Oriented Business Process Management*, 1–10 (cit. on p. 38).

Sensoy, M., Kaplan, L., & Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. *Proceedings of the International Conference on Neural Information Processing Systems* (cit. on p. 167).

Sethi, T. S., & Kantardzic, M. (2015). Don't pay for validation: Detecting drifts from unlabeled data using margin density. *Procedia Computer Science*, *53*, 103–112 (cit. on p. 166).

Sethi, T. S., & Kantardzic, M. (2017). On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, *82*, 77–99 (cit. on pp. 36, 37).

Shafique, M., Hafiz, R., Javed, M. U., Abbas, S., Sekanina, L., Vasicek, Z., & Mrazek, V. (2017). Adaptive and energy-efficient architectures for machine learning: Challenges, opportunities, and research roadmap. *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 627–632 (cit. on pp. 71–73).

Shalev-Shwartz, S. (2011). Online learning and online convex optimization. *Foundations and Trends in Machine Learning* (cit. on pp. 24, 25, 163).

Sharp, J., & Babb, J. (2018). Is Information Systems late to the party? The current state of DevOps research in the Association for Information Systems eLibrary. *Proceedings of the Americas Conference on Information Systems (AMCIS)* (cit. on p. 46).

Shea, C., Page, A., & Mohsenin, T. (2018). SCALENet: A scalable low power accelerator for real-time embedded deep neural networks. *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 129–134 (cit. on p. 72).

Sheu, J.-J., Chu, K.-T., Li, N.-F., & Lee, C.-C. (2017). An efficient incremental learning mechanism for tracking concept drift in spam filtering. *PloS one*, *12*(2) (cit. on p. 38).

Shipp, M. A., Ross, K. N., Tamayo, P., Weng, A. P., Kutok, J. L., Aguiar, R. C., Gaasenbeek, M., Angelo, M., Reich, M., Pinkus, G. S., Ray, T. S., Koval, M. A., Last, K. W., Norton, A., Lister, T. A., Mesirov, J., Neuberg, D. S., Lander, E. S., Aster, J. C., & Golub, T. R. (2002). Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature Medicine*, *8*(1), 68–74 (cit. on p. 49).

Shmueli, & Koppius. (2011). Predictive analytics in information systems research. *MIS Quarterly*, *35*(3), 553–572 (cit. on pp. 18, 45, 47, 49, 143).

Silver, D. L. (2011). Machine lifelong learning: Challenges and benefits for artificial general intelligence. *International Conference on Artificial General Intelligence*, 370–375 (cit. on p. 71).

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359 (cit. on p. 3).

Soares, S. G., & Araújo, R. (2015). A dynamic and on-line ensemble regression for changing environments. *Expert Systems with Applications*, *42*(6), 2935–2948 (cit. on p. 130).

Soares, S. G., & Araújo, R. (2016). An adaptive ensemble of online extreme learning machines with variable forgetting factor for dynamic system prediction. *Neurocomputing*, *171*, 693–707 (cit. on p. 112).

Sobolewski, P., & Wozniak, M. (2013). Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors. *Journal of Universal Computer Science*, *19*(4), 462–483 (cit. on p. 37).

Söllner, M., Hoffmann, A., & Leimeister, J. M. (2016). Why different trust relationships matter for information systems users. *European Journal of Information Systems*, *25*(3), 274–287 (cit. on p. 143).

Somasundaram, A., & Reddy, S. (2019). Parallel and incremental credit card fraud detection model to handle concept drift and data imbalance. *Neural Computing and Applications*, *31*(1), 3–14 (cit. on p. 38).

Sonali, P., & Kumar, D. N. (2013). Review of trend detection methods and their application to detect temperature changes in India. *Journal of Hydrology*, *476*, 212–227 (cit. on pp. 33, 146, 152).

Song, Y., Lu, J., Lu, H., & Zhang, G. (2019). Fuzzy clustering-based adaptive regression for drifting data streams. *IEEE Transactions on Fuzzy Systems*, *28*(3), 544–557 (cit. on pp. 35, 36).

Souza, V. M. A., dos Reis, D. M., Maletzke, A., & Batista, G. E. A. P. A. (2020). Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, *34*(6), 1805–1858 (cit. on pp. 36, 37, 169, 173, 217).

Spangler, W. E., Chung, H. M., & Gey, F. C. (2000). Data mining: A brief introduction to the field and research community. *Proceedings of the Americas Conference on Information Systems (AMCIS)* (cit. on pp. 73, 82).

Spuler, M., Sarasola-Sanz, A., Birbaumer, N., Rosenstiel, W., & Ramos-Murguialday, A. (2015). Comparing metrics to evaluate performance of regression methods for decoding of neural signals. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 1083–1086 (cit. on p. 52).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1) (cit. on p. 167).

Stange, M., & Funk, B. (2015). How much tracking is necessary - The learning curve in Bayesian user journey analysis. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 63).

Staples, M., Zhu, L., & Grundy, J. (2016). Continuous validation for data analytics systems. *Proceedings of the International Conference on Software Engineering Companion*, 769–772 (cit. on pp. 71, 72).

Staudt, P., Träris, Y., Rausch, B., & Weinhardt, C. (2018). Predicting redispatch in the German electricity market using information systems based on machine learning. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 20).

Stefanou, C. J. (2001). A framework for the ex-ante evaluation of ERP software. *European Journal of Information Systems*, *10*(4), 204–215 (cit. on p. 6).

Stein, N., Flath, C., & Boehm, C. (2018). Predictive analytics for application management services. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on pp. 19, 146).

Steins, K., Matinrad, N., & Granberg, T. (2019). Forecasting the demand for emergency medical services. *Proceedings of the 52nd Hawaii International Conference on System Sciences* (cit. on p. 146).

Steuer, D., Hutterer, V., Korevaar, P., & Fromm, H. (2018). A similarity-based approach for the all-time demand prediction of new automotive spare parts. *Proceedings of the 51st Hawaii International Conference on System Sciences* (cit. on p. 146).

Stowers, K., Kasdaglis, N., Newton, O., Lakhmani, S., Wohleber, R., & Chen, J. (2016). Intelligent agent transparency: The design and evaluation of an interface to facilitate human and intelligent agent collaboration. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *60*(1), 1706–1710 (cit. on p. 134).

Street, W. N., & Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 377–382 (cit. on p. 37).

Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Mueller, K.-R. (2020). Towards CRISP-ML (Q): A machine learning process model with quality assurance methodology. *arXiv preprint arXiv:2003.05155* (cit. on pp. 47, 64).

Sturm, B., & Sunyaev, A. (2019). A good beginning makes a good ending: Incipient sources of knowledge in design science research. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 8).

Sun, J., Fujita, H., Chen, P., & Li, H. (2017). Dynamic financial distress prediction with concept drift based on time weighting combined with Adaboost support vector machine ensemble. *Knowledge-Based Systems*, *120*, 4–14 (cit. on p. 130).

Sun, Y., Tang, K., Minku, L. L., Wang, S., & Yao, X. (2016). Online ensemble learning of data streams with gradually evolved classes. *IEEE Transactions on Knowledge and Data Engineering* (cit. on p. 112).

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., et al. (2020). Test-time training with self-supervision for generalization under distribution shifts. *International Conference on Machine Learning* (cit. on p. 166).

Sunyaev, A. (2020a). Design of good information systems architectures. *Internet computing: Principles of distributed systems and emerging internet-based technologies* (pp. 51–81). Springer International Publishing. (Cit. on p. 6).

Sunyaev, A. (2020b). Emerging technologies. *Internet computing: Principles of distributed systems and emerging internet-based technologies* (pp. 373–406). Springer International Publishing. (Cit. on p. 3).

Suthaharan, S. (2014). Big data classification: Problems and challenges in network intrusion prediction with machine learning. *ACM SIGMETRICS Performance Evaluation Review*, *41*(4), 70–73 (cit. on p. 71).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (cit. on p. 163).

Tackett, J. L., Brandes, C. M., King, K. M., & Markon, K. E. (2019). Psychology's replication crisis and clinical psychological science. *Annual review of clinical psychology*, *15*, 579–604 (cit. on p. 43).

Tatbul, N., Çetintemel, U., Zdonik, S., Cherniack, M., & Stonebraker, M. (2003). Load shedding in a data stream manager. *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 309–320 (cit. on p. 23).

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *IEEE symposium on computational intelligence for security and defense applications* (cit. on p. 37).

Terveen, L. G. (1995). Overview of human-computer collaboration. *Knowledge-Based Systems*, *8*(2-3), 67–81 (cit. on p. 18).

Thiebes, S., Lins, S., & Sunyaev, A. (2020). Trustworthy artificial intelligence. *Electronic Markets* (cit. on p. 19).

Thrall, J. H., Li, X., Li, Q., Cruz, C., Do, S., Dreyer, K., & Brink, J. (2018). Artificial intelligence and machine learning in radiology: Opportunities, challenges, pitfalls, and criteria for success. *Journal of the American College of Radiology*, *15*(3), 504–508 (cit. on p. 3).

Timmerman, Y., & Bronselaer, A. (2019). Measuring data quality in information systems research. *Decision Support Systems*, *126* (cit. on p. 46).

TLC. (2019). Taxi and Limousine Commission (TLC) trip record data. Retrieved April 4, 2019, from https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page. (Cit. on pp. 132, 144, 147)

Tomin, N., Zhukov, A., Sidorov, D., Kurbatsky, V., Panasetsky, D., & Spiryaev, V. (2015). Random forest based model for preventing large-scale emergencies in power systems. *International Journal of Artificial Intelligence*, *13*(1), 211–228 (cit. on p. 38).

Tripathi, M., & Kaur, I. (2018). Oil prices forecasting: A comparative analysis. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 20).

Tsymbal, A. (2004). *The problem of concept drift: Definitions and related work* (tech. rep.). Computer Science Department, Trinity College Dublin. (Cit. on pp. 26, 29, 72, 91, 93, 107, 108, 110, 128, 133, 143, 145, 165, 182).

Tsymbal, A., Pechenizkiy, M., Cunningham, P., & Puuronen, S. (2008). Dynamic integration of classifiers for handling concept drift. *Information Fusion*, *9*(1), 56–68 (cit. on p. 112).

Tušar, T., Gantar, K., Koblar, V., Ženko, B., & Filipič, B. (2017). A study of overfitting in optimization of a manufacturing quality control procedure. *Applied Soft Computing*, *59*, 77–87 (cit. on p. 53).

Urbanke, P., Uhlig, A., & Kranz, J. (2017). A customized and interpretable deep neural network for high-dimensional business data - Evidence from an e-commerce application. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 63).

Van Der Aalst, W. (2011). *Process Mining: Discovery, Conformance and Enhancement of Business Processes* (Vol. 2). Springer. (Cit. on p. 94).

Van Der Aalst, W., La Rosa, M., & Santoro, F. M. (2016). Business process management. *Business & Information Systems Engineering*, *58*(1) (cit. on p. 92).

Van Der Aalst, W., Reijers, H. A., Weijters, A. J. M. M., van Dongen, B. F., De Medeiros, A. K. A., Song, M., & Verbeek, H. M. W. (2007). Business process mining: An industrial application. *Information Systems*, *32*(5), 713–732 (cit. on p. 94).

Van Der Aalst, W., Schonenberg, M. H., & Song, M. (2011). Time prediction based on process mining. *Information Systems*, *36*(2), 450–475 (cit. on pp. 94, 95).

Van Der Aalst, W., & Weijters, A. (2004). Process mining: A research agenda. *Computers in Industry*, *53*(3), 231–244 (cit. on p. 94).

van der Aalst, W., Adriansyah, A., de Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., Bose, J. C., van den Brand, P., Brandtjen, R., Buijs, J., Burattin, A., Carmona, J., Castellanos, M., Claes, J., Cook, J., Costantini, N., Curbera, F., Damiani, E., de Leoni, M., ... Wynn, M. (2012). Process mining manifesto. In F. Daniel, K. Barkaoui, & S. Dustdar (Eds.), *Business Process Management Workshops* (pp. 169–194). Springer Berlin Heidelberg. (Cit. on pp. 92, 94, 95).

Verbesselt, J., Hyndman, R., Newnham, G., & Culvenor, D. (2010). Detecting trend and seasonal changes in satellite image time series. *Remote Sensing of Environment*, *114*(1), 106–115 (cit. on pp. 32, 130).

Verenich, I., Dumas, M., Rosa, M. L., Maggi, F. M., & Teinemaa, I. (2019). Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(4), 1–34 (cit. on p. 94).

Vermorel, J., & Mohri, M. (2005). Multi-armed bandit algorithms and empirical evaluation. *European Conference on Machine Learning*, 437–448 (cit. on p. 25).

VHB. (2012). VHB-JOURQUAL3. Retrieved June 1, 2019, from https://vhbonline.org/en/service/jourqual/vhb-jourqual-3/. (Cit. on pp. 59, 65)

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., ... Silver, D. (2019). AlphaStar: Mastering the real-time strategy game StarCraft II. Retrieved January 4, 2021, from https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/. (Cit. on p. 3)

Voets, M., Møllersen, K., & Bongo, L. A. (2018). Replication study: Development and validation of deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *arXiv preprint arXiv:1803.04337* (cit. on p. 47).

Vong, C.-M., Wong, P.-K., & Li, Y.-P. (2006). Prediction of automotive engine power and torque using least squares support vector machines and Bayesian inference. *Engineering Applications of Artificial Intelligence*, *19*(3), 277–287 (cit. on p. 185).

Vössing, M., Potthoff, F., Kühl, N., & Satzger, G. (2019). Designing useful transparency to improve process performance—Evidence from an automated production line. *Proceedings of the European Conference on Information Systems (ECIS)* (cit. on p. 218).

Vreeken, J., Van Leeuwen, M., & Siebes, A. (2007). Characterising the difference. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 765–774 (cit. on p. 37).

Wagstaff, K. (2012). Machine learning that matters. *Proceedings of the International Conference on Machine Learning*, 1851–1856 (cit. on pp. 68, 73).

Walden, E., Cogo, G. S., Lucus, D. J., Moradiabadi, E., & Safi, R. (2018). Neural correlates of multidimensional visualizations: An fMRI comparison of bubble and three-dimensional surface graphs using evolutionary theory. *MIS Quarterly*, *42*(4), 1097–1116 (cit. on p. 59).

Wang, D., Ram, P., Weidele, D. K. I., Liu, S., Muller, M., Weisz, J. D., Valente, A., Chaudhary, A., Torres, D., Samulowitz, H., et al. (2020). AutoAI: Automating the end-to-end AI lifecycle with humans-in-the-loop. *Proceedings of the International Conference on Intelligent User Interfaces Companion*, 77–78 (cit. on p. 4).

Wang, R. Y., Kon, H. B., & Madnick, S. E. (1993). Data quality requirements analysis and modeling. *Proceedings of IEEE International Conference on Data Engineering*, (April), 670–677 (cit. on p. 51).

Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., & Chan, K. (2018). When edge meets learning: Adaptive control for resource-constrained distributed machine learning. *IEEE Conference on Computer Communications*, 63–71 (cit. on p. 183).

Wang, T., Kannan, K. N., & Ulmer, J. R. (2013). The association between the disclosure and the realization of information security risk factors. *Information Systems Research*, *24*(2), 201–218 (cit. on p. 61).

Wang, W., Zhang, M., Chen, G., Jagadish, H., Ooi, B. C., & Tan, K.-L. (2016). Database meets deep learning: Challenges and opportunities. *ACM SIGMOD Record*, *45*(2), 17–22 (cit. on p. 71).

Wang, W., & Benbasat, I. (2005). Trust in and adoption of online recommendation agents. *Journal of the Association for Information Systems*, *6*(3), 72–101 (cit. on p. 143).

Wang, W., Li, B., & Luo, X. (2020). AI agents for sequential promotions: Combining deep reinforcement learning and dynamic field experimentation. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 19).

Wang, X., Chen, W., Xia, J., Chen, Z., Xu, D., Wu, X., Xu, M., & Schreck, T. (2020). ConceptExplorer: Visual analysis of concept drifts in multi-source time-series data. *Proceedings IEEE Conference on Visual Analytics Science and Technology* (cit. on p. 39).

Wanner, J., Heinrich, K., Janiesch, C., & Zschech, P. (2020). How much AI do you require? Decision factors for adopting AI technology. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 20).

Wanner, J., Herm, L.-V., Heinrich, K., Janiesch, C., & Zschech, P. (2020). White, grey, black: Effects of XAI augmentation on the confidence in AI-based decision support systems. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 20).

Webb, G., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, *30*(4), 964–994 (cit. on pp. 26, 29, 38, 93, 110, 128).

Webb, G., Lee, L. K., Goethals, B., & Petitjean, F. (2018). Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, *32*(5), 1179–1199 (cit. on pp. 38, 39).

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly*, *26*(2) (cit. on p. 69).

Weinhardt, C., Van der Aalst, W. M. P., & Hinz, O. (2019). Introducing registered reports to the information systems community. *Business & Information Systems Engineering*, *61*(4), 381–384 (cit. on p. 43).

Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. *International Conference on Machine Learning* (cit. on p. 167).

Werts, N., & Adya, M. (2000). Data mining in healthcare: Issues and a research agenda. *Proceedings of the Americas Conference on Information Systems (AMCIS)* (cit. on pp. 70, 73).

White, H., & Granger, C. W. (2011). Consideration of trends in time series. *Journal of Time Series Econometrics*, *3*(1) (cit. on p. 33).

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, *23*(1), 69–101 (cit. on pp. 4, 27, 72, 92, 93, 110, 127, 145, 164, 184).

Widyantoro, D. H., & Yen, J. (2005). Relevant data expansion for learning concept drift from sparsely labeled data. *IEEE Transactions on Knowledge and Data Engineering*, *17*(3), 401–412 (cit. on p. 112).

Wilson, M. (2017). AI is inventing languages humans can't understand. Should we stop it? Retrieved February 9, 2021, from https://www.fastcompany.com/90132632/ai-is-inventing-its-own-perfect-languages-should-we-let-it. (Cit. on p. 7)

Winkler-Schwartz, A., Bissonnette, V., Mirchi, N., Ponnudurai, N., Yilmaz, R., Ledwos, N., Siyar, S., Azarnoush, H., Karlik, B., & Del Maestro, R. F. (2019). Artificial intelligence in medical education: Best practices using machine learning to assess surgical expertise in virtual reality simulation. *Journal of Surgical Education*, *76*(6), 1681–1690 (cit. on pp. 47, 50).

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. *Proceedings of the International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 29–39 (cit. on pp. 46, 50, 57, 69, 215).

Witten, I. H., Frank, E., & Hall, M. a. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann. (Cit. on p. 46).

Wodecki, A. (2019). *Artificial Intelligence in Value Creation*. Springer. (Cit. on p. 18).

Wu, X., Li, P., & Hu, X. (2012). Learning from concept drifting data streams with unlabeled data. *Neurocomputing*, *92*, 145–155 (cit. on p. 165).

Xiao, J., Xiao, Z., Wang, D., Bai, J., Havyarimana, V., & Zeng, F. (2019). Short-term traffic volume prediction by ensemble learning in concept drifting environments. *Knowledge-Based Systems*, *164*, 213–225 (cit. on pp. 130, 137).

Xie, J., & Zhang, B. (2018). Readmission risk prediction for patients with heterogeneous hazard: A trajectory-aware deep learning approach. *Proceedings of the International Conference on Information Systems (ICIS)* (cit. on p. 20).

Xie, P., Bilenko, M., Finley, T., Gilad-Bachrach, R., Lauter, K., & Naehrig, M. (2014). Crypto-nets: Neural networks over encrypted data. *arXiv preprint arXiv:1412.6181* (cit. on pp. 70, 71).

Xie, Y., Kistner, A., & Bleile, T. (2018). *Optimal Automated Calibration of Model-Based ECU-Functions in Air System of Diesel Engines* (tech. rep.). SAE Technical Paper. (Cit. on p. 185).

Xu, J., Rahmatizadeh, R., Boloni, L., & Turgut, D. (2018). Real-time prediction of taxi demand using recurrent neural networks. *IEEE Transactions on Intelligent Transportation Systems*, *19*(8), 2572–2581 (cit. on pp. 131, 141, 147).

Xu, Y., Xu, R., & Yan, W. (2017). Power plant performance modeling with concept drift. *International Joint Conference on Neural Networks (IJCNN)*, 2096–2103 (cit. on p. 112).

Yang, W., Li, Z., Liu, M., Lu, Y., Cao, K., Maciejewski, R., & Liu, S. (2020). Diagnosing concept drift with visual analytics. *IEEE Conference on Visual Analytics Science and Technology* (cit. on p. 39).

Yeshchenko, A., Di Ciccio, C., Mendling, J., & Polyvyanyy, A. (2019). Comprehensive process drift detection with visual analytics. *International Conference on Conceptual Modeling*, 119–135 (cit. on p. 95).

Ying, J. J.-C., Lin, B.-H., Tseng, V. S., & Hsieh, S.-Y. (2015). Transfer learning on high variety domains for activity recognition. *Proceedings of the ASE BigData & SocialInformatics* (cit. on p. 71).

Yu, H.-F., Jain, P., Kar, P., & Dhillon, I. (2014). Large-scale multi-label learning with missing labels. *International Conference on Machine Learning*, 593–601 (cit. on p. 181).

Yu, S., & Abraham, Z. (2017). Concept drift detection with hierarchical hypothesis testing. *Proceedings of the SIAM International Conference on Data Mining*, 768–776 (cit. on pp. 30, 32).

Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., et al. (2018). Accelerating the machine learning lifecycle with MLflow. *IEEE Data Engineering Bulletin*, *41*(4), 39–45 (cit. on p. 54).

Zeileis, A., Kleiber, C., Krämer, W., & Hornik, K. (2003). Testing and dating of structural changes in practice. *Computational Statistics & Data Analysis*, *44*(1-2), 109–123 (cit. on pp. 32, 33, 130, 146).

Zenisek, J., Holzinger, F., & Affenzeller, M. (2019). Machine learning based concept drift detection for predictive maintenance. *Computers & Industrial Engineering*, *137* (cit. on p. 38).

Zerlang, J. (2017). GDPR: A milestone in convergence for cyber-security and compliance. *Network Security*, *2017*(6), 8–11 (cit. on p. 123).

Zhang, J., Zheng, Y., & Qi, D. (2017). Deep spatio-temporal residual networks for citywide crowd flows prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 1655–1661 (cit. on pp. 131, 136).

Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, *1*(1), 7–18 (cit. on p. 181).

Zhang, Y., Fu, J., Yang, C., & Xiao, C. (2019). A local expansion propagation algorithm for social link identification. *Knowledge and Information Systems*, *60*(1), 545–568 (cit. on pp. 69–71).

Zhou, J., Khawaja, M. A., Li, Z., Sun, J., Wang, Y., & Chen, F. (2016). Making machine learning useable by revealing internal states update—A transparent approach. *International Journal of Computational Science and Engineering*, *13*(4), 378–389 (cit. on p. 70).

Zhou, J., Cheng, L., Bischof, W. F., et al. (2008). Prediction and change detection in sequential data for interactive applications. *Proceedings of the AAAI Conference on Artificial Intelligence*, 805–810 (cit. on pp. 112, 118).

Zhou, L., Pan, S., Wang, J., & Vasilakos, A. V. (2017). Machine learning on big data: Opportunities and challenges. *Neurocomputing*, *237*, 350–361 (cit. on p. 3).

Zhou, X., Shen, Y., Zhu, Y., & Huang, L. (2018). Predicting multi-step citywide passenger demands using attention-based neural networks. *Proceedings of the ACM International Conference on Web Search and Data Mining*, 736–744 (cit. on p. 148).

Zhou, X., & Belkin, M. (2014). Semi-supervised learning. *Academic Press Library in Signal Processing* (pp. 1239–1269). Elsevier. (Cit. on p. 217).

Zhou, Z.-H. (2017). Machine learning challenges and impact: an interview with Thomas Dietterich. *National Science Review*, *5*(1), 54–58 (cit. on pp. 67, 71–73, 182).

Zhu, L., & Laptev, N. (2017). Deep and confident prediction for time series at Uber. *IEEE International Conference on Data Mining Workshops (ICDMW)*, 103–110 (cit. on pp. 146, 152).

Zhu, X., Ghahramani, Z., & Lafferty, J. D. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. *International Conference on Machine Learning* (cit. on p. 121).

Zimek, A., Campello, R. J., & Sander, J. (2014). Ensembles for unsupervised outlier detection: Challenges and research questions. *ACM SIGKDD Explorations Newsletter*, *15*(1), 11–22 (cit. on p. 185).

Zimek, A., Schubert, E., & Kriegel, H.-P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *5*(5), 363–387 (cit. on p. 192).

Zizzo, G., Hankin, C., Maffeis, S., & Jones, K. (2019). Intrusion detection for industrial control systems: Evaluation analysis and adversarial attacks. *arXiv preprint*. https://arxiv.org/abs/1911.04278 (cit. on p. 38)

Žliobaitė, I. (2010). Learning under concept drift: An overview. *arXiv preprint*. http://arxiv.org/abs/1010.4784 (cit. on pp. 29, 34, 93, 110, 114, 128, 145, 182)

Žliobaitė, I. (2013). How good is the electricity benchmark for evaluating concept drift adaptation. *arXiv preprint*. https://arxiv.org/abs/1301.3524 (cit. on p. 37)

Žliobaitė, I., Bakker, J., & Pechenizkiy, M. (2012). Beating the baseline prediction in food sales: How intelligent an intelligent predictor is? *Expert Systems with Applications*, *39*(1), 806–815 (cit. on p. 35).

Žliobaitė, I., Bifet, A., Pfahringer, B., & Holmes, G. (2014). Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(1), 27–39 (cit. on pp. 112, 113, 165).

Žliobaitė, I., Pechenizkiy, M., & Gama, J. (2016). An overview of concept drift applications. In N. Japkowicz & J. Stefanowski (Eds.), *Big Data Analysis: New Algorithms for a New Society* (pp. 91–114). Springer International Publishing. (Cit. on pp. 4, 29, 34, 38, 97, 103, 108, 110, 112, 114, 120, 147, 184, 215).

Zur Muehlen, M., & Shapiro, R. (2015). Business process analytics. In J. vom Brocke & M. Rosemann (Eds.), *Handbook on Business Process Management 2: Strategic Alignment, Governance, People and Culture* (pp. 243–263). Springer Berlin Heidelberg. (Cit. on p. 92).

List of Figures

1.1 D&M information systems success model, adapted from Delone and McLean (2003, p. 24). 5
1.2 Lifecycle of information systems, adapted from Duarte and Costa (2012, p. 27). 6
1.3 Overview of thesis content. 11
1.4 Structure of this thesis. 13
1.5 Overview of research methods and research questions. 16

2.1 Overview of solutions for machine learning in data streams. 22
2.2 Virtual and real concept drift, adapted from Gama et al. (2014). . . . 28
2.3 Overview of structural types of concept drift, adapted from Žliobaitė (2010). 29
2.4 Adaptive learning strategies, adapted from Žliobaitė et al. (2016). . . 34

3.1 The spectrum of reproducibility; extended figure based on Peng (2011). 46
3.2 Overview of supervised machine learning steps. 49
3.3 Data sets for training, testing, and final deployment. 54
3.4 Overview of supervised machine learning steps and corresponding problem characteristics and key choices. 55

4.1 Conceptual phase model for categories of challenges. 69
4.2 Challenges identified in literature. 70
4.3 Challenges identified in interviews as well as in literature. 76

5.1 Depiction of learning mode retraining. 96
5.2 Three different approaches for retraining of model. 98
5.3 Typical process variant for a P2P process. 99
5.4 Histogram of the throughput time [h]. 101
5.5 Accuracy of Naïve Bayes without retraining and no drift detection method. 103
5.6 Rolling mean (window 1000) of feature *automation*. 103
5.7 Accuracy of Naïve Bayes with Page-Hinkley. 104

7.1 Model adaptation and drift detection options. 129
7.2 Overall NYC yellow cab and Uber demand per month. 133
7.3 Taxi demand during a blizzard on 2015-01-27. 133
7.4 Design approach of EIA. 136
7.5 Overview of applied drift-detector and model combinations in this work and for future work. 137
7.6 Predictions of EIA during blizzard on 2015-01-27. 139

8.1 Taxi demand in NYC per month. 148
8.2 Explanation of switching scheme. 151
8.3 Quarterly rolling RMSE of static models. 154
8.4 Quarterly rolling SMAPE of static models. 155
8.5 Yearly average SMAPE of best strategies. 157

9.1 Partitioning of data stream. 171
9.2 Behaviour of *UDD* and *KSWIN* on synthetic Friedman data set. 175
9.3 Relationship between deciles of uncertainty and prediction performance. 177
9.4 Relationship between deciles of confidence and accuracy. 178

10.1 Operating principle of the prediction method. 186
10.2 Normalized pressure trajectory. 190
10.3 Sensitivity of the Mahalanobis distance regarding a phase shift. . . . 194
10.4 Error plot with complex neural network. 199

11.1 Overview of methodological concept drift contributions. 213

A.1 Behaviour of *UDD* and *KSWIN* on synthetic Mixed data set. 270

List of Tables

2.1 Different categories of drift detection algorithms and popular representatives. . . . 30
2.2 Popular synthetic and real-world data sets for concept drift evaluation. 37

3.1 Exemplary reportcard based on the Iris data set. Bold writing indicates a problem characteristic or choice from the Reportcard. . . . 58
3.2 Number of screened and relevant articles for each outlet from 2010 to 2018. . . . 59
3.3 Amount of supervised machine learning articles in the outlets of MISQ, ISR, JMIS, ICIS, and ECIS from 2010 to 2018. . . . 60
3.4 Overview of Supervised Machine Learning Reportcard steps and their documentation. . . . 62

4.1 Industry overview of interviewees. . . . 75

5.1 Overview of predictive features. . . . 100
5.2 Overview of multi-class target variable. . . . 101
5.3 Pretest with different models on subset of data. . . . 102
5.4 Performance of drift detection strategies on process mining data set. 104
5.5 Performance of data selection strategies on process mining data set. . 105

6.1 Setup decisions for predictive service. . . . 113
6.2 Algorithmic decisions for predictive service. . . . 115
6.3 Operation of predictive service. . . . 117
6.4 Heatmap of existing research classified into framework. . . . 119
6.5 Different scenarios for label availability. . . . 121

7.1 First results of EIA in comparison to benchmarks, based on RMSE and SMAPE (the lower the better). . . . 138
7.2 First results of EIA in comparison to benchmarks, based on RMSE and SMAPE (the lower the better). . . . 140

8.1 Overview on drift handling strategies. . . . 149
8.2 Overall evaluation of static models. . . . 153

8.3 Evaluation of periodic adaptation. 155
8.4 Evaluation of triggered adaptation. 156
8.5 Evaluation on flight records data set. 158

9.1 Evaluation on synthetic data sets. 175
9.2 RMSE (the lower the better) on *regression* benchmark data sets. Number of retrainings in brackets (the lower the less computationally expensive). *No Retraining* depicts the lower-bound benchmark, while *KSWIN(unl.)* and *ADWIN* represent the upper-bound performance benchmark. 176
9.3 MCC (the higher the better) on *classification* benchmark data sets. Number of retrainings in brackets (the lower the less computationally expensive). *No Retraining* depicts the lower-bound benchmark, while *KSWIN(unl.)* and *ADWIN* represent the upper-bound performance benchmark. 176

10.1 Evaluation of different outlier detection approaches. 192
10.2 Results of pretest with various prediction models. 194
10.3 Parameter results of grid search. 195
10.4 Evaluation of different prediction models. 196
10.5 Performance of outlier detection in combination with prediction model (MAPE). 198
10.6 5 combinations with the lowest MAPE. 199

11.1 Classification of different methods regarding detection and adaptation. 212

A.1 Lifecycle of Microsoft Team Data Science Process and equivalent of Reportcard. 265
A.2 Steps of CRISP-DM and equivalent of Reportcard. 266
A.3 Overview of Supervised Machine Learning Reportcard steps, their problem characteristics and choices as well as their documentation in the journal publications analyzed. 267
A.4 Overview of Supervised Machine Learning Reportcard steps, their problem characteristics and choices as well as their documentation in the conference publications analyzed. 268
A.5 Neural network architecture for each data set. 269
A.6 Characteristics of used data sets. 269
A.7 Different values of α for *UDD*. 270

A.8 SMAPE (the lower the better) on *regression* benchmark data sets. Number of retrainings in brackets (the lower the less computationally expensive). *No Retraining* depicts the lower-bound benchmark, while *KSWIN(unl.)* and *ADWIN* represent the upper-bound performance benchmark. 270
A.9 F1-score* (the higher the better) on *classification* benchmark data sets. Number of retrainings in brackets (the lower the less computationally expensive). *No Retraining* depicts the lower-bound benchmark, while *KSWIN(unl.)* and *ADWIN* represent the upper-bound performance benchmark. 271

Appendix

A

Appendix Chapter 3

Tab. A.1.: Lifecycle of Microsoft Team Data Science Process and equivalent of Reportcard.

MTDSP stages	**Related reportcard choices / characteristics**
Business understanding	
Define objectives	Model initiation – Problem statement
Identify data sources	Model initiation – Data gathering
Data acquisition and understanding	
Ingest the data	Model initiation – Data gathering
Explore the data	Model initiation – Data distribution, Model initiation – Data quality
Set up a data pipeline	n.a.
Modeling	
Feature engineering	Model initiation – Data preprocessing methods, Model initiation – Feature engineering and vectorizing
Model training	Model training – Algorithm, Performance estimation – Data Splitting method, Model testing – Performance metric
Suitability for production	Model testing – Performance evaluation (benchmarks)
Deployment	
Operationalize a model	Model deployment
Customer acceptance	
System validation	n.a.
Project hand-off	n.a.

Tab. A.2.: Steps of CRISP-DM and equivalent of Reportcard.

CRISP DM phases and tasks	Related reportcard choices / characteristics
Business understanding	
Determine business objectives	Model initation – Problem statement
Assess situation	n.a.
Determine data mining goals	Model initation – Problem statement
Produce project plan	n.a.
Data understanding	
Collect initial data	Model initiation – Data gathering
Describe data	Model initiation – Data distribution
Explore data	Model initiation – Data distribution
Verify data quality	Model initiation – Data quality
Data preparation	
Select data	Model initiation – Sampling
Clean data	Model initiation – Data quality
Construct data	Model initiation – Data preprocessing methods
Integrate data	Model initiation – Data gathering
Format data	odel initiation – Feature engineering and vectorizing
Modeling	
Select Modeling Technique	Model training – Algorithm
Generate Test Design	Performance estimation – Data Splitting method
Build Model	Model training – Algorithm/ Performance Estimation – Parameter optimization
Assess Model	Model testing – Performance metric
Evaluation	
Evaluate results	Model testing – Performance evaluation (benchmarks)
Review process	n.a.
Determine next steps	n.a.
Deployment	
Plan deployment	Model deployment – Data used
Plan monitoring and maintenance	Model deployment – Model validity (continuous improv./robustness)
Produce final report	n.a.
Review project	n.a.

Tab. A.3.: Overview of Supervised Machine Learning Reportcard steps, their problem characteristics and choices as well as their documentation in the journal publications analyzed.

Step	Indicator		Described in articles		Positive Example
Model initiation	Problem statement		100,00%	(35/35)	Abbasi et al. 2012
	Data gathering		88,57%	(31/35)	Lin et al. 2017
	Data distribution		82,86%	(29/35)	T. Wang et al. 2013
	Sampling		37,14%	(13/35)	Samtani et al. 2017
	Data quality		37,14%	(13/35)	Dong, Liao, and Zhang 2018
	Data preprocessing methods		71,43%	(25/35)	Pant and Srinivasan 2013
	Feature engineering and vectorizing		62,86%	(22/35)	Twyman et al. 2015
Performance estimation	Parameter Optimization	Search Space	8,57%	(3/35)	Martens et al. 2016
		Search Algorithm	11,43%	(4/35)	Martens et al. 2016
	Data split		94,29%	(33/35)	Lash and Zhao 2016
	Algorithm		100,00%	(35/35)	W. Li, Chen, and Nunamaker Jr 2016
	Sampling		5,71%	(2/35)	Kitchens et al. 2018
	Performance metric (reasoned)		42,86%	(15/35)	Shi et al. 2017
	Performance evaluation		68,57%	(24/35)	Cui, Wong, and Wan 2012
Model deployment	Data used		2,86%	(1/35)	Abbasi et al. 2018
	Model validity	Continuous Improvement	2,86%	(1/35)	Abbasi et al. 2018
		Robustness	8,57%	(3/35)	Mo, Sarkar, and Menon 2018

Tab. A.4.: Overview of Supervised Machine Learning Reportcard steps, their problem characteristics and choices as well as their documentation in the conference publications analyzed.

Step	Indicator		Described in articles			Positive Example
Model initiation	Problem statement		100,00%	(86/86)		Kowatsch and Maass 2018
	Data gathering		87,21%	(75/86)		Ram 2015
	Data distribution		70,93%	(61/86)		K.-Y. Huang, Nambisan, and Uzuner 2010
	Sampling		5,81%	(5/86)		Stange and Funk 2015
	Data quality		80,23%	(69/86)		Riekert et al. 2017
	Data preprocessing methods		77,91%	(67/86)		Pröllochs, Feuerriegel, and Neumann 2015
	Feature engineering and vectorizing		79,07%	(68/86)		Baumann et al. 2015
Performance estimation	Parameter Optimization	Search Space	13,95%	(12/86)		Tafti and Gal 2018
		Search Algorithm	13,95%	(12/86)		Staudt, Rausch, and Weinhardt 2018
	Data split		96,51%	(83/86)		Chatterjee et al. 2018
	Algorithm		100,00%	(86/86)		Tripathi and Kaur 2018
	Sampling		6,98%	(6/86)		Lüttenberg, Bartelheimer, and Beverungen 2018
	Performance metric (reasoned)		51,16%	(44/86)		Blanc and Setzer 2015
	Performance evaluation		40,70%	(35/86)		Geva and Oestreicher-Singer 2013
Model deployment	Data used		1,16%	(1/86)		Laing and Kühl 2018
	Model validity	Continuous Improvement	1,16%	(1/86)		Seebach, Pahlke, and Beck 2011
		Robustness	18,60%	(16/86)		Goby et al. 2016

Appendix Chapter 10

Table A.5 explains details regarding the neural network architecture for each data set. The column *Network Structure* indicates how many neurons per hidden layer are applied. Column *Dropout Rate* contains the dropout rate for each dropout layer. The last column *# Forward Passes* explains how many forward passes for MCD are computed.

Tab. A.5.: Neural network architecture for each data set.

Data Set	Network Structure	Dropout Rate	# Forward Passes
Air Quality	(128, 64, 32, 16)	(0.2, 0.2, 0.1, 0.1)	100
Bike Sharing	(128, 64, 32, 16)	(0.2, 0.2, 0.1, 0.1)	100
Insects Abrupt	(128, 64, 32, 16, 8)	(0.1, 0.1, 0.1, 0.1, 0.1)	50
Insects Inc	(128, 64, 32, 16, 8)	(0.1, 0.1, 0.1, 0.1, 0.1)	50
Insects IncAbr	(32, 16, 8)	(0.1, 0.1, 0.1)	50
Insects IncReo	(128, 64, 32)	(0.1, 0.1, 0.1)	50
KDDCUP99	(32, 16, 8)	(0.1, 0.1, 0.1)	50
Gas Sensor	(128, 64, 32, 16, 8)	(0.2, 0.2, 0.2, 0.2, 0.2)	50
Electricity	(32, 16, 8)	(0.1, 0.1, 0.1)	50
Rialto Bridge	(512, 512, 256, 32)	(0.2, 0.2, 0.1, 0.1)	50

Tab. A.6.: Characteristics of used data sets.

Data Set	Samples	Features	Target
Air Quality	9,357	8	continuous
Bike Sharing	17,379	12	continuous
Insects Abrupt	52,848	33	6 classes
Insects Inc	57,018	33	6 classes
Insects IncAbr	79,986	33	6 classes
Insects IncReo	79,986	33	6 classes
KDDCUP99	494,020	118	23 classes
Gas Sensor	13,910	128	6 classes
Electricity	45,312	8	2 classes
Rialto Bridge	82,500	27	10 classes

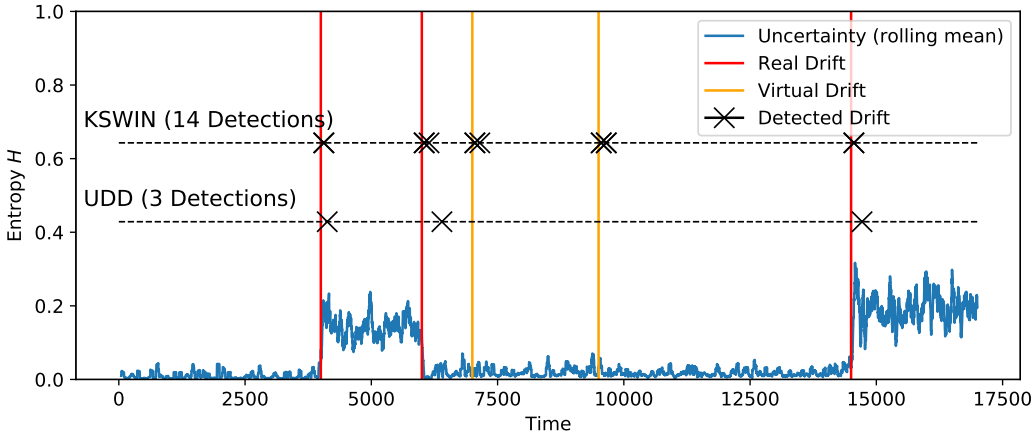


Fig. A.1.: Behaviour of *UDD* and *KSWIN* on synthetic Mixed data set.

Tab. A.7.: Different values of α for *UDD*.

Data Set	Parameter
Air Quality	10^{-5}
Bike Sharing	10^{-27}
Insects Abrupt	0.002
Insects Inc	0.1
Insects IncAbr	0.1
Insects IncReo	0.01
KDDCUP99	10^{-43}
Gas Sensor	0.01
Electricity	10^{-9}
Rialto Bridge	10^{-86}

Tab. A.8.: SMAPE (the lower the better) on *regression* benchmark data sets. Number of retrainings in brackets (the lower the less computationally expensive). *No Retraining* depicts the lower-bound benchmark, while *KSWIN(unl.)* and *ADWIN* represent the upper-bound performance benchmark.

		Limited Label Avail.				Unlimited Label Avail.	
Data Set	**No Retr.**	**Uninf.**	**Equal D.**	**KSWIN**	**UDD**	***KSWIN(unl.)***	***ADWIN***
Air Quality	12.46 (0)	10.14 (14)	**8.84** (14)	10.56 (14)	8.96 (14)	10.33 (19)	10.68 (12)
Bike Sharing	71.88 (0)	62.76 (5)	56.94 (5)	61.04 (5)	**55.49** (5)	46.67 (27)	54.74 (8)

Tab. A.9.: F1-score* (the higher the better) on *classification* benchmark data sets. Number of retrainings in brackets (the lower the less computationally expensive). *No Retraining* depicts the lower-bound benchmark, while *KSWIN(unl.)* and *ADWIN* represent the upper-bound performance benchmark.

		Limited Label Avail.				Unlimited Label Avail.	
Data Set	**No Retr.**	**Uninf.**	**Equal D.**	**KSWIN**	**UDD**	***KSWIN(unl.)***	***ADWIN***
Insects Abrupt	0.498 (0)	0.514 (9)	0.514 (9)	0.507 (9)	**0.564** (9)	0.575 (177)	0.555 (11)
Insects Inc	0.202 (0)	0.323 (4)	0.321 (4)	0.303 (4)	**0.353** (4)	0.345 (27)	0.356 (3)
Insects IncAbr	0.331 (0)	0.519 (22)	0.522 (22)	0.512 (22)	**0.567** (22)	0.542 (107)	0.561 (23)
Insects IncReo	0.219 (0)	0.301 (10)	0.302 (10)	0.265 (10)	**0.311** (10)	0.329 (149)	0.336 (13)
KDDCUP99	0.136 (0)	0.188 (20)	0.176 (20)	0.148 (20)	**0.258** (20)	0.354 (345)	0.411 (61)
Gas Sensor	0.287 (0)	0.533 (39)	0.537 (39)	0.365 (39)	**0.546** (39)	0.504 (149)	0.537 (49)
Electricity	0.077 (0)	0.641 (13)	0.641 (13)	0.529 (13)	**0.672** (13)	0.712 (269)	0.695 (45)
Rialto Bridge	0.580 (0)	0.601 (14)	0.604 (14)	0.622 (14)	**0.625** (14)	0.625 (17)	0.640 (116)

*F1-score for data sets with more than two classes is computed by macro averaging.

Declarations

Eidesstattliche Versicherung

gemäß §13 Absatz 2 Ziffer 3 der Promotionsordnung des Karlsruher Instituts für Technologie für die KIT-Fakultät für Wirtschaftswissenschaften

1. Bei der eingereichten Dissertation zu dem Thema *"Concept Drift Handling in Information Systems: Preserving the Validity of Deployed Machine Learning Models"* handelt es sich um meine eigenständig erbrachte Leistung.

2. Ich habe nur die angegebenen Quellen und Hilfsmittel benutzt und mich keiner unzulässigen Hilfe Dritter bedient. Insbesondere habe ich wörtlich oder sinngemäß aus anderen Werken übernommene Inhalte als solche kenntlich gemacht.

3. Die Arbeit oder Teile davon habe ich bislang nicht an einer Hochschule des In- oder Auslands als Bestandteil einer Prüfungs- oder Qualifikationsleistung vorgelegt.

4. Die Richtigkeit der vorstehenden Erklärungen bestätige ich.

5. Die Bedeutung der eidesstattlichen Versicherung und die strafrechtlichen Folgen einer unrichtigen oder unvollständigen eidesstattlichen Versicherung sind mir bekannt.

Ich versichere an Eides statt, dass ich nach bestem Wissen die reine Wahrheit erklärt und nichts verschwiegen habe.

Karlsruhe, den 15.04.2021

Lucas Baier