SN Computer Science (2022) 3:445
https://doi.org/10.1007/s42979-022-01338-z

ORIGINAL RESEARCH

An Empirical Evaluation of Constrained Feature Selection

Jakob Bach[1] · **Kolja Zoller**[2] · **Holger Trittenbach**[1] · **Katrin Schulz**[2,3] · **Klemens Böhm**[1]

Received: 18 March 2022 / Accepted: 19 July 2022

Abstract
While feature selection helps to get smaller and more understandable prediction models, most existing feature-selection techniques do not consider domain knowledge. One way to use domain knowledge is via constraints on sets of selected features. However, the impact of constraints, e.g., on the predictive quality of selected features, is currently unclear. This article is an empirical study that evaluates the impact of propositional and arithmetic constraints on filter feature selection. First, we systematically generate constraints from various types, using datasets from different domains. As expected, constraints tend to decrease the predictive quality of feature sets, but this effect is non-linear. So we observe feature sets both adhering to constraints and with high predictive quality. Second, we study a concrete setting in materials science. This part of our study sheds light on how one can analyze scientific hypotheses with the help of constraints.

Keywords Feature selection · Constraints · Domain knowledge · Theory-guided data science

Introduction

Motivation

Feature selection targets at identifying the variables in a dataset that are most useful for predictions [23]. Feature selection can increase prediction quality, reduce hardware requirements and ease understanding of the data [8]. Most feature-selection algorithms optimize a quantitative quality criterion, like the accuracy of predictions. However, it also has become important to have constraints which sets of selected features must adhere to [19, 30, 36, 43]. Formulating such constraints typically requires familiarity with the respective domain. We see at least four situations when using constraints is helpful:

1. *Firm domain knowledge* In many scientific settings, domain knowledge is available [33, 60]. One kind of domain knowledge is firm, established knowledge of scientific communities. To obtain so-called physically consistent models, researchers want to express known facts explicitly so that machine learning considers this knowledge. For feature selection, one can make relationships between features explicit in the form of constraints. To illustrate, a researcher might know that some features are redundant and thus want to rule out combinations of these features. This idea of considering established knowledge has gained much attention in the literature recently [33].
2. *Hypotheses* 'Domain knowledge' also includes hypotheses, i.e., ideas or expectations that a domain expert would like the data-analysis process or, more specifically here, a feature-selection algorithm to respect. An example is that researchers are hypothesizing that certain features are redundant, and they want to study

✉ Jakob Bach
jakob.bach@kit.edu

Kolja Zoller
kolja.zoller@kit.edu

Holger Trittenbach
holger.trittenbach@kit.edu

Katrin Schulz
katrin.schulz@kit.edu

Klemens Böhm
klemens.boehm@kit.edu

[1] Department of Informatics, Karlsruhe Institute of Technology (KIT), Am Fasanengarten 5, 76131 Karlsruhe, Baden-Württemberg, Germany

[2] Department of Mechanical Engineering, Karlsruhe Institute of Technology (KIT), Kaiserstraße 12, 76131 Karlsruhe, Baden-Württemberg, Germany

[3] Faculty of Mechanical Engineering and Mechatronics, Karlsruhe University of Applied Sciences, Moltkestraße 30, 76133 Karlsruhe, Baden-Württemberg, Germany

Published online: 17 August 2022

whether subsets of features without these hypothesized redundancies suffice for predictions. To do this, they can formulate a respective constraint: If prediction quality drops significantly with feature sets respecting the constraint, this speaks against the hypothesis.

3. *Preferences* Researchers might have specific preferences, or there are external requirements. An example is that they want to inspect the feature-selection results later manually, and they only can do this efficiently if the results have a particular structure.
4. *Alternatives* Constraints are helpful to implement the 'alternate' paradigm, i.e., targeting at analysis results that are different from ones already obtained earlier, to obtain a broader understanding of the scientific phenomenon under investigation. Researchers have studied this in contexts like association rules and clustering [4], and it is useful for feature selection as well. In other words, constraints may explicitly exclude the currently selected feature set, and the feature-selection algorithm must find an alternative. Such constraints can yield different feature sets that have similar prediction quality.

In all four cases, constraints tend to be abstract and do not prescribe a concrete feature set yet. Constraints also give way to interpretability, a recent and highly relevant topic in machine learning [7, 17]. For instance, aligning sets of features with domain knowledge, the first situation, makes things easier to understand for a domain expert. In general, the notion of 'constraint' is broad, as constraints can be domain-independent as well as domain-specific. For instance, establishing an upper bound on the number of selected features is a domain-independent constraint.

Example 1 This example features a domain-specific constraint in materials science. Specifically, we look at the phenomenon that the microstructure of a material specimen changes if a load is applied and the material deforms. The evolution of the microstructure is a complex process, at present typically analyzed with simulations with high computational costs [57]. Prediction models may help uncover and describe relationships between physical quantities in such simulation datasets. Feature selection is of great interest here, as the set of quantities is large, and it is worthwhile to narrow it down to a small set of predictors for a target quantity. The physical mechanisms of deformation are related to so-called twelve slip systems, i.e., twelve planes and directions in space. For most physical quantities, one can derive twelve features correspondingly. However, domain experts in our team have come up with the hypothesis that not all slip systems are needed to describe the microstructural evolution. Constraints allow integrating this hypothesis into feature selection. One can now formulate the following constraint: "For each physical quantity, select at most two out of those twelve features."

Problem Statement

In this article, we evaluate the impact of constraints on feature selection empirically, in a domain-specific as well as in a domain-independent study. As a shorthand, we use the term *constrained feature selection*. We assume the constraints to be pre-defined before feature selection, e.g., based on the requirements of domain experts. Thus, constraints should not improve the predictive quality of selected features in usual cases. Instead, constraints should fulfill different purposes, like improving the interpretability of feature-selection results. In fact, constraints prune feature sets that would be allowed without constraints. Therefore, constrained feature selection should be seen as complementary rather than a competitor to traditional feature selection. In particular, constraints can be combined with various existing feature-selection techniques.

Ideally, the quality of the features selected under constraints is similar to the unconstrained case. However, it can become significantly lower. An important question is if there are sweet spots, i.e., high-quality feature sets that adhere to the given constraints. Put differently, the exact relationship between constraints and their characteristics on the one hand and that decrease of quality on the other hand currently tends to be unclear. Next, the number of constraints one can formulate is vast, as follows: First, there are various constraint types, e.g., different logical operators like AND, OR, etc., to provide a simple example. Second, one can formulate constraints for different subsets of the feature set. Third, one can have several constraints simultaneously, resulting in interactions between constraints. All this calls for a systematic evaluation of how constraints affect feature-selection results.

Constraints have been used in many areas of knowledge discovery like pattern mining [44] and clustering [12]. In feature selection, integrating constraints has also been a topic for years. But existing work tends to aim at just one constraint type, like group constraints [15, 29, 56, 64, 69], cost constraints [30, 43, 48, 50, 68] or cardinality constraints [34, 36, 54, 63], but does not systematically cover the effects of constraints on feature selection. The focus of that work tends to be on developing efficient algorithms that take (specific) constraints into account. In this article, in turn, we empirically evaluate the impact of constraints on the feature-selection results. We consider various constraint types and explicitly look at combinations of constraints and their effects.

Challenges

Evaluating constrained feature selection is challenging for several reasons: (C1) size of solution space, (C2) size of evaluation space, and (C3) choice of evaluation metrics. Challenge (C1) (size of solution space) refers to the fact that feature selection is a combinatorial problem. The presence of constraints does not change this. To illustrate, if one formulates constraints with propositional logic, then it is an NP-complete problem only to determine if there is any valid feature set at all [11]. When combining constraints, the space of valid feature sets can take an arbitrary shape and volume. Thus, finding the best feature set under constraints often is costly. Challenge (C2) (size of evaluation space) refers to the large number of potential constraints. The challenge is to develop an experimental design that covers this space broadly and systematically. Just working with a few manually defined constraints from a particular use case might not suffice to obtain general insights.

Challenge (C3) (choice of evaluation metrics) is how to describe and measure the impact of constraints. This challenge is twofold: First, one needs to identify interesting characteristics of constraints, e.g., how many feature sets they prune, and quantifying them may not always be obvious. Second, one needs to characterize the feature-selection results, e.g., the predictive quality of a feature set. For both categories, it is not clear which metrics one should use.

Contributions

Having started with a case in materials science, we have seen the usefulness of applying constraints to feature selection. However, as mentioned in Challenge (C2), it is difficult to systematically evaluate the impact of constraints when focusing on one particular domain and the (relatively few) constraints occurring there. So we make two contributions: We conduct a systematic, domain-independent study as well as a domain-specific case study. The first study lets us generally analyze the impact of constraints on feature selection. We use 35 regression datasets from various domains. For these datasets, we generate synthetic constraints systematically. We vary the number of constraints, constraint types, and features used when formulating constraints. In particular, we use a diverse set of constraints formulated in propositional logic and linear arithmetic. Such kinds of constraints are typical in related work. Next, we focus on filter feature selection, which is computationally efficient and independent of the choice of a prediction model [37]. We analyze the relationships between various characteristics, describing the constraints and the feature-selection results.

Second, we conduct a case study in materials science to evaluate the impact of constraints for a concrete use case. Materials science is a domain where literature argues for integrating domain knowledge into machine learning [41, 60]. In our case study, we apply an established numerical simulation method to a material specimen subjected to tensile loading. The evolution of the microstructural processes in the material is computed and post-processed to extract a dataset. The dataset is high-dimensional, containing 135 features. The physical processes behind the data are complex and currently not fully understood. This means that the constraints used here mainly are domain-specific hypotheses. We evaluate the hypotheses by analyzing how the corresponding constraints affect feature selection.

Results

We have both concrete results regarding the impact of constraints as well as insights on a methodological level, which might help domain scientists wanting to do constrained feature selection. Using synthetic constraints, we observe that the impact of constraints is similar for all datasets, i.e., observations tend to generalize over datasets. At the same time, we see that this impact strongly depends on the constraint type. For example, some constraint types decrease the quality of the selected features much more than others. An overall trend is that the quality of feature sets decreases if constraints become stronger, as one would expect. However, our experiments show that there is a positive side to this: Namely, the effect is not necessarily linear, i.e., even if constraints prune a significant fraction of feature sets, the quality of the features does not necessarily decrease to the same extent. In other words, there seem to be sweet spots with high-quality feature sets. This means that it might be worthwhile for domain experts to look out for such spots.

In our case study, none of our domain-specific hypotheses, expressed as constraints, decreases prediction quality by much. This means that the data does not invalidate any hypothesis. Domain scientists could now think about more elaborate (but at the same time more involved) ways of verifying the hypotheses with more rigor. We find this encouraging as well—one can interpret our results as an indication that all this is a way to evaluate scientific hypotheses, at least in a preliminary fashion. To carry out such analyses, looking back at our case study, we now recommend formulating constraints not necessarily upfront but after seeing unconstrained results, and refining the constraints iteratively. Such an approach should yield a more focused search for feature sets. Finally, some constraints trigger the selection of features that differ from the

ones in the unconstrained case. In other words, the result are alternative feature sets of similar quality. We take this as a sign that the 'alternate' paradigm is feasible in the context of feature selection and benefits domain experts.

All experimental data[1] and our code[2] are available online.

Paper Outline

"Specification of Problem" specifies the problem of constrained feature selection. "Related Work" summarizes related work. "Study with Synthetic Constraints" presents the design and results of the domain-independent study with synthetic constraints. "Case Study in Materials Science" features the case study in materials science. "Conclusions and Future Work" concludes.

Specification of Problem

Objective Function

Feature Selection

Let $X \in \mathbb{R}^{m \times n}$ be a dataset in the form of a matrix. Each row x_i is a data object, and each column f_j is a feature. In particular, we assume the features are already defined before training a prediction model rather than being extracted by the model. Further, let $y \in \mathbb{R}^m$ be a vector representing the prediction target. With feature selection, one makes binary selection decisions $s_j \in \{0, 1\}$ for each feature, i.e., either do not select it or do select it. The vector $s \in \{0, 1\}^n$ combines all these selection decisions. The selected features form a feature set. The function $Q(s, X, y)$ returns the quality of a feature set and should be maximized. Feature-selection approaches differ in their quality function $Q(s, X, y)$ and the algorithm determining the optimal feature set. Two big categories of approaches are wrapper feature selection and filter feature selection. We focus on the latter in this article, as explained later.

Wrapper Feature Selection

With wrapper feature selection, $Q(s, X, y)$ is evaluated by training a prediction model with the selected features [37]. These evaluations are part of a search that iterates over candidate feature sets, using a metaheuristic [2]. Because of the repetitions of the model training, wrapper feature selection is expensive. In addition, if the type of model is unknown, this is a black-box optimization problem. Thus, optimization algorithms cannot use the internal structure of the objective function. Instead, they can only observe its outputs.

Filter Feature Selection

Filter feature selection evaluates the quality $Q(s, X, y)$ without prediction models. There are univariate as well as multivariate filter approaches [37]. Univariate filter feature selection breaks $Q(s, X, y)$ down to the qualities $q_j(s_j, f_j, y)$ of individual features. One can compute these qualities independently of each other. The overall quality of the feature set, $Q(s, X, y)$, is an aggregate of the individual qualities, e.g., their sum. Thus, given the individual feature qualities, computing $Q(s, X, y)$ is cheap. Next, one can use white-box optimizers here. For multivariate filter methods, one needs to consider relationships within groups of features. These relationships increase the number of terms in the objective function, so multivariate methods scale worse than univariate approaches. For example, the number of pairwise relationships between features grows quadratically with the number of features. So, for optimizing $Q(s, X, y)$, we focus on univariate filter feature selection in this article. However, to evaluate feature selection, we also train prediction models with the selected features. Next, in our case study in materials science, we consider the pairwise correlation between features in the form of additional constraints.

As quality measure in this article, we take the absolute Pearson correlation $|\rho_{f_j,y}|$ of each feature with the target variable: $q_j(s_j, f_j, y) = s_j \cdot |\rho_{f_j,y}|$ This correlation measure results in a feature quality between zero and one, with zero denoting uninformative or de-selected features. Pearson correlation is one of the simplest and most common feature-quality measures. However, one could also use any other univariate score for filter feature selection instead. One example is mutual information, which also supports categorical features. In preliminary experiments with mutual information, we obtained results similar to those with absolute Pearson correlation. So we stick to the latter for our detailed evaluation.

We now state the optimization objective as follows:

$$\max_s \; Q(s, X, y) = \sum_{j=1}^{n} s_j \cdot |\rho_{f_j,y}| \tag{1}$$

Constraints

Without constraints, the objective in Eq. (1) is trivial to optimize by selecting all features. By introducing constraints, one narrows down the number of valid feature sets. The remaining feature sets are the solutions of the optimization problem. In our work, we consider constraints in

[1] https://doi.org/10.5445/IR/1000148891.

[2] https://github.com/Jakob-Bach/Constrained-Filter-Feature-Selection.

propositional logic and linear arithmetic. These two categories allow formulating various expressive constraints. In particular, all constraints proposed by the domain scientists in our case study can be formulated in that way. Additionally, propositional and linear-arithmetic constraints are typical in related work in software engineering. Finally, the nature of the objective function and the constraints still allow using white-box solvers, e.g., Satisfiability Modulo Theories (SMT) solvers. Many SMT solvers also support other logical theories to formulate constraints, e.g., arrays, bit vectors, or strings.

Propositional Constraints

Propositional constraints describe relationships between features with propositional logic. The logical values $\{False, True\}$ represent the binary selection decisions $s_j \in \{0, 1\}$. One can now use logical operators like $\wedge$ (AND), $\vee$ (inclusive OR), $\oplus$ (XOR, the exclusive or), $\neg$ (NOT), $\rightarrow$ (IMPLIES) and $\leftrightarrow$ (IFF). For example, $s_1 \rightarrow s_2$ means that if Feature 1 is selected, then Feature 2 has to be selected as well. By combining the operators, one can also create formulas for groups of features. For example, $(s_4 \wedge s_7 \wedge s_{10}) \oplus (s_2 \wedge s_{11})$ means that either the three features from the first group or the two features from the second group have to be selected.

Arithmetic Constraints

Linear arithmetic allows the use of the four operators $+$ (addition), $-$ (subtraction), $\cdot$ (multiplication) and $\leq$ (less than or equal to). In the terminology of first-order logic, the first three operators are functions, returning another arithmetic value, while the inequality is a predicate, yielding a logical value. Linear arithmetic does not allow multiplication of two variables, as at least one operand has to be a constant [5]. The objective function in Eq. (1) also is an expression in linear arithmetic once the feature qualities $|\rho_{f_j,y}|$ are computed. In general, one can use linear-arithmetic constraints by assigning each feature a fixed value and formulating an inequality based on this. To illustrate, suppose that each feature has a measurement cost c_j, and one wants to select a subset of features with a total cost below a threshold $C_{\max}$. This yields the following linear inequality:

$$\sum_{j=1}^{n} s_j \cdot c_j \leq C_{\max} \tag{2}$$

For instance, such a situation occurs if feature values are obtained with sensors with different energy consumption, and the total amount of energy per measurement is limited.

Finally, one may use propositional and arithmetic constraints interchangeably in some situations. Think of a group of five features, and one wants to select at least one of them. One way to express this is an arithmetic inequality: the sum over selection decisions for these five features should be at least one. Another way to express this is with propositional logic, using the OR operator.

Experiments

We conduct two types of experiments: first, we systematically generate synthetic constraints on datasets from various domains. By doing so, we conduct a comprehensive evaluation of the impact of constraints, independent from the domain. Second, we carry out a case study in the domain of materials science. In this study, we use constraints to analyze domain-related hypotheses.

Implementation

We implement our entire experimental pipeline in Python and make the code available online. We use the SMT theorem prover Z3 [13], which is popular in related work in software engineering. It can handle optimization problems in addition to checking satisfiability. Among other logical theories, it supports propositional logic and linear arithmetic, which suffices for our constraint types. However, our work does not depend on this particular solver; one can use any other solver with these capabilities. Next, while the objective function in Eq. (1) is easy to evaluate, finding valid feature sets under SMT constraints still is an NP-hard problem in general [5], even without optimization. However, developing strategies for SMT solving is beyond the scope of this paper. We rely on the search heuristics used in Z3 to address Challenge (C1) (size of solution space) efficiently.

Related Work

We review related work from three fields. We begin with data mining, where feature selection is a subfield, and constraints are used within feature selection as well as in other subfields. Second, we look into software engineering, which offers various approaches for constrained feature selection, though the use cases are different from ours. Finally, we discuss feature selection in the domain of materials science, as our case study resides in this domain as well.

Data Mining

Overview

There is a broad spectrum of work on feature selection in general, i.e., without constraints [2, 8, 14, 37]. We refrain from discussing individual approaches in detail here, as our

goal is not to develop a new feature-selection technique. In particular, we do not use constraints to find feature sets superior to those of an unconstrained approach. We rather perceive constraints as domain knowledge that must be considered, potentially reducing prediction performance. Thus, we evaluate constraints in combination with a popular existing feature-selection technique.

Working with constraints is an area of research for various sub-fields of data mining [18], e.g., for clustering [12] and pattern mining [44]. While there also are approaches for considering constraints in feature selection, which we will present next, they usually differ from our work in focus and scope. First, they tend to focus on integrating constraints into particular feature-selection algorithms. In contrast, we systematically evaluate the impact of constraints on feature-selection results. Second, related work usually targets at specific constraint types. In contrast, we work with a variety of constraint types and also combinations of constraints of different types.

Constraint Types and Approaches

There is work on cost constraints [30, 43, 48, 50, 68] and cardinality constraints [34, 36, 54, 63] in supervised feature selection. Feature selection with pre-defined groups of features has been studied as well, though mainly specific to linear models, e.g., group lasso and its variants [15, 29, 56, 64, 69]. Groves [19] is more general than the previous reference, as he presents four feature-selection approaches for arbitrary constraints. However, the evaluation in this article only uses particular types of manually defined constraints.

Other Notions of Constrained Feature Selection

Three further subfields of data mining use constraints in feature selection. However, the approaches are similar to our direction rather by name than by methodology. First, feature selection with constraints is studied in semi-supervised learning [26, 52, 55, 65]. However, the constraints used there express relationships between data objects, not between features. For example, pairwise 'must-link' and 'cannot-link' constraints express whether two objects belong to the same class or not, without assigning a class label. Second, constraints play a role in unsupervised feature selection [39, 66, 67]. These constraints do not directly express user needs on feature sets, but help find a low-dimensional representation of the data. Third, there is constraint-based feature selection that builds on Bayesian network learning [35]. It does not involve user-defined constraints but conditional independence constraints between features, which are learned and propagated.

Software Engineering

Overview

Outside the field of data mining, there are approaches for constrained feature selection in software engineering [6, 16, 24]. From a technical perspective, the feature-selection problem and constraint types are similar to our scenario. Thus, approaches from that area can also be applied to our scenario, so we will discuss them shortly. However, our article goes beyond applying constraints and focuses on how they impact feature-selection results.

Constraint Types and Approaches

In software engineering, a feature is a characteristic of a software product, not a dataset. There, feature selection aims at configuring a software system. Feature models express logical relationships, i.e., constraints, between these features. These relationships often result in a hierarchical structure of the feature models. Additionally, feature attributes express properties like component costs, memory requirements, etc. These properties form the base for further arithmetic constraints or one or several target functions [20, 25, 53]. Typical solution approaches to find feature sets include sampling [47], constraint solving [62], adapting general-purpose optimization techniques [20] or combining satisfiability solving with general-purpose optimization [21, 22, 25].

Constraint Generation

Related work in software engineering often uses feature models with pre-defined constraints, which can be found in repositories, e.g., LVAT [40] and SPLOT [42]. Only some literature in software engineering also generates constrained feature models for a more systematic evaluation. Similar to our study with synthetic constraints, some proposals iteratively generate constraints, randomly picking constraint types and the features to be used [20, 46, 58]. However, these proposals use generated constraints to evaluate new approaches for dealing with constraints, while we focus on evaluating the impact of constraints themselves.

Materials Science

Overview

In materials science, various feature-selection approaches have already been applied, e.g., [1, 28, 32], but without considering constraints. At the same time, several papers argue for the integration of domain knowledge in machine learning. Mangal and Holm [41] compare feature-selection techniques for the classification of stress hotspots. The

article observes that the results of many standard feature-selection techniques might not be interpretable from a domain perspective. Wagner and Rondinelli [60] describe general problems which might occur when applying standard machine-learning techniques directly in materials science. The article presents a workflow for machine learning in materials science and argues for relying on domain knowledge as guidance.

Feature Engineering

Childs and Washburn [10] and Ramprasad et al. [51] survey machine-learning approaches in materials science. Some approaches presented there use domain knowledge when creating new features. As one example, physical laws can guide which interaction terms should be created from the original features. As another example, when describing materials with 'fingerprint' feature vectors [27], implicit physical constraints limit possible values of features. Our work differs from both of these directions. First, constrained feature selection focuses on reducing existing feature sets instead of extending them with new features. Second, we constrain the selection of features instead of constraining feature values.

Involving Domain Experts

Apart from constrained feature selection, one can directly involve domain experts in the selection process. Liu et al. [38] use a multi-layer feature-selection approach to predict properties of materials. This approach lets domain experts rate the importance of features. The approach combines these manuals ratings with several automated feature-selection steps. Such an approach requires knowledge about the usefulness of individual features. Our work, in turn, focuses on knowledge or hypotheses regarding relationships between features. Depending on the type of knowledge available in a concrete use case, either approach or even a combination of both might be suitable.

Study with Synthetic Constraints

Experimental Design

Research Questions

To systematically analyze the impact of constraints, we generate many constraints in an automated manner. In comparison, our case study in materials science will involve a relatively small set of constraints, defined intellectually based on domain knowledge. With synthetic constraints, we study the following research questions:

(Q1) Which relationships between constraints and feature-selection results exist?
(Q2) How does the impact of constraints differ between constraint types?
(Q3) How does the impact of constraints differ between datasets?

Question (Q1) is the main question, while Question (Q2) and Question (Q3) target at a deeper understanding of these effects.

Evaluation Metrics

Overview We now address Challenge (C3) (choice of evaluation metrics). For all evaluations, we use two kinds of metrics: metrics describing the constraints and metrics describing the feature-selection results. Both kinds of metrics are needed to evaluate the impact of constraints on feature selection. To describe the constraints, we consider four metrics, i.e., number of constraints n_{co}, number of constrained features n_{cf}, number of unique constrained features n_{ucf}, and number of solutions n_{so}. To describe the feature-selection results, we use three metrics, i.e., number of selected features n_{se}, objective value $Q(s, X, y)$, and prediction performance R^2. We mostly normalize metrics to a minimum of zero and a maximum of one. This normalization facilitates comparisons of characteristics between datasets with different numbers of features and different distributions of feature qualities. The normalized versions of the metrics have the superscript norm, e.g., $n_{\text{so}}^{\text{norm}}$.

Observe that we do not study the runtimes of finding feature sets under constraints. As we use an off-the-shelf solver, which one can replace by another one, runtime benchmarking would mainly yield insights regarding the solver but not on constrained feature selection in general. For the same reason, we also do not compare against other approaches that consider constraints.

In the following, we describe the evaluation metrics informally. "Formulas of Evaluation Metrics" introduces them formally and explains the normalization process.

Metrics describing constraints Constraints are logical and arithmetic formulas, which can become arbitrarily complex. So one can come up with many metrics describing the constraints. In the field of SAT solving, there are dozens of metrics that characterize logical formulas [3, 45]. However, many of these metrics require formulas in a specific form, i.e., conjunctive normal form. Many metrics also capture rather complex concepts, e.g., referring to a graph representation of the logical formula or specific SAT solving approaches. In this study, we use a small set of metrics with simple interpretations. We also pick metrics that describe constraints at different levels of granularity. The first metric we use is the number of constraints n_{co}. This metric is

relatively coarse-grained, as it neither describes the strength of individual constraints nor the interaction between constraints. Second, we look at the number of features involved in constraints n_{cf}. This metric counts the number of features in the logical formula representing a constraint. If a feature occurs several times in a constraint or a set of constraints, it is counted that many times. For example, the constraint set $C = \{s_1 \leftrightarrow s_2, s_1 \vee s_3, s_4 \rightarrow s_5\}$ refers to Feature 1 two times. Third, we study the number of unique features involved in constraints n_{ucf}. Here, each feature is only counted once, even if it appears in a constraint or a set of constraints more than once. Fourth, we evaluate the number of solutions n_{so} remaining after we added constraints to the optimization problem. Phrased in a way specific to feature selection, this is the number of valid feature sets in the presence of constraints. This metric quantifies the size of the solution space. It is the most fine-grained metric describing the constraints. However, it is also costly to compute, as it requires checking whether each possible feature set meets the constraints. The number of possible feature sets grows exponentially with the number of features, i.e., for n features, there are 2^n feature sets. Thus, iterating over all feature sets becomes infeasible with growing n. This effect limits the dimensionality of the datasets in our evaluation.

Metrics describing feature-selection results The result, i.e., solution, of feature selection is a feature set. One can analyze the selected features as well as their use for downstream tasks, like generating predictions. First, we look at the number of selected features n_{se}, i.e., the size of the solution. This metric is a purely quantitative description of the features selected. In a case study, one can also evaluate the selected features qualitatively, i.e., which domain-specific conclusions one can draw if particular features are selected or not. Our second metric is the objective value $Q(s, X, y)$ of the constrained optimization problem, see Eq. (1). This metric guides the search for the optimal feature set and therefore is essential for the evaluation. Third, we take the features selected under constraints and train regression models with them. We consider prediction performance in terms of the R^2 value. R^2 is a standard evaluation metric for regression problems [31]. One could use other prediction-performance measures as well, for instance, if the use case requires a different measure. We summarize objective value and prediction performance as solution quality.

Constraint Generation

We address Challenge (C2) (size of evaluation space) by distinguishing between two aspects when generating constraints. First, the generation strategy varies the features involved in constraints, i.e., the operands of the constraints. Second, the constraint types define the operators for formulating constraints, e.g., AND, OR, etc. Keeping these two aspects separate from each other allows for a comprehensive evaluation.

Generation strategy We employ the same generation strategy for all constraint types except for two cases, which we discuss later, and we use this strategy for all datasets. The generation process varies several characteristics of a constraint set: the number of constraints, the number of features in each constraint, and the actual features in each constraint. On the one hand, this flexibility gives way to broad coverage of the evaluation space. On the other hand, this causes a considerable variance between single runs of constraint generation. Consequently, we repeat the generation process 1000 times per constraint type.

Algorithm 1 Generate and evaluate constraints

Input: Constraint type t
Input: Optimization problem o
Output: Evaluation metrics from Section 4.1.2

for $i \leftarrow 1$ **to** 1000 **do**
  $o' \leftarrow o$ ▷ make copy
  $C \leftarrow \emptyset$ ▷ set of constraints
  $n_{co} \leftarrow$ Choose $n_{co} \in \{1, \ldots, 10\}$ uniformly at random ▷ number of constraints
  for $j \leftarrow 1$ **to** n_{co} **do**
    if t is 'single' constraint **then**
      $n' \leftarrow 2$ ▷ number of features in constraint
    else if t is 'group' constraint **then**
      $n' \leftarrow$ Choose $n' \in \{2, \ldots, n\}$ uniformly at random
    end if
    $F \leftarrow$ Choose n' distinct features uniformly at random
    $c \leftarrow$ Apply t to F ▷ one constraint
    $C \leftarrow C \cup \{c\}$
  end for
  $o' \leftarrow$ Add C to o'
  Solve o'
  Evaluate o'
end for

Algorithm 1 is the process of generating and evaluating constraints. The initial optimization problem o only consists of the objective function, quantifying the feature qualities of one dataset. For a single run of constraint generation, we first vary the number of constraints uniformly at random between one and ten. Next, we decide on the number of features per constraint. Some constraint types always involve two features, and some support larger sets. We refer to the former type as single constraints and the latter type as group

constraints. As we do not want to prefer a particular group size, we choose the group size $n' \in \{2, \ldots, n\}$ uniformly at random. The choice of features involved in a constraint is uniformly random as well. We sample without replacement. After creating constraints, we add them to the optimization problem, run the optimizer, and compute the evaluation metrics from "Evaluation Metrics".

The runtime of this generation-and-evaluation algorithm depends on the number of features in each dataset. In particular, the optimization problem is NP-hard [5], and computing evaluation metric number of solutions n_{so} requires iterating over an exponentially growing number of potential feature sets. Also, the prediction models for evaluating feature-set quality require longer training if the datasets are larger. We consider these limitations when choosing datasets in "Datasets".

Constraint types To specify constraint types, we have to select from the broad range of possible logical operators. In particular, combining logical operators yields other logical operators. For example, the NAND operator can be expressed with an AND and a NOT operator. Furthermore, the same constraint can often be expressed equivalently with different operators. Thus, exhaustively evaluating all possible constraint types is not feasible. The datasets in this study also come from different domains. Thus, it is impossible to formulate a common set of domain-specific constraints. Instead, we use various generic constraint types, which one can apply to any dataset. When compiling the list that follows, we have aimed at simple and, at the same time, diverse constraint types. Appendix section "Study with Synthetic Constraints" contains the formulas for all constraint types.

(T1) *Global-AT-MOST* From the set of all n features, select at most k features. As this constraint type always refers to all features instead of a random subset, we do not use Algorithm 1 here. Instead, we evaluate all possible values of k exhaustively.

(T2) *Group-AT-MOST* From a group of features of size n', select at most k. We choose $k \in \{1, \ldots, n' - 1\}$ uniformly at random.

(T3) *Group-AT-LEAST* From a group of features of size n', select at least k. Again, we choose $k \in \{1, \ldots, n' - 1\}$ uniformly at random. This constraint type alone does not exclude the trivial solution of selecting all features. So we combine it with (T1), requiring that at most half of all features are selected globally.

(T4) *Single-IFF* From a pair of features, select either both or none. To exclude the trivial solution of selecting all features, we combine this constraint type with (T1), requiring that at most half of all features are selected globally.

(T5) *Group-IFF* The transitive extension of (T4): from a group of features of size n', either select all or none. To exclude the trivial solution, we combine this constraint type with (T1), requiring that at most half of all features are selected globally.

(T6) *Single-NAND* From a pair of features, do not select both simultaneously. This constraint is a special case of (T2).

(T7) *Group-NAND* From a group of features of size n', select at most $n' - 1$.

(T8) *Single-XOR* From a pair of features, select exactly one.

(T9) *Group-MIXED* With equal probability, pick between (T2), (T3), (T5), (T7) and (T8). There is no global cardinality constraint (T1).

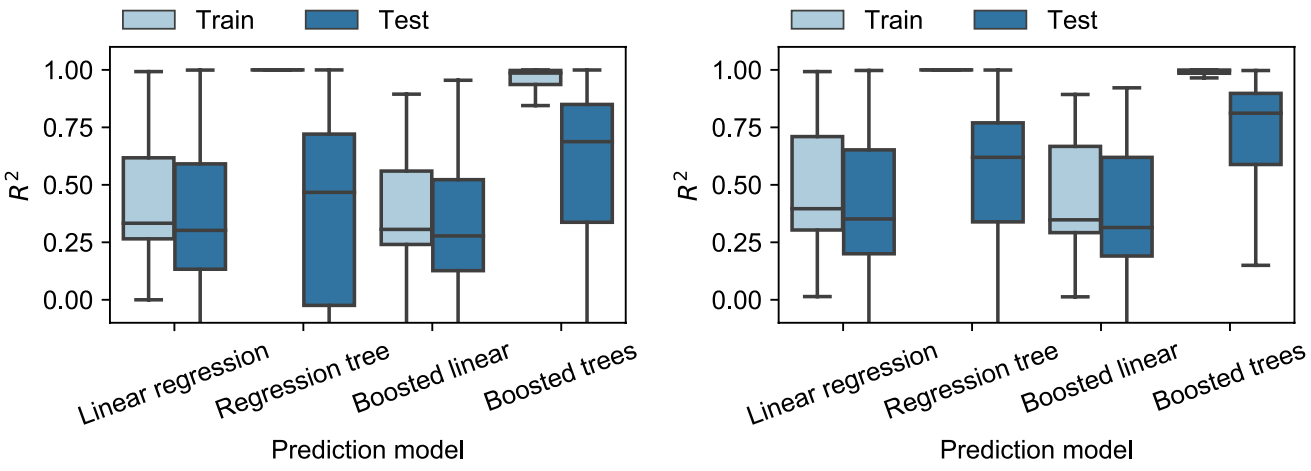
(T10) *UNCONSTRAINED* Mainly to have a reference point, we also consider the unconstrained solution, i.e., the upper bound for the objective value. We only compute this once for each dataset instead of using Algorithm 1. This is because there are no random effects or parameters which would change the outcome in a repeated generation.

Datasets

The study with synthetic constraints is domain-independent. We take 35 regression datasets from the OpenML repository [59]; see our experimental data for details on each dataset. For regression tasks, we can use absolute Pearson correlation between a feature and the target variable as a feature-quality measure. We select and filter the datasets by technical criteria only. To this end, OpenML's Python API allows us to retrieve datasets based on well-defined characteristics. We do not conduct further pre-processing steps.

Dataset size We use medium-sized datasets with between 100 and 10,000 data objects. This number does not affect the runtime of the constrained optimization problem. It only affects the one-time effort to compute the feature-quality values. Additionally, dataset size affects training time for prediction models, which we use to evaluate feature sets. As we evaluate several thousand feature sets for each dataset, we limit dataset size to keep overall evaluation time under control.

Dataset dimensionality We choose datasets with 10–14 numeric features. We drop categorical features since Pearson correlation is undefined for them. This number of features might seem relatively low, considering that feature selection usually targets at high-dimensional feature spaces. As we show in our case study in materials science, the solver we use can handle larger optimization problems. However, in preliminary experiments with high-dimensional data and the same constraint-generation strategy, we observed the following phenomena: First, counting the number of valid solutions under constraints n_{so}, a key evaluation metric, has been infeasible. Second, for some constraint types, optimization has become very costly.



(a) All experimental runs. **(b)** Unconstrained experimental runs.

Fig. 1 Prediction performance, measured with R^2. To keep the plots readable, we exclude outlier points from plotting and truncate the y-axis at -0.1

Third, for the feasible experiments, we have obtained results similar to those of the study conducted here.

Dealing with missing values We exclude datasets with missing values. Of course, when computing feature quality, one does need a strategy to deal with missing values. For example, one could replace missing values with the mean of that feature, exclude individual data objects with missing values, etc. However, it is beyond the scope of our study to include methods that handle missing values.

Prediction Models

After selecting features under constraints, we use the resulting feature sets in regression models. We choose four models: Linear regression, regression tree, boosted linear regression, and boosted trees. The first two models are from the scikit-learn library [49], the last two from xgboost [9]. These models follow two different learning paradigms, linear and tree-based, and have different complexities. We do not optimize the hyperparameters of the models but mostly stick to the defaults. For reproducibility, we set random seeds for the non-deterministic models. For the two xgboost models, we set the size of the ensembles to 20 trees.

We evaluate prediction performance in terms of R^2, i.e., the proportion of variance explained. We conduct tenfold cross-validation for each constraint evaluation. We do not only consider train-test splits for predictions but also in the optimization problem itself: We compute feature qualities on the training splits only.

Evaluation

Comparison of Prediction Performance

Before studying the impact of constraints, we analyze prediction performance. The rationale is twofold: first, we compare prediction models to decide which ones we should keep for subsequent evaluations. Second, we analyze the impact of datasets on prediction quality in the unconstrained case. If datasets had a significant impact, this might overshadow the impact of constraints.

To measure prediction performance, we evaluate prediction models trained after feature selection. In this section, we use R^2 without additional normalization per dataset or prediction model. Figure 1a shows that there is a considerable variation of prediction performance for each prediction model. Note that the plot contains all pipeline runs, i.e., data from different constraint types, datasets, and repetitions of the randomized constraint generation process. However, as we can see in Fig. 1b, there still is a considerable variation in prediction performance when only looking at the unconstrained (T10) run from each dataset. Thus, the datasets themselves have a substantial impact on prediction performance.

We also see that test performance is lower than training performance, i.e., overfitting occurs. However, in our study, we see prediction performance only as a metric to evaluate the impact of constraints, and optimizing prediction performance is not a target per se. In the following sections, we focus on performance on the test folds. We min–max normalize R^2 per dataset and prediction model, i.e., we report

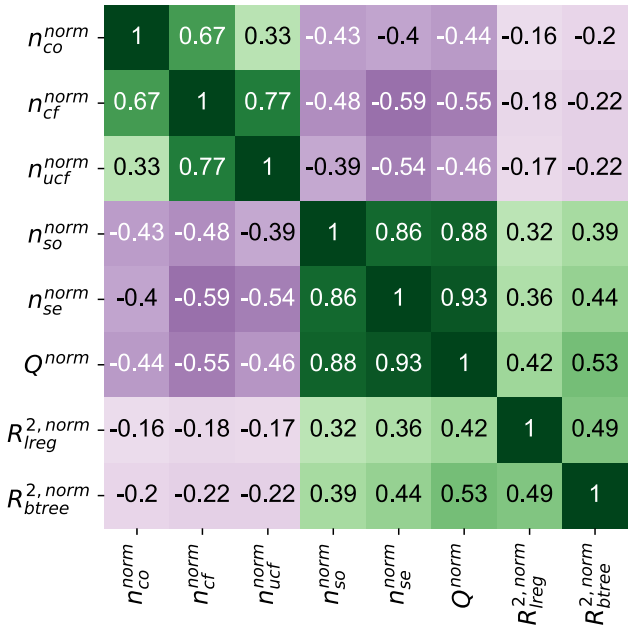


Fig. 2 Correlation between normalized evaluation metrics

$R^{2,\text{norm}}$. This normalization allows focusing on the relative impact of constraints, ignoring differences in dataset difficulty and model complexity.

We limit our analysis to two prediction models: The first one is linear regression, the simplest model of the ones we use. It also is conceptually related to our study's optimization objective, i.e., capturing linear dependencies with Pearson correlation. A disadvantage of linear regression is that it exhibits a relatively low average prediction performance in this study. Second, we use gradient-boosted trees, which yield the best average performance of the models.

Relationship Between Evaluation Metrics (Q1)

We study the relationships between evaluation metrics in two ways. First, we analyze these relationships in terms of correlation. Later, we also plot the metrics against each other. For correlation analysis, we take the values of each normalized evaluation metric over all experiments and compute the Spearman rank correlation between them. See Fig. 2 for the correlation matrix. There is a strong positive correlation between the number of selected features $n_{\text{se}}^{\text{norm}}$, the number of solutions $n_{\text{so}}^{\text{norm}}$, and the objective value Q^{norm}. All pairwise correlations between these metrics are higher than 0.8. Figure 3a shows that there is a roughly linear relationship between the number of selected features $n_{\text{se}}^{\text{norm}}$ and the objective value Q^{norm}. As the objective function of the optimization problem in Eq. (1) is monotonous, it is plausible that selecting more features increases the objective value. However, the quality of the selected features can vary. For the same size of the feature set, different constraints can lead to different objective values, as they might exclude different feature combinations. Figure 3b shows that the relationship between the number of solutions $n_{\text{so}}^{\text{norm}}$ and objective value Q^{norm} is more involved. Decreasing the number of valid feature sets tends to decrease the objective value. However, it also matters which feature sets become invalid due to the constraints. The optimizer might find feature sets with a high objective value even in small solution spaces, depending on the concrete constraints and the distribution of feature qualities. In contrast, we have not observed any scenario with a large solution space but a low objective value.

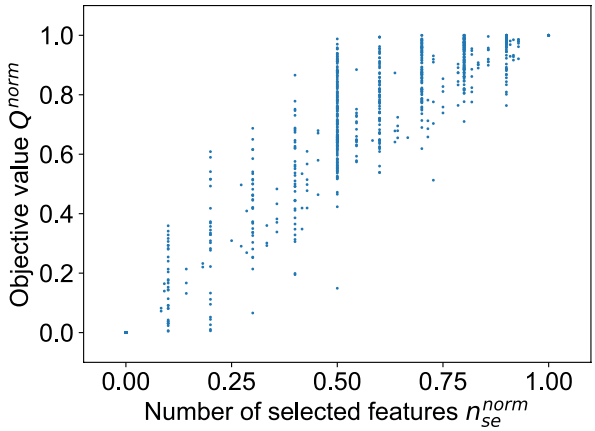
Coming back to the correlation matrix in Fig. 2, there is a moderately negative correlation between metrics describing the complexity of constraints (i.e., number of constraints $n_{\text{co}}^{\text{norm}}$, number of constrained features $n_{\text{cf}}^{\text{norm}}$, number of unique constrained features $n_{\text{ucf}}^{\text{norm}}$) on the one side and number of solutions $n_{\text{so}}^{\text{norm}}$ as well as objective value Q^{norm} on the other side. This effect is expected—using more constraints prunes more solutions. The correlation is only moderate, though, since the features affected by constraints might have different qualities, and constraints also interact. Thus, the other metrics can vary even for the same number of constraints or number of features involved in constraints. Figure 3c graphs the relationship of the number of constraints $n_{\text{co}}^{\text{norm}}$ and the objective value Q^{norm}. For the number of constrained features, the relationship with the number of solutions might even be positive, e.g., for NAND constraints.

Figure 2 reveals that prediction performance $R^{2,\text{norm}}$ is not more than moderately correlated with the objective value and with metrics for the solution space. Figure 3d also shows this in more detail. The objective function cannot describe interactions between features, as it only sums up the individual feature qualities. This caveat is a general characteristic of univariate filter feature selection. In contrast, prediction models evaluate a feature set as a whole.

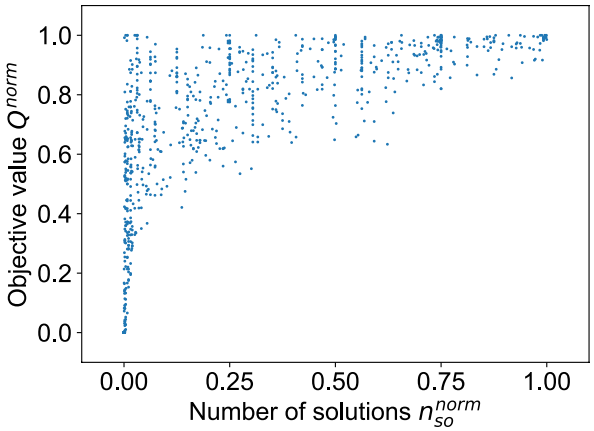
Comparison of Constraint Types (Q2)

Figure 4 shows how evaluation metrics vary between constraint types. As one extreme, without any constraint, i.e., type UNCONSTRAINED (T10), all feature sets are valid, and the objective value is maximal. All features remain selected, except maybe features with a quality of zero, for which selection is optional. Once constraints enter the optimization problem, both the number of solutions and the objective value decrease. As another extreme, there might be no solution satisfying the constraints or only the solution where no features are selected. In these situations, the objective value is zero.

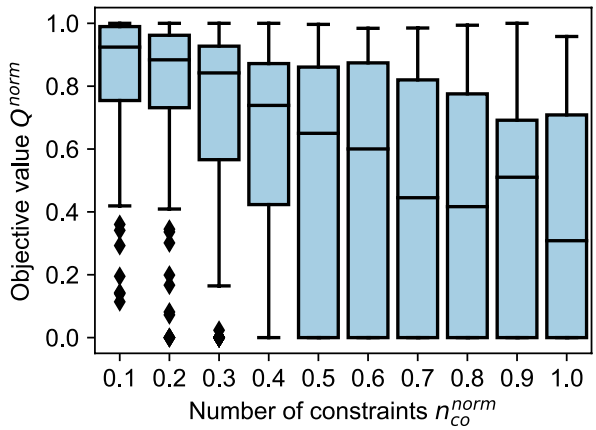
The impact of constraints strongly depends on the constraint type. If one combines different constraint types in



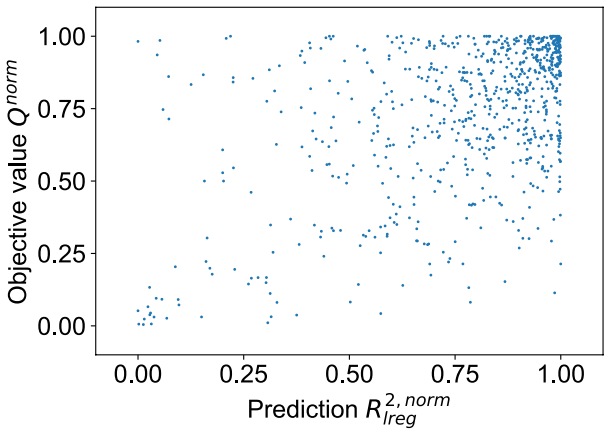
(a) Number of selected features vs. objective value.



(b) Number of solutions vs. objective value.



(c) Number of constraints vs. objective value.

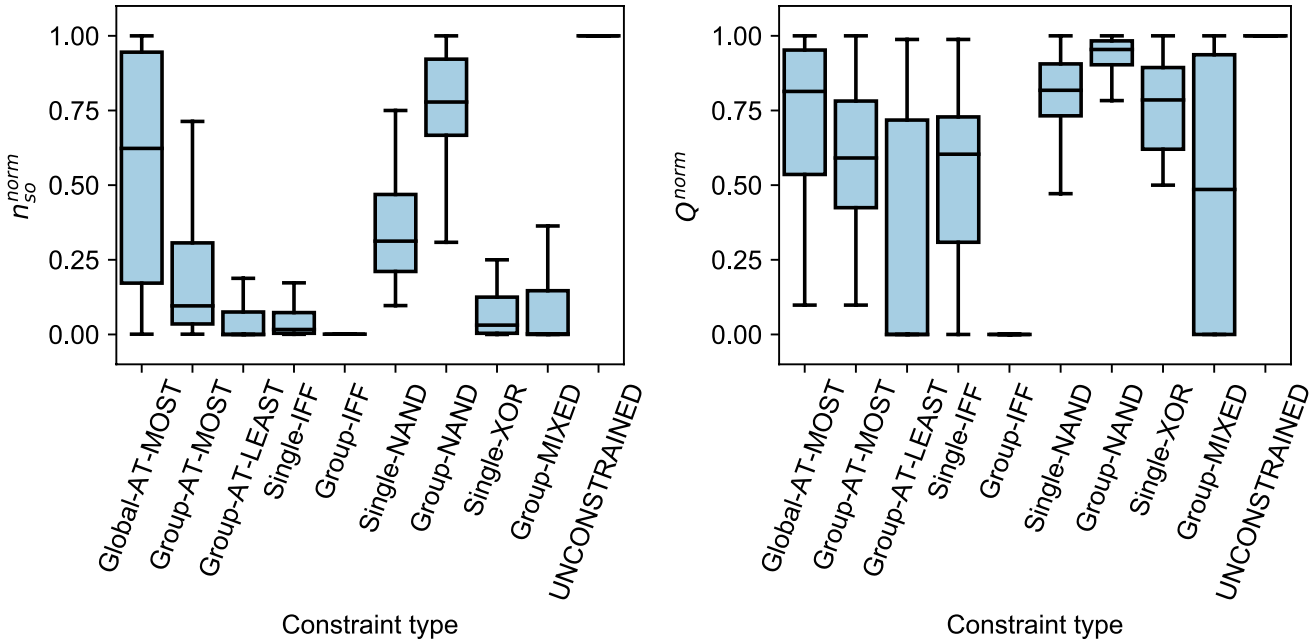


(d) Test R^2 of linear regression vs. objective value.

Fig. 3 Relationship between normalized evaluation metrics. In the scatter plots, we have sampled 1000 experimental runs to keep plot size reasonable

the type Group-MIXED (T9), the objective value shows the widest distribution. Regarding the number of solutions, Global-AT-MOST (T1) has the largest inter-quartile range. This effect is expected since we systematically vary the cardinality to cover all feature-set sizes. Single-NAND (T6) constraints show only a small decrease in objective value. This effect is even more pronounced if the same operator is applied to more features as Group-NAND (T7). The explanation is that NAND only excludes selecting all involved features simultaneously. The more features are involved in the constraint, the less restrictive the constraint becomes. In contrast, Single-IFF (T4) constraints become stronger when applied to several features in the form of Group-IIF (T5). This is because they require either all of these features to be selected or none. Remember that we combine IFF with a Global-AT-MOST (T1) constraint to prevent the trivial outcome of selecting all features. The more features are grouped in IFFs, the more difficult it becomes to satisfy the global cardinality constraint. In the worst case, the set of selected features is empty, yielding an objective value of zero.

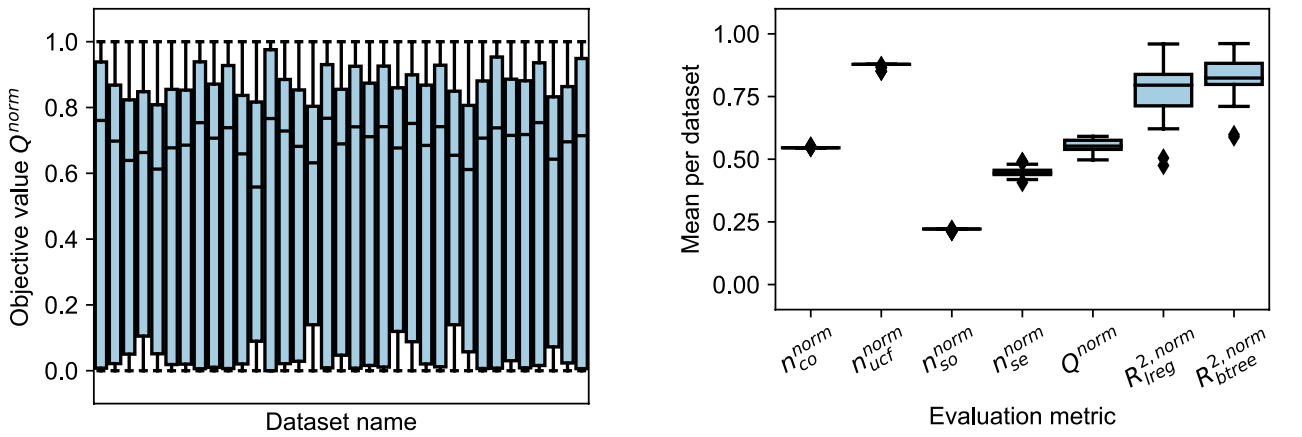
We do not show the plots for prediction performance grouped by constraint type here. Namely, if we normalize prediction performance per dataset and prediction model, the picture is similar to the one for normalized objective values in Fig. 4b. Finally, one could also compute the mean of each evaluation metric per constraint type and dataset. Such a procedure averages out randomness over the iterations of the generation process but still allows to analyze variation over constraint types or datasets. If we conduct such a procedure, there is still a considerable variation between constraint types but low variance within each constraint type, i.e., between datasets.



(a) Number of solutions.

(b) Objective value.

Fig. 4 Distribution of normalized evaluation metrics per constraint type. To keep the plots readable, we exclude outlier points from plotting



(a) Normalized objective value Q^{norm} per dataset.

(b) Mean normalized evaluation metrics per dataset.

Fig. 5 Comparison of datasets

Comparison of Datasets (Q3)

We could compare datasets in the same way as constraint types. However, evaluation metrics can vary considerably within each dataset, making the comparison between them difficult. Figure 5a shows this for the objective value. Except for prediction performance, all evaluation metrics vary more within datasets than between them. This observation makes sense for two reasons. First, we have already seen that the impact of the constraint type varies considerably. Second, there are multiple sources of randomness in the generation process for each type. Thus, we take the average of each evaluation metric per dataset in Fig. 5b and compare this over datasets. Now there is a slight variation between datasets. Our constraint-generation mechanism, which is the same for each dataset, is the cause for this. Repetition averages out randomness in the generation process. So it is expected that there nearly is no variation over datasets for

the number of constraints $n_{\text{co}}^{\text{norm}}$, the number of unique constrained features $n_{\text{ucf}}^{\text{norm}}$, and the number of solutions $n_{\text{so}}^{\text{norm}}$. These three metrics solely depend on the generation process. In contrast, the number of selected features $n_{\text{se}}^{\text{norm}}$ and the objective value Q^{norm} do not only depend on the generation but also feature qualities. For each dataset, the distribution of feature qualities may vary. For example, qualities of features might be relatively uniform in one dataset and heavily skewed in another one. In the first case, it does not matter which features are part of a constraint; only the strength of the constraint itself affects the objective value. In the second case, the features involved in a constraint significantly impact the objective value. To illustrate, adding a XOR between two highly relevant features decreases the objective value, while a XOR between two unimportant features does not affect it much. However, as Fig. 5b shows, the mean objective value per dataset does not vary much between datasets in our experiments. A reason for this could be the large number of repetitions in constraint generation. With many repetitions, lots of different feature sets take part in constraints, and the effect of the imbalanced distribution of feature qualities is averaged out. An averaging effect might also cause the number of selected features to only vary slightly between datasets. The most considerable difference between datasets occurs for prediction performance $R^{2,\text{norm}}$. Our explanation is the earlier observation that prediction performance is only moderately correlated with the constrained optimization problem.

Summary

We observe a trade-off between the size of the solution space $n_{\text{so}}^{\text{norm}}$ and the objective value Q^{norm}. Both the number of selected features $n_{\text{se}}^{\text{norm}}$ and the objective value decrease when constraints prune solutions. However, the effect is non-linear, i.e., stronger constraints can still result in a high objective value. This observation means there may be sweet spots, i.e., high-quality feature sets that adhere to the constraints. While the objective value is strongly related to the number of solutions, the latter is costly to compute. To a lesser extent, more coarse-grained metrics like the number of constraints $n_{\text{co}}^{\text{norm}}$ and number of constrained features $n_{\text{cf}}^{\text{norm}}$ are also related to the objective value. All aforementioned effects and dependencies occur similarly for all datasets. Most of the individual evaluation metrics show a similar distribution for all datasets as well. This observation might be an effect of our systematic constraint-generation procedure. In contrast, the distribution of individual evaluation metrics strongly depends on the constraint type. This is because some types prune more solutions than other ones. Thus, it is difficult to make general statements about the exact impact of constraints. Finally, we see that prediction performance behaves differently than the objective value of the optimization problem. Prediction performance depends on the size of the solution space only moderately. However, it strongly depends on the dataset. Thus, one should not use the objective value of Eq. (1) as a proxy for actual prediction performance.

Case Study in Materials Science

Experimental Design

Scenario and Dataset

The scenario of our case study involves crystalline materials, such as most metals, on the micro-scale. This material class incorporates defect structures, so-called dislocations, within the regular crystal structure. The line-like dislocations evolve in time and space in a material volume and build complex 3D networks. These networks are mainly responsible for the permanent deformation of the material under loading. Understanding the ongoing processes during the network evolution is vital to characterize the deformation behavior. A thorough understanding enables a predictive description of crystalline materials in micro- and nano-structures.

To investigate dislocation networks, the materials scientists in our team performed extensive numerical simulations using the method of discrete dislocation dynamics [61]. In this current study, we use data from simulations representing an aluminum specimen under tensile loading. See [57] for details. The goal of analyzing the simulation data from a materials-science perspective is to find out how various properties evolve. These properties characterize the dislocation network structures if a specific force is applied and the material deforms. An example of a property is the slip system a dislocation can move in. It is defined by a crystallographic slip plane characterized by the plane containing the greatest number of atoms per area and the slip direction characterized as close-packed directions with the highest number of atoms per length. Another example is the so-called Schmid-factor. It describes the slip plane and the slip direction that resolves the highest mechanical stress affecting a slip system.

It is known in materials science that a dislocation line can react with other dislocation lines in the volume in several ways. These reaction types are the usual way of characterizing the evolution of a dislocation network. In our case study, we focus on one type of reaction, the so-called glissile reaction. The target property is the density of dislocation line segments attached to such a reaction in the evolving dislocation network. The features are other physical quantities of the system, like dislocation density, shear stress, etc. Our dataset represents different locations in the simulated

material specimen as well as different loading respectively time steps. The dataset consists of 14,903 data objects and 135 features. Each data object represents the state of the material at a particular location and time step.

Constraints

Goals The primary purpose of this case study is to evaluate the impact of constraints on a concrete use case in materials science. As the underlying physical processes are complex and not fully understood, there is no firm domain knowledge suited for constraints. However, we do have some hypotheses about relationships between features. We formulate them as constraints, and feature selection must respect them. We evaluate how the resulting feature sets and solution quality change compared to an unconstrained selection. Several outcomes are possible when considering constraints. For example, the selected features might change due to constraints, but solution quality remains similar. This outcome could indicate an alternative scientific explanation for the target variable. Instead, one might obtain a different feature set but a much lower solution quality. This outcome might indicate that the hypothesis behind the constraints is wrong. By formulating different sets of constraints and evaluating them independently, one also can compare different scientific hypotheses.

In our case study, we use domain-specific as well as domain-independent constraints. The domain-specific constraints represent the hypotheses. The domain-independent ones express preferences on the resulting feature sets. Appendix section "Case Study in Materials Science" contains the respective formalizations.

Domain-independent constraint types In preliminary experiments, we observed three phenomena that made it difficult for domain experts to interpret feature sets. We believe that these phenomena are not specific to our case study.

(P1) *Size of the feature set* We observe that, if feature sets become larger, they are more difficult to interpret for domain experts.

(P2) *Inter-feature correlation* Datasets may contain features strongly correlated with each other. For a domain expert, having such highly correlated features in the result is undesirable, at least in some situations. To illustrate, such features are somewhat likely to describe the same physical phenomenon, just measured or encoded differently. In this case, selecting one of these features is sufficient. However, the objective function in Eq. (1) considers features independently and thus ignores inter-feature correlation.

(P3) *Low-quality features* Datasets may contain features with a feature quality close to zero. Even if these features are neither pruned by other constraints nor cut during optimization, they provide little value when being in the set of selected features. However, given the exponential growth of possible feature sets, such low-quality features increase optimization time dramatically. It makes sense to remove such features manually before optimization—or to add a respective constraint, as follows: "If the quality of a feature is under a certain threshold, do not select it."

In the following, we present three constraint types we use to alleviate these phenomena.

(I1) *Global-cardinality* To counter the issue with large feature sets (P1), we apply a global cardinality constraint. In our case study, we apply limits of at most five and at most ten features. Both limits were acceptable for the domain experts in our case study. We use two limits, so we can compare how the solution quality changes from the smaller (five) to the larger (ten) cardinality.

(I2) *Inter-correlation* To deal with inter-feature correlation (P2), we use a threshold on the Pearson correlation between features. If a pair of features has an absolute Pearson correlation of at least 0.8, we select at most one of these features.

(I3) *Quality-filter* To filter low-quality features (P3), we set a quality threshold of 0.2. We remove features with a lower quality before optimization.

Domain-specific constraint types Using domain knowledge, we specify the following domain-specific constraint types. Appendix section "Case Study in Materials Science" contains the corresponding formulas.

(D1) *Schmid-group* Some physical quantities in our dataset are measured for the twelve slip systems of the material, i.e., have twelve features each. For the crystal orientation of the considered specimen, the slip systems can be divided into two groups based on the Schmid factor. We hypothesize that it should suffice to have features from at most one of these groups. Within the chosen group, an arbitrary number of features can be selected.

(D2) *Quantity-Schmid-group* The constraint (D1) goes over all physical quantities. Alternatively, one can choose between the two slip system groups for each quantity independently. With such a constraint, we hypothesize that one slip system group might be relevant for some quantities, and the other group for other quantities. This constraint makes feature selection more flexible, i.e., the hypothesis is less strict.

(D3) *Schmid-group-representative* Based on the grouping of (D1), instead of selecting features from at most one

group, one can also select at most one feature from each group. This constraint corresponds to the situation where both groups are important, but it is sufficient to pick a representative feature in each group.

(D4) *Quantity-Schmid-group-representative* We merge the ideas of (D2) and (D3): For each quantity independently, select at most one feature per slip system group.

(D5) *Plastic-strain-tensor* Six features in our dataset describe the plastic strain tensor, corresponding to six different directions in space. We hypothesize that selecting three of the six directions should be sufficient.

(D6) *Dislocation-density* In our dataset, several features are based on the same physical quantity, namely dislocation density, aggregated over the slip systems. This observation does not only apply to the standard aggregates we use in (D8) and (D10) but also further features. We hypothesize that selecting at most one such feature is sufficient.

(D7) *Plastic-strain-rate* For the physical quantity describing the plastic strain rate according to the material loading, the simulation data contains features originating from two different computation methods. We hypothesize that selecting features computed with at most one of the methods suffices.

(D8) *Aggregate* As mentioned, several physical quantities are measured in twelve slip systems individually. To get a broader picture, we can aggregate over the slip systems, e.g., minimum, maximum, median, mean, and standard deviation. However, having several aggregates of the same quantity might not provide much additional information. Thus, we establish the constraint that at most one kind of aggregate function is used.

(D9) *Quantity-aggregate* Constraint (D8) can be refined to choose the aggregate function for each physical quantity independently. This new constraint allows choosing different aggregate functions for different quantities.

(D10) *Aggregate-or-original* Aggregates describe the same physical quantities as the corresponding features they aggregate. Thus, we establish a constraint that enforces for each quantity to either select original features or aggregates or none of the two.

(D11) *Mixed* Various combinations of the constraint types above are possible, though some constraint types are redundant or compete with each other. Besides this, as our domain-specific constraint types represent scientific hypotheses, we prefer to evaluate them individually. For comparison, we still consider one combination of several constraint types: the union of (D4), (D5), (D6), (D7), (D9), and (D10).

(D12) *Unconstrained* To have a reference point, we evaluate the case without domain-specific constraints.

Combining domain-independent and domain-specific constraint types From the three domain-independent constraint types (I1), (I2), and (I3), we create two sets of constraints. Both sets involve all three types, but one uses a cardinality of five, the other one a cardinality of ten. We combine each of these two constraint sets with each of twelve sets of domain-specific constraints. So we get 24 constraint sets overall. In the evaluation of the case study, we always refer to these combinations of domain-specific with domain-independent constraint types, though we only mention the name of the domain-specific constraint types.

Evaluation Metrics

To evaluate our hypotheses expressed as constraints, we focus on solution quality and selected features. We analyze how solution quality and selected features differ between hypotheses, compared to the reference case in particular (D12). Technically, we can use the complete set of evaluation metrics as in the study with synthetic constraints. However, we only have 24 constraint evaluations here, i.e., we evaluate each set of constraints only once. This number of evaluations is much smaller than in the synthetic case, where we do 1000 evaluations for each constraint type. So statements on the relationship between evaluation metrics are less generalizable here.

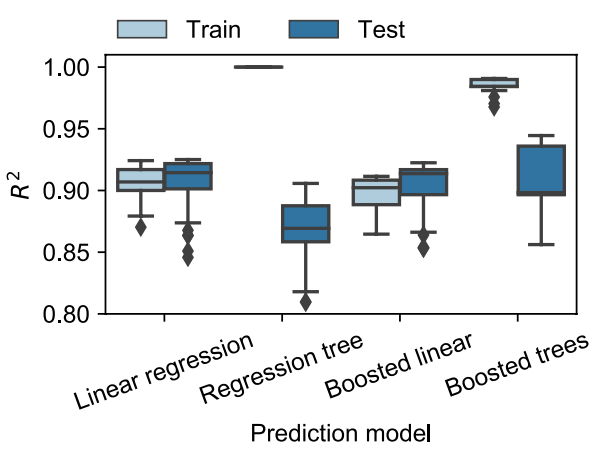
Prediction Models

We use the same prediction models and the same measure for prediction performance as in the study with synthetic constraints. As our data has a temporal dimension, we apply a time-based 80:20 holdout split instead of cross-validation, i.e., all data objects in the training data are from earlier time steps than the test data. Observe that our predictions do not go into the future but predict from feature values at one time step the target quantity at the same time step.

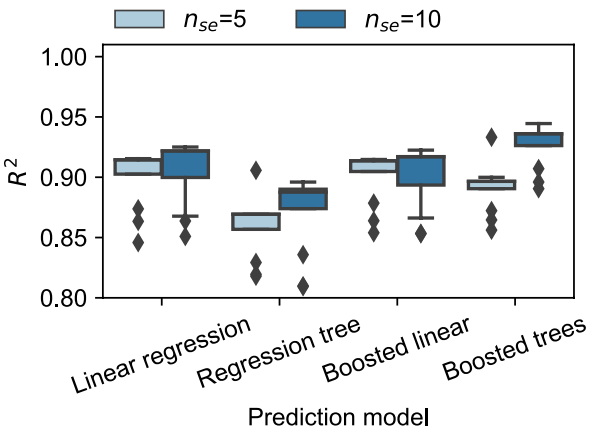
Evaluation

Solution Quality

Prediction performance Figure 6a shows the prediction performance in the case study over all 24 constraint types. All in all, prediction performance with all models is good, even with simple linear regression. The plot also shows that different constraint types cause some but not a substantial variation in prediction performance. Figure 6b shows that models with five features tend to perform slightly worse than models with ten features. The difference is relatively



(a) All data, comparing models and splits.



(b) Only test data, comparing models and cardinalities.

Fig. 6 Prediction performance in the case study, measured with R^2

small, depending on the model. We conclude that one can make good predictions with a small set of features, i.e., the scenario of our case study is suitable for feature selection.

Objective value If we look at the objective value instead of prediction performance, we observe that the selected features have high quality as well. As the variance between constraint types is low, we do not plot this and only report the numbers. For a cardinality of five, the objective value varies between 3.79 and 4.04, with a median of 4.04. For a cardinality of ten, the objective value varies between 6.45 and 7.48, with a median of 7.36. The theoretical upper bound for the objective value is 5 respectively 10. However, this theoretical upper bound assumes that all selected features have a quality of 1, which usually is not true. In our dataset, the maximum feature quality is 0.92, and the average feature quality is 0.27. So the objective values achieved are considerably higher than for randomly selecting features with average quality, i.e., $0.27 \cdot 5 = 1.35$ and $0.27 \cdot 10 = 2.7$.

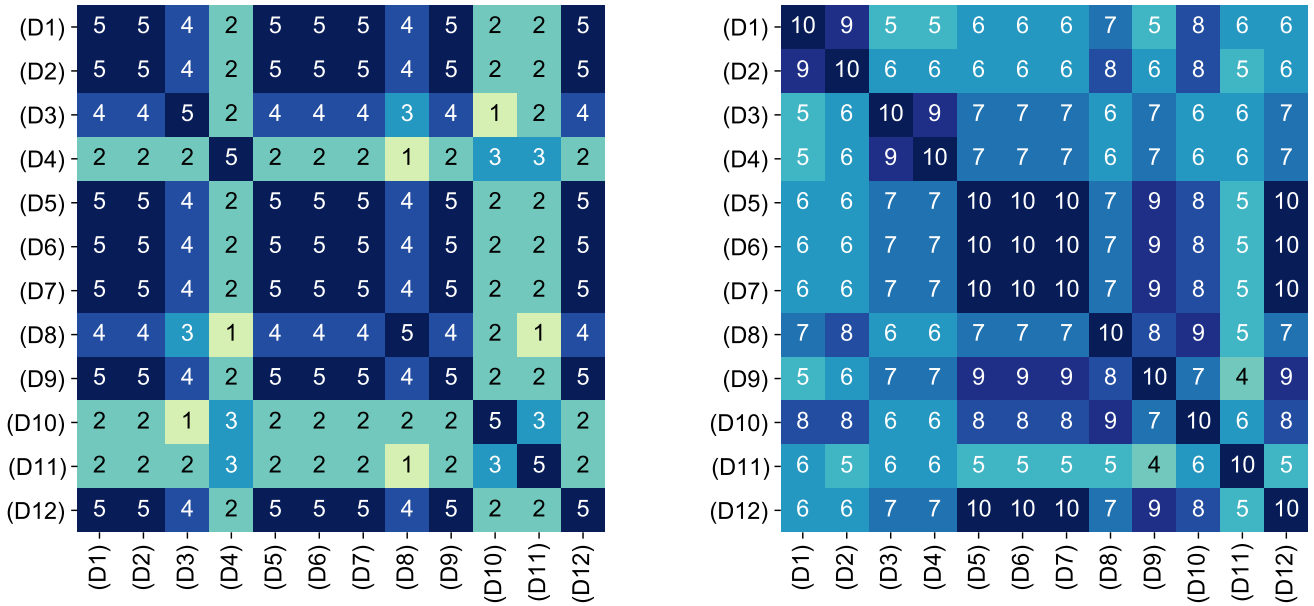
For both cardinalities, the lowest objective value is the one for constraint type Mixed (D11). This type combines six other constraint types and therefore narrows down the solution space more than individual constraint types. So the reduction in quality is plausible. Leaving aside this constraint type, the ranges of the objective value narrow down to [3.92, 4.04] and [7.03, 7.48]. Thus, in our case study, the domain-specific constraint types do not considerably impact the objective value. We will examine the potential causes and implications of this later. In contrast, cardinality does have a significant impact on the objective value. The fact that this value increases is a straightforward consequence of its monotonicity. However, the size of the increase indicates that the sixth to tenth selected features still have relatively high quality. This observation means that several features in the dataset have a moderate to strong relationship with the prediction target. This should give way to alternative feature sets with similar quality. We will analyze this later.

Comparing hypotheses As none of the domain-specific constraint types significantly decreases solution quality, none of the underlying hypotheses seems to be invalidated. On the other hand, given the modest variation of solution quality between hypotheses, it is difficult to draw conclusions regarding individual hypotheses. Thus, one should analyze the hypotheses more thoroughly with domain-specific methods. However, this is out of the scope of this current article. Nevertheless, the selected features offer insights into the domain, and our next step is to examine them in some detail.

Selected Features

Cardinality of five For all constraint types, feature sets contain several dislocation-density features, representing dislocation density on individual slip systems or in an aggregated manner. This observation is expected, as the target variable quantifies reactions of dislocations. Figure 7 shows how many selected features are the same when comparing results with different constraint types. We can make this comparison in the case study since we only have one dataset, and each constraint type is only evaluated once for each cardinality, yielding just one feature set. In the study with synthetic constraints, some aggregation strategy over datasets and repeated generations of constraints would be necessary to conduct such a feature-overlap analysis.

For a cardinality of five, several constraint types share more than half of their selected features. In particular, six constraint types yield the same feature set as the reference



(a) Cardinality of five. **(b)** Cardinality of ten.

Fig. 7 Number of common features between results with different constraint types

case, i.e., Unconstrained (D12). One could conclude that the constraints are inactive. We see two reasons for this. First, the constraints may relate to features that are part of the unconstrained result, but the constraints are consistent with that result. For example, without domain-specific constraints, only dislocation densities from one slip-system group are selected. So the constraint types Schmid-group (D1) and Quantity-Schmid-group (D2) are already satisfied in the reference case. In contrast, Schmid-group-representative (D3) and Quantity-Schmid-group-representative (D4) are violated in this case. When considering the latter two types, dislocation densities from different slip systems are selected as features. Second, some constraint types may not affect any of the selected features in the reference case. For example, Plastic-strain-rate (D7) and Plastic-strain-tensor (D5) refer to physical quantities which seem to be unimportant in the prediction scenario. In future work, one might apply an iterative approach to avoid inactive constraints, as follows: inspect the result without domain-specific constraints, formulate constraints, inspect the results, etc. This procedure should allow focusing the intellectual formulation and evaluation effort on constraints with an actual impact on feature selection.

Cardinality of ten For a cardinality of ten, the picture becomes a bit more diverse. Most pairs of constraint types still have at least half the features in common. However, there are fewer cases where all features are the same. This is because some constraints consistent with the reference case before are now violated. Still, as observed in the previous section, objective value and prediction quality are pretty similar between the constraint types. For example, type Mixed (D11) results in swapping 3/5 or 5/10 features compared to Unconstrained (D12). At the same time, the objective value only drops by 6.2% or 13.8%, which is a much smaller share. This observation indicates that there are alternative feature combinations that differ from the reference case but still yield a similar quality. Thus, the analyzed prediction scenario shows a certain 'robustness' against constraints. More generally speaking, these findings indicate that constraints can be used to implement the 'alternate' paradigm in feature selection.

Summary

We observe that our domain-specific constraints have a relatively small impact on the quality of feature sets. In other words, our analysis does not yield any evidence against our domain-specific hypotheses. So the hypotheses should be analyzed further with domain-specific methods. Another observation is that there are many features with high quality in our prediction scenario. This has two implications. First, for a high prediction quality, it tends to suffice to select a few features. This observation gives way to small and understandable prediction models. Second, in the optimization problem, constraints result in features being replaced with alternatives of similar quality. Additionally, we observe that several constraints are already satisfied in the solution of the reference case, i.e., without domain-specific constraints. In such situations, constraints do not have any impact. So it is

not possible to arrive at any insights regarding the underlying hypotheses. For future research on constrained feature selection, we recommend an iterative approach of inspecting solutions and formulating new constraints instead of formulating all constraints upfront.

Conclusions and Future Work

Problem and Approach

Feature selection helps to obtain small and understandable yet accurate prediction models. Vanilla feature-selection techniques do not consider any domain knowledge. Constraints allow incorporating a broad range of such knowledge into feature selection, e.g., physical laws, hypotheses, etc. This begs the question of how constraints affect feature-selection results. While using constraints in feature selection has been a topic for years, the focus has been on integrating specific types of constraints into feature-selection algorithms.

In this article, we have systematically evaluated the impact of constraints on the results of filter feature selection. We have empirically examined how the quality of selected features depends on constraints. To do so, we have compiled various constraint types, and we have also combined them. We have conducted two empirical studies: The first one is broad, with constraints generated systematically on datasets from various domains. Second, there is a case study with domain-specific constraints on data from materials science.

Results and Discussion

Constraint-Quality Trade-Off

We have seen that, while constraints generally reduce the quality of feature sets, this effect is non-linear. In particular, we observed sweet spots, i.e., high-quality feature sets adhering to the constraints. Finding such feature sets is of vital interest for domain experts wanting to do constrained feature selection. We have also observed that the constraints' impact strongly depends on the constraint type. In other words, the existence of such sweet spots is not guaranteed for arbitrary constraints. This calls for further studies to assess the exact impact of domain-specific constraints. When systematically generating the same constraint types for different datasets, we noted that high-level observations on the impact of constraints generalize over datasets.

Besides the constraints, the type of feature-selection technique might influence the impact of constraints as well. While we focused on a filter-feature-selection technique, other feature selectors might behave differently if equipped with constraints. In particular, the objective value of our filter technique was only moderately correlated to prediction performance with the feature set. Analyzing wrapper feature selectors might bring the objective of feature selection closer to actual prediction performance. However, wrappers are also much more computationally expensive than filters, and can usually not directly be integrated into white-box solvers.

Alternative Feature Sets

In our case study in materials science, we have noticed that most of our domain-specific constraint types did not significantly affect the resulting feature sets' quality. Instead, some constraints have yielded alternative feature sets with similar quality. This indicates that constraints allow finding alternative solutions that are interesting for domain experts. However, the alternatives in our study mainly were a by-product of domain-specific constraints. In future work, one could develop a systematic and domain-independent approach to find alternative feature sets.

Future Work

User-Centric Systems

In the future, bringing our findings closer to users will be essential, i.e., making strides towards user-centric systems. So far, we have hard-coded the constraints for our studies. In the case study, we did this after talking to domain experts. For better usability, we envision tools where users can formulate and evaluate constraints themselves. For example, users could input a file with constraints, formulated in a domain-specific language, or edit constraints via a graphical user interface.

Iterative Constraint Formulation

As seen in our case study, formulating and evaluating constraints iteratively might be beneficial. Systems for constrained feature selection could support this as well. However, depending on the dataset and the specific constraints, constraint evaluation might take too long for real-time interaction, so more work is needed here. Ideally, a system could guide the user towards promising constraints, e.g., using heuristics to estimate the impact of constraints on feature-set quality.

Soft Constraints

Currently, all constraints are hard, i.e., they all have to be fulfilled. As we have observed, this can result in reduced quality of feature sets or even an empty solution space. With soft constraints, users could attach individual penalties to the violation of each constraint. In other words, they could specify how to

trade off constraint satisfaction against feature-set quality. As an alternative approach, one could treat constraint satisfaction as another optimization objective besides feature quality and apply multi-objective optimization [22, 53]. Similar to our current evaluation, there is no systematic study of how soft constraints affect feature selection, though there is work on considering soft constraints algorithmically.

Constrained Feature Engineering

In our current studies, we took all datasets as-is, i.e., assuming all features were engineered before formulating constraints. However, one could integrate constraints into feature-engineering approaches as well. In that area, constraints would steer the creation of new features rather than the selection of existing ones. For example, one could desire to only engineer features with specific characteristics or to only allow combining certain feature-engineering operators.

Appendix

Formulas of Evaluation Metrics

Here, we provide formulas for the metrics described in "Evaluation Metrics". We build on the notation introduced in "Objective Function". For each metric, we give two formulas, one not normalized, one normalized.

Number of constraints Let C be a set of constraints. The number of constraints is the size of this set. To normalize this metric, let n_{co}^{max} be the maximum n_{co} over all experiments, i.e., datasets, constraint types and repetitions of constraint generation.

$$n_{\text{co}} = |C| \tag{3}$$

$$n_{\text{co}}^{\text{norm}} = \frac{n_{\text{co}}}{n_{\text{co}}^{max}} \tag{4}$$

Number of features involved in constraints Let $features(c)$ be a function that returns a list with the names of all features that appear in the formula of constraint c. We iterate over all constraints and sum up the number of involved features. Features appearing multiple times count multiple times. To normalize this metric, we divide by the total number of features in the dataset. Note that this normalized metric might assume values greater than one.

$$n_{\text{cf}} = \sum_{c \in C} |\text{features}(c)| \tag{5}$$

$$n_{\text{cf}}^{\text{norm}} = \frac{n_{\text{cf}}}{n} \tag{6}$$

Number of unique features involved in constraints In contrast to the previous metric, we first collect all features involved in any of the constraints and then make sure we count each feature only once. To normalize this metric, we divide by the total number of features in the dataset.

$$n_{\text{ucf}} = |\bigcup_{c \in C} \text{features}(c)| \tag{7}$$

$$n_{\text{ucf}}^{\text{norm}} = \frac{n_{\text{ucf}}}{n} \tag{8}$$

Number of solutions Each constraint $c(s) \in C$ evaluates to zero (false) or one (true), depending on the feature-selection decisions s. To determine the number of solutions, one needs to count how many values of s result in all constraints being satisfied. Mathematically, all constraints are satisfied if the minimum of all constraint evaluations equals one. To normalize this metric, we divide by the total number of possible feature sets, which is 2^n for n features.

$$n_{\text{so}} = \sum_{s \in \{0,1\}^n} \min_{c \in C} c(s) \tag{9}$$

$$n_{\text{so}}^{\text{norm}} = \frac{n_{\text{so}}}{2^n} \tag{10}$$

Number of selected features Mathematically, the number of selected features is the sum over all selection decisions s_j. To normalize this metric, we divide by the total number of features in the corresponding dataset.

$$n_{\text{se}} = \sum_{j=1}^{n} s_j \tag{11}$$

$$n_{\text{se}}^{\text{norm}} = \frac{n_{\text{se}}}{n} \tag{12}$$

Objective value The objective value is the result of the objective function in Eq. (1). To normalize this metric, we divide by the value achievable in the unconstrained problem. One gets the latter value by selecting all features, i.e., summing up all feature qualities.

$$Q(s, X, y) = \sum_{j=1}^{n} s_j \cdot |\rho_{f_j, y}| \tag{13}$$

$$Q(s, X, y)^{\text{norm}} = \frac{Q(s, X, y)}{\sum_{j=1}^{n} |\rho_{f_j, y}|} \tag{14}$$

Prediction performance Let $\hat{y} \in \mathbb{R}^m$ denote the prediction for a dataset, while the true target values are $y \in \mathbb{R}^m$. Technically speaking, y and $\hat{y}$ do not refer to a whole dataset, but either the training split or the test split in cross-validation.

Let $\bar{y}$ be the mean of the target variable. To evaluate prediction performance, we use the proportion of variance explained, R^2, which is a typical metric for regression problems [31]. *lreg* as subscript denotes linear regression as the prediction model, and *btree* stands for boosted trees.

Though R^2 already has a natural upper bound of one, it might vary considerably between datasets and prediction models. We analyze this behavior more closely in "Comparison of Prediction Performance". Thus, for some evaluations, we partition the results by prediction model and split of a dataset. In each partition, we record the minimum and maximum R^2, i.e., we aggregate over constraint types and repetitions of constraint generation. With these aggregates, we perform min-max normalization.

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{m}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{m}(y_i-\bar{y})^2} \tag{15}$$

$$R^{2,\text{norm}}(y,\hat{y}) = \frac{R^2(y,\hat{y}) - R^{2,\text{min}}_{\text{model,split}}}{R^{2,\text{max}}_{\text{model,split}} - R^{2,\text{min}}_{\text{model,split}}} \tag{16}$$

Formulas of Constraint Types

Here, we formalize the constraint types from both our studies. We build on the notation introduced in "Objective Function". Note that the actual implementation in our Python code might use different logical and arithmetic operators but is logically equivalent.

Study with Synthetic Constraints

Here, we provide formulas for the constraint types described in "Constraint Generation". For illustration purposes, we sometimes give two equivalent definitions. Each constraint type either refers to two features, a group of n' features, or all n features. Note that some constraint types have an additional parameter k.

(T1) *Global-AT-MOST*

$$\text{Global-AT-MOST}(s_1,\dots,s_n,k) = \sum_{j=1}^{n} s_j \le k, \text{ with } k \in \{1,\dots,n-1\} \tag{17}$$

(T2) *Group-AT-MOST*

$$\text{Group-AT-MOST}(s_1,\dots,s_{n'},k) = \sum_{j=1}^{n'} s_j \le k, \text{ with } k \in \{1,\dots,n'-1\} \tag{18}$$

(T3) *Group-AT-LEAST*

$$\text{Group-AT-LEAST}(s_1,\dots,s_{n'},k) = \sum_{j=1}^{n'} s_j \ge k, \text{ with } k \in \{1,\dots,n'-1\} \tag{19}$$

(T4) *Single-IFF*

$$\text{Single-IFF}(s_1,s_2) = s_1 \leftrightarrow s_2 = (s_1 \wedge s_2) \vee (\neg s_1 \wedge \neg s_2) \tag{20}$$

(T5) *Group-IFF*

$$\text{Group-IFF}(s_1,\dots,s_{n'}) = s_1 \leftrightarrow s_2 \leftrightarrow \cdots \leftrightarrow s_{n'} = (s_1 \wedge s_2 \wedge \cdots \wedge s_{n'}) \vee (\neg s_1 \wedge \neg s_2 \wedge \cdots \wedge \neg s_{n'}) \tag{21}$$

(T6) *Single-NAND*

$$\text{Single-NAND}(s_1,s_2) = \neg(s_1 \wedge s_2) \tag{22}$$

(T7) *Group-NAND*

$$\text{Group-NAND}(s_1,\dots,s_{n'}) = \neg(s_1 \wedge s_2 \wedge \cdots \wedge s_{n'}) = \sum_{j=1}^{n'} s_j \le n'-1 \tag{23}$$

(T8) *Single-XOR*

$$\text{Single-XOR}(s_1,s_2) = s_1 \oplus s_2 = (s_1 \wedge \neg s_2) \vee (\neg s_1 \wedge s_2) \tag{24}$$

(T9) *Group-MIXED* For each constraint to be generated, choose with equal probability between the following:

- Group-AT-MOST (T2) (Eq. 18)
- Group-AT-LEAST (T3) (Eq. 19)
- Group-IFF (T5) (Eq. 21)
- Group-NAND (T7) (Eq. 23)
- Single-XOR (T8) (Eq. 24)

(T10) *UNCONSTRAINED* This constraint is always satisfied.

$$\text{UNCONSTRAINED}(s_1,\dots,s_n) = \textit{True} \tag{25}$$

Case Study in Materials Science

Here, we provide formulas for the constraint types described in "Constraints".

(I1) *Global-cardinality* This is just a special case of (T1) in Eq. (17).

$$\text{Global-cardinality}(s_1,\dots,s_n,k) = \sum_{j=1}^{n} s_j \le k, \text{ with } k \in \{5,10\} \tag{26}$$

(I2) *Inter-correlation* For this constraint type, one needs to pre-compute the Pearson correlation $\rho_{f_{j_1} f_{j_2}}$ for each pair of features f_{j_1}, f_{j_2}. After computation, these values are numeric

constants in the formula. The correlation threshold τ is a numeric constant as well.

$$\begin{aligned}&\text{Inter-correlation}(s_1, \dots, s_n, \tau)\\ &= \bigwedge_{\substack{(j_1,j_2) \in \{1,\dots,n\}^2\\ j_1 \neq j_2}} \left(\left(|\rho_{f_{j_1} f_{j_2}}| \geq \tau \right) \rightarrow \neg(s_{j_1} \wedge s_{j_2}) \right)\\ &\text{with } \tau = 0.8\end{aligned} \tag{27}$$

(I3) *Quality-filter* For this constraint type, the feature-target correlations $|\rho_{f_j,y}|$ and the quality threshold τ are numeric constants.

$$\begin{aligned}&\text{Quality-filter}(s_1, \dots, s_n, \tau)\\ &= \bigwedge_{j=1}^{n} \left(\left(|\rho_{f_j,y}| < \tau \right) \rightarrow \neg s_j \right), \text{ with } \tau = 0.2\end{aligned} \tag{28}$$

(D1) *Schmid-group* Based on the Schmid factor, one can partition the twelve slip systems into non-overlapping groups. Let $\mathbb{G}$ be the partitioning, i.e., a set of sets. In our case study, $|\mathbb{G}| = 2$. Let P be the set of physical quantities measured for the twelve slip systems. Let subscript p, g identify a feature derived for a physical quantity p and a slip system g.

$$\text{Schmid-group}(s_1, \dots, s_n) = \sum_{G \in \mathbb{G}} \left(\bigvee_{p \in P,\ g \in G} s_{p,g} \right) \leq 1 \tag{29}$$

(D2) *Quantity-Schmid-group* Notation is the same as for Eq. (29).

$$\begin{aligned}&\text{Quantity-Schmid-group}(s_1, \dots, s_n)\\ &= \bigwedge_{p \in P} \left(\sum_{G \in \mathbb{G}} \left(\bigvee_{g \in G} s_{p,g} \right) \leq 1 \right)\end{aligned} \tag{30}$$

(D3) *Schmid-group-representative* Notation is the same as for Eq. (29).

$$\begin{aligned}&\text{Schmid-group-representative}(s_1, \dots, s_n)\\ &= \bigwedge_{G \in \mathbb{G}} \left(\sum_{p \in P,\ g \in G} s_{p,g} \leq 1 \right)\end{aligned} \tag{31}$$

(D4) *Quantity-Schmid-group-representative* Notation is the same as for Eq. (29).

$$\begin{aligned}&\text{Quantity-Schmid-group-repr.}(s_1, \dots, s_n)\\ &= \bigwedge_{p \in P,\ G \in \mathbb{G}} \left(\sum_{g \in G} s_{p,g} \leq 1 \right)\end{aligned} \tag{32}$$

(D5) *Plastic-strain-tensor* Let T be a set of indices, identifying features that describe the plastic strain tensor.

$$\text{Plastic-strain-tensor}(s_1, \dots, s_n) = \sum_{t \in T} s_t \leq 3 \tag{33}$$

(D6) *Dislocation-density* Let D be a set of indices, identifying features that describe dislocation density, aggregated over all slip systems.

$$\text{Dislocation-density}(s_1, \dots, s_n) = \sum_{d \in D} s_d \leq 1 \tag{34}$$

(D7) *Plastic-strain-rate* Let R_1 and R_2 be sets of indices, identifying features originating from the first and second computation method of the plastic strain rate, respectively.

$$\begin{aligned}&\text{Plastic-strain-rate}(s_1, \dots, s_n)\\ &= \left(\bigvee_{r \in R_1} s_r \right) + \left(\bigvee_{r \in R_2} s_r \right) \leq 1\end{aligned} \tag{35}$$

(D8) *Aggregate* Let P be the set of physical quantities measured for the twelve slip systems. Let A be the set of aggregate functions. Let subscript p, a identify a feature derived for a physical quantity p and an aggregate function a.

$$\text{Aggregate}(s_1, \dots, s_n) = \sum_{a \in A} \left(\bigvee_{p \in P} s_{p,a} \right) \leq 1 \tag{36}$$

(D9) *Quantity-aggregate* Notation is the same as for Eq. (36).

$$\text{Quantity-aggregate}(s_1, \dots, s_n) = \bigwedge_{p \in P} \left(\sum_{a \in A} s_{p,a} \leq 1 \right) \tag{37}$$

(D10) *Aggregate-or-original* Additional to the notation for Eq. (36), let subscript p, l identify a feature derived for a physical quantity p and a slip system $l \in \{1, \dots, 12\}$.

$$\begin{aligned}&\text{Aggregate-or-original}(s_1, \dots, s_n)\\ &= \bigwedge_{p \in P} \left(\left(\bigvee_{a \in A} s_{p,a} \right) + \left(\bigvee_{l \in \{1,\dots,12\}} s_{p,l} \right) \leq 1 \right)\end{aligned} \tag{38}$$

(D11) *Mixed* Several constraint types have to be satisfied at the same time.

$$\begin{aligned}&\text{Mixed}(s_1, \dots, s_n)\\ &= \text{Quantity-Schmid-group-repr.}(s_1, \dots, s_n) \wedge\\ &\quad \text{Plastic-strain-tensor}(s_1, \dots, s_n) \wedge\\ &\quad \text{Dislocation-density}(s_1, \dots, s_n) \wedge\\ &\quad \text{Plastic-strain-rate}(s_1, \dots, s_n) \wedge\\ &\quad \text{Quantity-aggregate}(s_1, \dots, s_n) \wedge\\ &\quad \text{Aggregate-or-original}(s_1, \dots, s_n)\end{aligned} \tag{39}$$

(D12) *Unconstrained* This is the same definition as for (T10) in Eq. (25).

$$\text{Unconstrained}(s_1, \ldots, s_n) = \text{True} \tag{40}$$

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the Ministry of Science, Research and Arts Baden-Württemberg (Az: 33-7533.-9-10/20/1).

Availability of Data and Materials All experimental data are available online at https://doi.org/10.5445/IR/1000148891.

Code Availability The code is available online at https://github.com/Jakob-Bach/Constrained-Filter-Feature-Selection.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Agrawal A, Deshpande PD, Cecen A, et al. Exploration of data science techniques to predict fatigue strength of steel from composition and processing parameters. Integr Mater Manuf Innov. 2014;3(1):90–108. https://doi.org/10.1186/2193-9772-3-8.
2. Agrawal P, Abutarboush HF, Ganesh T, et al. Metaheuristic algorithms on feature selection: a survey of one decade of research (2009–2019). IEEE Access. 2021;9:26766–91. https://doi.org/10.1109/ACCESS.2021.3056407.
3. Alfonso EM, Manthey N. New CNF features and formula classification. In: Proc. PoS@SAT; 2014. p. 57–71. https://doi.org/10.29007/b8t1
4. Bae E, Bailey J. COALA: a novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In: Proc. ICDM; 2006. p. 53–62. https://doi.org/10.1109/ICDM.2006.37
5. Barrett C, Tinelli C. Satisfiability modulo theories. In: Handbook of model checking, chap 11. Springer; 2018. p. 305–343. https://doi.org/10.1007/978-3-319-10575-8_11
6. Benavides D, Segura S, Ruiz-Cortés A. Automated analysis of feature models 20 years later: a literature review. Inf Syst. 2010;35(6):615–36. https://doi.org/10.1016/j.is.2010.01.001.
7. Carvalho DV, Pereira EM, Cardoso JS. Machine learning interpretability: a survey on methods and metrics. Electronics. 2019. https://doi.org/10.3390/electronics8080832.
8. Chandrashekar G, Sahin F. A survey on feature selection methods. Comput Electr Eng. 2014;40(1):16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024.
9. Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proc. KDD; 2016. p. 785–794. https://doi.org/10.1145/2939672.2939785
10. Childs CM, Washburn NR. Embedding domain knowledge for machine learning of complex material systems. MRS Commun. 2019;9(3):806–20. https://doi.org/10.1557/mrc.2019.90.
11. Cook SA. The complexity of theorem-proving procedures. In: Proc. STOC; 1971. p. 151–158. https://doi.org/10.1145/800157.805047
12. Dao TBH, Duong KC, Vrain C. A declarative framework for constrained clustering. In: Proc. ECML PKDD; 2013. p. 419–434. https://doi.org/10.1007/978-3-642-40994-3_27
13. De Moura L, Bjørner N. Z3: an efficient SMT solver. In: Proc. TACAS; 2008. p. 337–340. https://doi.org/10.1007/978-3-540-78800-3_24
14. Dhal P, Azad C. A comprehensive survey on feature selection in the various fields of machine learning. Appl Intell. 2021;52(4):4543–81. https://doi.org/10.1007/s10489-021-02550-9.
15. Friedman J, Hastie T, Tibshirani R. A note on the group lasso and a sparse group lasso; 2010. arXiv:1001.0736 [math.ST]
16. Galindo JA, Benavides D, Trinidad P, et al. Automated analysis of feature models: Quo vadis? Computing. 2019;101(5):387–433. https://doi.org/10.1007/s00607-018-0646-1.
17. Gilpin LH, Bau D, Yuan BZ, et al. Explaining explanations: an overview of interpretability of machine learning. In: Proc. DSAA; 2018. p. 80–89. https://doi.org/10.1109/DSAA.2018.00018
18. Grossi V, Romei A, Turini F. Survey on using constraints in data mining. Data Min Knowl Disc. 2017;31(2):424–64. https://doi.org/10.1007/s10618-016-0480-z.
19. Groves WC. Toward automating and systematizing the use of domain knowledge in feature selection. Ph.D. thesis, University of Minnesota; 2015. https://hdl.handle.net/11299/175444
20. Guo J, White J, Wang G, et al. A genetic algorithm for optimized feature selection with resource constraints in software product lines. J Syst Softw. 2011;84(12):2208–21. https://doi.org/10.1016/j.jss.2011.06.026.
21. Guo J, Zulkoski E, Olaechea R, et al. Scaling exact multi-objective combinatorial optimization by parallelization. In: Proc. ASE; 2014. p. 409–420. https://doi.org/10.1145/2642937.2642971
22. Guo J, Liang JH, Shi K, et al. SMTIBEA: a hybrid multi-objective optimization algorithm for configuring large constrained software product lines. Softw Syst Model. 2019;18(2):1447–66. https://doi.org/10.1007/s10270-017-0610-0.
23. Guyon I, Elisseeff A. An introduction to variable and feature selection. J Mach Learn Res 2003;3(Mar):1157–1182. https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf
24. Harman M, Jia Y, Krinke J, et al. Search based software engineering for software product line engineering: a survey and directions for future work. In: Proc. SPLC; 2014. p. 5–18. https://doi.org/10.1145/2648511.2648513
25. Henard C, Papadakis M, Harman M, et al. Combining multi-objective search and constraint solving for configuring large software product lines. In: Proc. ICSE; 2015. pp 517–528. https://doi.org/10.1109/ICSE.2015.69
26. Hijazi S, Hamad D, Kalakech M, et al. Active learning of constraints for weighted feature selection. Adv Data Anal Classif. 2021;15(2):337–77. https://doi.org/10.1007/s11634-020-00408-5.
27. Huan TD, Mannodi-Kanakkithodi A, Ramprasad R. Accelerated materials property predictions and design using motif-based fingerprints. Phys Rev B. 2015;92(1):014106. https://doi.org/10.1103/PhysRevB.92.014106.
28. Imbalzano G, Anelli A, Giofré D, et al. Automatic selection of atomic fingerprints and reference configurations for machine-learning potentials. J Chem Phys. 2018;148(24):241730. https://doi.org/10.1063/1.5024611.

29. Jacob L, Obozinski G, Vert JP. Group lasso with overlap and graph lasso. In: Proc. ICML; 2009. p. 433–440. https://doi.org/10.1145/1553374.1553431
30. Jagdhuber R, Lang M, Stenzl A, et al. Cost-constrained feature selection in binary classification: adaptations for greedy forward selection and genetic algorithms. BMC Bioinform. 2020. https://doi.org/10.1186/s12859-020-3361-9.
31. James G, Witten D, Hastie T, et al. Linear regression. In: An introduction to statistical learning: with applications in R, chap 3. Springer; 2013. p. 59–126. https://doi.org/10.1007/978-1-4614-7138-7_3
32. Janet JP, Kulik HJ. Resolving transition metal chemical space: Feature selection for machine learning and structure–property relationships. J Phys Chem A. 2017;121(46):8939–54. https://doi.org/10.1021/acs.jpca.7b08750.
33. Karpatne A, Atluri G, Faghmous JH, et al. Theory-guided data science: a new paradigm for scientific discovery from data. IEEE Trans Knowl Data Eng. 2017;29(10):2318–31. https://doi.org/10.1109/TKDE.2017.2720168.
34. Khushaba RN, Al-Ani A, Al-Jumaily A. Feature subset selection using differential evolution and a statistical repair mechanism. Expert Syst Appl. 2011;38(9):11515–26. https://doi.org/10.1016/j.eswa.2011.03.028.
35. Lagani V, Athineou G, Farcomeni A, et al. Feature selection with the R package MXM: discovering statistically equivalent feature subsets. J Stat Softw. 2017;80(7):1–25. https://doi.org/10.18637/jss.v080.i07.
36. Lee J, Seo W, Kim DW. Effective evolutionary multilabel feature selection under a budget constraint. Complexity. 2018. https://doi.org/10.1155/2018/3241489.
37. Li J, Cheng K, Wang S, et al. Feature selection: a data perspective. ACM Comput Surv. 2017;50(6):1–45. https://doi.org/10.1145/3136625.
38. Liu Y, Wu JM, Avdeev M, et al. Multi-layer feature selection incorporating weighted score-based expert knowledge toward modeling materials with targeted properties. Adv Theor Simul. 2020;3(2):1900215. https://doi.org/10.1002/adts.201900215.
39. Lu G, Li B, Yang W, et al. Unsupervised feature selection with graph learning via low-rank constraint. Multimed Tools Appl. 2018;77(22):29531–49. https://doi.org/10.1007/s11042-017-5207-7.
40. LVAT Linux variability analysis tools. 2021. https://code.google.com/archive/p/linux-variability-analysis-tools/. Accessed 10 Aug 2021
41. Mangal A, Holm EA. A comparative study of feature selection methods for stress hotspot classification in materials. Integr Mater Manuf Innov. 2018;7(3):87–95. https://doi.org/10.1007/s40192-018-0109-8.
42. Mendonca M, Branco M, Cowan D. S.P.L.O.T.: Software product lines online tools. In: Proc. OOPSLA; 2009. p. 761–762. https://doi.org/10.1145/1639950.1640002
43. Momeni N, Arza A, Rodrigues J, et al. Cafs: cost-aware features selection method for multimodal stress monitoring on wearable devices. IEEE Trans Biomed Eng. 2021. https://doi.org/10.1109/TBME.2021.3113593.
44. Ng RT, Lakshmanan LVS, Han J, et al. Exploratory mining and pruning optimizations of constrained associations rules. In: Proc. SIGMOD; 1998. p. 13–24. https://doi.org/10.1145/276305.276307
45. Nudelman E, Leyton-Brown K, Hoos HH, et al. Understanding random SAT: Beyond the clauses-to-variables ratio. In: Proc. CP; 2004. p. 438–452. https://doi.org/10.1007/978-3-540-30201-8_33
46. Ochoa L, González-Rojas O, Cardozo N, et al. Constraint programming heuristics for configuring optimal products in multi product lines. Inf Sci. 2019;474:33–47. https://doi.org/10.1016/j.ins.2018.09.042.
47. Oh J, Batory D, Myers M, et al. Finding near-optimal configurations in product lines by random sampling. In: Proc. ESEC/FSE; 2017. p. 61–71. https://doi.org/10.1145/3106237.3106273
48. Paclík P, Duin RPW, van Kempen GMP, et al. On feature selection with measurement cost and grouped features. In: Proc. SSPR/SPR; 2002. p. 461–469. https://doi.org/10.1007/3-540-70659-3_48
49. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in python. J Mach Learn Res 2011;12:2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html
50. Plasberg JH, Kleijn WB. Feature selection under a complexity constraint. IEEE Trans Multimed. 2009;11(3):565–71. https://doi.org/10.1109/TMM.2009.2012944.
51. Ramprasad R, Batra R, Pilania G, et al. Machine learning in materials informatics: recent applications and prospects. NPJ Comput Mater. 2017;3(54):1–13. https://doi.org/10.1038/s41524-017-0056-5.
52. Rostami M, Berahmand K, Forouzandeh S. A novel method of constrained feature selection by the measurement of pairwise constraints uncertainty. J Big Data. 2020;7(1):83. https://doi.org/10.1186/s40537-020-00352-3.
53. Sayyad AS, Menzies T, Ammar H. On the value of user preferences in search-based software engineering: a case study in software product lines. In: Proc. ICSE; 2013. p. 492–501. https://doi.org/10.1109/ICSE.2013.6606595
54. Serpico SB, Bruzzone L. A new search algorithm for feature selection in hyperspectral remote sensing images. IEEE Trans Geosci Remote Sens. 2001;39(7):1360–7. https://doi.org/10.1109/36.934069.
55. Sheikhpour R, Sarram MA, Gharaghani S, et al. A survey on semi-supervised feature selection methods. Pattern Recognit. 2017;64:141–58. https://doi.org/10.1016/j.patcog.2016.11.003.
56. Simon N, Friedman J, Hastie T, et al. A sparse-group lasso. J Comput Graph Stat. 2013;22(2):231–45. https://doi.org/10.1080/10618600.2012.681250.
57. Sudmanns M, Bach J, Weygand D, et al. Data-driven exploration and continuum modeling of dislocation networks. Modell Simul Mater Sci Eng. 2020;28(6):065001. https://doi.org/10.1088/1361-651x/ab97ef.
58. Thum T, Batory D, Kastner C. Reasoning about edits to feature models. In: Proc. ICSE; 2009. p. 254–264. https://doi.org/10.1109/ICSE.2009.5070526
59. Vanschoren J, Van Rijn JN, Bischl B, et al. Openml: networked science in machine learning. ACM SIGKDD Explor Newsl. 2014;15(2):49–60. https://doi.org/10.1145/2641190.2641198.
60. Wagner N, Rondinelli JM. Theory-guided machine learning in materials science. Front Mater Sci. 2016;3:28. https://doi.org/10.3389/fmats.2016.00028.
61. Weygand D, Friedman L, van der Giessen E, et al. Discrete dislocation modeling in three-dimensional confined volumes. Mater Sci Eng A. 2001;309:420–4. https://doi.org/10.1016/S0921-5093(00)01632-4.
62. White J, Schmidt DC, Benavides D, et al. Automated diagnosis of product-line configuration errors in feature models. In: Proc. SPLC; 2008. p. 225–234. https://doi.org/10.1109/SPLC.2008.16
63. Yang H, Xu Z, Lyu MR, et al. Budget constrained non-monotonic feature selection. Neural Netw. 2015;71:214–24. https://doi.org/10.1016/j.neunet.2015.08.004.
64. Yuan M, Lin Y. Model selection and estimation in regression with grouped variables. J R Stat Soc Ser B (Stat Methodol). 2006;68(1):49–67. https://doi.org/10.1111/j.1467-9868.2005.00532.x.
65. Zhang D, Chen S, Zhou ZH. Constraint score: a new filter method for feature selection with pairwise constraints. Pattern Recognit. 2008;41(5):1440–51. https://doi.org/10.1016/j.patcog.2007.10.009.

66. Zhang L, Li Y, Zhang J, et al. Nonlinear sparse feature selection algorithm via low matrix rank constraint. Multimed Tools Appl. 2019;78(23):33319–37. https://doi.org/10.1007/s11042-018-6909-1.
67. Zhang R, Zhang Y, Li X. Unsupervised feature selection via adaptive graph learning and constraint. IEEE Trans Neural Netw Learn Syst. 2020. https://doi.org/10.1109/TNNLS.2020.3042330.
68. Zhang Z, Wang Q, Si L, et al. Learning for efficient supervised query expansion via two-stage feature selection. In: Proc. SIGIR; 2016. p. 265–274. https://doi.org/10.1145/2911451.2911539
69. Zhao P, Rocha G, Yu B. Grouped and hierarchical model selection through composite absolute penalties. Tech. rep., Department of Statistics, UC Berkeley; 2006. https://statistics.berkeley.edu/sites/default/files/tech-reports/703.pdf