arXiv:2307.11607v1 [cs.LG] 21 Jul 2023

Finding Optimal Diverse Feature Sets with Alternative Feature Selection

Jakob Bach
Karlsruhe Institute of Technology (KIT), Germany
jakob.bach@kit.edu

Abstract

Feature selection is popular for obtaining small, interpretable, yet highly accurate prediction models. Conventional feature-selection methods typically yield one feature set only, which might not suffice in some scenarios. For example, users might be interested in finding alternative feature sets with similar prediction quality, offering different explanations of the data. In this article, we introduce alternative feature selection and formalize it as an optimization problem. In particular, we define alternatives via constraints and enable users to control the number and dissimilarity of alternatives. Next, we analyze the complexity of this optimization problem and show $\mathcal{NP}$-hardness. Further, we discuss how to integrate conventional feature-selection methods as objectives. Finally, we evaluate alternative feature selection with 30 classification datasets. We observe that alternative feature sets may indeed have high prediction quality, and we analyze several factors influencing this outcome.

Keywords: feature selection, alternatives, constraints, mixed-integer programming, explainability, interpretability, XAI

1 Introduction

Motivation Feature-selection methods are ubiquitous for a variety of reasons. By reducing dataset dimensionality, they lower the computational cost and memory requirements of prediction models. Next, models may generalize better after removing irrelevant and spurious predictors. Finally, prediction models may become simpler [61], improving interpretability.

Most conventional feature-selection methods only return one feature set [11]. These methods optimize a criterion of feature-set quality, e.g., prediction performance. However, besides the optimal feature set, there might be other, differently composed feature sets with similar quality. Such alternative feature sets are interesting for users, e.g., to obtain several diverse explanations. Alternative explanations can provide additional insights into predictions, enable users to develop and test different hypotheses, appeal to different kinds of users, and foster trust in the predictions [50, 108].

Problem statement This article addresses the problem of alternative feature selection, which we informally define as follows: Find multiple, sufficiently different feature sets that optimize feature-set quality. We provide formal definitions in Section 3.2. This problem entails an interesting trade-off: Depending on how different the alternatives should be, one might have to compromise on quality. In particular, a stronger dissimilarity requirement might require selecting more low-quality features in the alternatives.

Two points are essential for alternative feature selection, which we both address in this article. First, one needs to formalize and quantify what an alternative feature set is. In particular, users should be able to control the dissimilarity of alternatives and hence the aforementioned quality trade-off. Second, one needs an approach to find alternative feature sets efficiently. Ideally, the approach should be general, i.e., cover a broad range of conventional feature-selection methods, given the variety of the latter [15, 61].

Related work While finding alternative solutions has already been addressed extensively in the field of clustering [9], there is a lack of such approaches for feature selection. Only a few feature-selection methods target at obtaining multiple, diverse feature sets [11]. In particular, techniques for ensemble feature selection [92, 96] and statistically equivalent feature subsets [56] produce multiple feature sets but not optimal alternatives. These approaches do not guarantee the diversity of the feature sets, nor do they let users control diversity. In fields related to feature selection, the goal of obtaining multiple, diverse solutions has been studied as well, e.g., for subspace clustering [42, 72], subgroup discovery [59], subspace search [102], or explainable-AI techniques [2, 49, 71, 91] like counterfactuals. These approaches are not directly applicable or easily adaptable to feature selection, and most of them provide limited or no user control over alternatives, as we will elaborate in Section 4.

Contributions Our contribution is fourfold.

First, we formalize alternative feature selection as an optimization problem. In particular, we define alternatives via constraints on feature sets. This approach is orthogonal to the feature-selection method itself so that users can choose the latter according to their needs. This approach also allows integrating other constraints on feature sets, e.g., to capture domain knowledge [6, 32]. Finally, this approach lets users control the search for alternatives with two parameters, i.e., the number of alternatives and a dissimilarity threshold.

Second, we analyze the computational complexity of this optimization problem. We show $\mathcal{NP}$-hardness, even for a simple notion of feature-set quality.

Third, we discuss how to solve this optimization problem. To that end, we describe how to integrate different categories of conventional feature-selection methods in the objective function of the optimization problem.

Fourth, we evaluate alternative feature selection with comprehensive experiments. In particular, we use 30 classification datasets from the Penn Machine Learning Benchmarks (PMLB) [82, 90] and five feature-selection methods. We

focus our evaluation on the feature-set quality of the alternatives relative to our user parameters. We publish all our code[1] and experimental data[2] online.

Experimental results We observe that several factors influence the quality of alternatives, i.e., the dataset, feature-selection method, notion of feature-set quality, and parameters for searching alternatives. As expectable, feature-set quality tends to decrease with the number of alternatives and the dissimilarity threshold for alternatives. Thus, these parameters allow users to control the trade-off between dissimilarity and quality of alternatives. Also, even no valid alternative may exist if the parameter values are too strict. Computationally, a sequential search for multiple alternatives was significantly faster than a simultaneous one while yielding a similar quality. Finally, we observe that the prediction performance of feature sets may only weakly correlate with the quality assigned by feature-selection methods. In particular, seemingly bad alternatives regarding the latter might still be good regarding the former.

Outline Section 2 introduces notation and fundamentals. Section 3 describes and analyzes alternative feature selection. Section 4 reviews related work. Section 5 outlines our experimental design, while Section 6 presents the experimental results. Section 7 concludes. Appendix A contains supplementary materials.

2 Fundamentals

In this section, we introduce basic notation (cf. Section 2.1) and review different methods to measure the quality of feature sets (cf. Section 2.2).

2.1 Notation

$X \in \mathbb{R}^{m \times n}$ stands for a dataset in the form of a matrix. Each row is a data object, and each column is a feature. $F = \{f_1, \ldots, f_n\}$ is the corresponding set of feature names. We assume that categorical features have already been made numeric, e.g., via one-hot encoding. $X_{\cdot j} \in \mathbb{R}^m$ denotes the vector representation of the j-th feature. $y \in Y^m$ represents the prediction target with domain Y, e.g., $Y = \{0, 1\}$ for binary classification or $Y = \mathbb{R}$ for regression.

In feature selection, one makes a binary decision $s_j \in \{0, 1\}$ for each feature, i.e., either selects it or not. The vector $s \in \{0, 1\}^n$ combines all these selection decisions and yields the selected feature set $F_s = \{f_j \mid s_j = 1\} \subseteq F$. The function $Q(s, X, y)$ returns the quality of such a feature set. Without loss of generality, we assume that this function should be maximized.

[1] `https://github.com/Jakob-Bach/Alternative-Feature-Selection`
[2] `https://doi.org/10.35097/1623`

2.2 Measuring Feature (Set) Quality

There are different ways to evaluate feature-set quality $Q(s, X, y)$. We only give a short overview here; see [15, 61, 81] for comprehensive studies and surveys of feature selection. A conventional categorization of feature-selection methods distinguishes between filter, wrapper, and embedded methods [36].

Filter methods Filter methods evaluate feature sets without training a prediction model. Univariate filters assess each feature independently. They often assign a score to each feature, e.g., the absolute Pearson correlation or the mutual information between a feature and the prediction target. Such methods ignore potential interactions between features, e.g., redundancies. In contrast, multivariate filters evaluate feature sets as a whole. Such methods often combine a measure of feature relevance with a measure of feature redundancy. Examples include CFS [37, 38], FCBF [115], and mRMR [85].

Wrapper methods Wrapper methods [52] evaluate feature sets by training prediction models with them and measuring prediction quality. They employ a generic search strategy to iterate over candidate feature sets, e.g., genetic algorithms. Feature-set quality is a black-box function in this search.

Embedded methods Embedded methods train prediction models with built-in feature selection, e.g., decision trees [13] or random forests [12]. Thus, the criterion for feature-set quality is model-specific. For example, tree-based models often use information gain or the Gini index to select features during training.

Post-hoc feature-importance methods Apart from conventional feature selection, there are various methods that assess feature importance after training a model. These methods range from local explanation methods like LIME [87] or SHAP [63] to global importance methods like permutation importance [12] or SAGE [20]. In particular, assessing feature importance plays a crucial role in the field of machine-learning interpretability [14, 68].

3 Alternative Feature Selection

In this section, we present the problem and approaches for alternative feature selection. First, we define the overall structure of the optimization problem, i.e., objective and constraints (cf. Section 3.1). Second, we formalize the notion of alternatives via constraints (cf. Section 3.2). Third, we discuss different objective functions corresponding to different feature-set quality measures from Section 2.2. In particular, we describe how to solve the resulting optimization problem (cf. Section 3.3). Fourth, we analyze the computational complexity of the optimization problem (cf. Section 3.4).

3.1 Optimization Problem

Alternative feature selection has two goals. First, the quality of an alternative feature set should be high. Second, an alternative feature set should differ from one or more other feature set(s). There are several ways to combine these two goals in an optimization problem:

First, one can consider both goals as objectives, obtaining an unconstrained multi-objective problem. Second, one can treat feature-set quality as objective and enforce alternatives with constraints. Third, one can consider being alternative as objective and constrain feature-set quality, e.g., with a lower bound. Fourth, one can define constraints for both, feature-set quality and being alternative, searching for feasible solutions instead of optimizing.

We stick to the second formulation, i.e., optimizing feature-set quality subject to being alternative. This formulation has the advantage of keeping the original objective function of feature selection. Thus, users do not need to specify a range or a threshold on feature-set quality but can control how alternative the feature sets must be instead. We obtain the following optimization problem for a single alternative feature set F_s:

$$\begin{aligned} \max_{s} \quad & Q(s, X, y) \\ \text{subject to:} \quad & F_s \text{ being alternative} \end{aligned} \tag{1}$$

In the following, we discuss different objective functions $Q(s, X, y)$ and suitable constraints for *being alternative*. Additionally, many feature-selection methods also limit the feature-set size $|F_s|$ to a user-defined value $k \in \mathbb{N}$, which adds a further, simple constraint to the optimization problem.

3.2 Constraints – Defining Alternatives

In this section, we formalize alternative feature sets. First, we discuss the base case where an individual feature set is an alternative to another one (cf. Section 3.2.1). Second, we extend this notion to multiple alternatives, considering sequential and simultaneous search methods (cf. Section 3.2.2).

Our notion of alternatives is independent of the feature-selection method. We provide two parameters, i.e., a dissimilarity threshold τ and the number of alternatives a, allowing users to control the search for alternatives.

3.2.1 Single Alternative

We consider a feature set an alternative to another feature set if it differs sufficiently. Mathematically, we express this notion with a set-dissimilarity measure [19, 26]. These measures typically assess how strongly two sets overlap and relate this to their sizes. E.g., a well-known set-dissimilarity measure is the Jaccard distance, which is defined as follows for the feature sets F' and F'':

$$d_{\text{Jacc}}(F', F'') = 1 - \frac{|F' \cap F''|}{|F' \cup F''|} = 1 - \frac{|F' \cap F''|}{|F'| + |F''| - |F' \cap F''|} \tag{2}$$

In this article, we use a dissimilarity measure based on the Dice coefficient:

$$d_{\text{Dice}}(F', F'') = 1 - \frac{2 \cdot |F' \cap F''|}{|F'| + |F''|} \tag{3}$$

Generally, we do not have strong requirements on the set-dissimilarity measure $d(\cdot)$. Our definitions of alternatives only assume symmetry, i.e., $d(F', F'') = d(F'', F')$, and non-negativity, i.e., $d(F', F'') \geq 0$, though one could adapt them to other conditions as well. In particular, the dissimilarity measure does not need to be a metric but can also be a semi-metric [110] like $d_{\text{Dice}}(\cdot)$.

We leverage the set-dissimilarity measure for the following definition:

Definition 1 (Single alternative)**.** Given a symmetric, non-negative set-dissimilarity measure $d(\cdot)$ and a dissimilarity threshold $\tau \in \mathbb{R}_{\geq 0}$, a feature set F' is an alternative to a feature set F'' (and vice versa) if $d(F', F'') \geq \tau$.

The threshold τ controls how alternative the feature sets must be and depends on the dataset as well as user preferences. In particular, requiring strong dissimilarity may cause a significant drop in feature-set quality. Some datasets may contain many features of similar utility, thereby enabling many alternatives of similar quality, while predictions on other datasets may depend on a few key features. Only users can decide which drop in feature-set quality is acceptable as a trade-off for obtaining alternatives. Thus, we leave τ as a parameter. In case the set-dissimilarity measure $d(\cdot)$ is normalized to $[0, 1]$, like the Dice dissimilarity or Jaccard distance, the interpretation of τ is user-friendly: Setting $\tau = 0$ allows identical alternatives, while $\tau = 1$ implies zero overlap.

If the choice of τ is unclear a priori, users can try out different values and compare the resulting feature-set quality. One systematic approach is a binary search: Start with the mid-range value of $\tau = 0$, i.e., 0.5 for $\tau \in [0, 1]$. If the quality of the resulting alternative is too low, decrease τ to 0.25, i.e., allow more similarity. If the quality of the resulting alternative is acceptably high, increase τ to 0.75, i.e., check a more dissimilar feature set. Continue this procedure till an alternative with an acceptable quality-dissimilarity trade-off is found.

When implementing Definition 1, we can leverage the following proposition:

Proposition 1 (Linearity of constraints for alternatives)**.** *Using the Dice dissimilarity (cf. Equation 3), one can express alternative feature sets (cf. Definition 1) with 0-1 integer linear constraints.*

Proof. We re-arrange terms in the Dice dissimilarity (cf. Equation 3) to get rid of the quotient of set sizes:

$$\begin{aligned} d_{\text{Dice}}(F', F'') = 1 - \frac{2 \cdot |F' \cap F''|}{|F'| + |F''|} &\geq \tau \\ \Leftrightarrow |F' \cap F''| &\leq \frac{1 - \tau}{2} \cdot (|F'| + |F''|) \end{aligned} \tag{4}$$

Next, we express set sizes in terms of the feature-selection vector s:

$$\begin{aligned} |F_s| &= \sum_{j=1}^{n} s_j \\ |F_{s'} \cap F_{s''}| &= \sum_{j=1}^{n} s'_j \cdot s''_j \end{aligned} \tag{5}$$

Finally, we replace each product $s'_j \cdot s''_j$ with an auxiliary variable t_j, bound by additional constraints, to linearize it [69]:

$$\begin{aligned} t_j &\leq s'_j \\ t_j &\leq s''_j \\ 1 + t_j &\geq s'_j + s''_j \\ t_j &\in \{0, 1\} \end{aligned} \tag{6}$$

Combining Equations 4, 5, and 6, we obtain a set of constraints that only involve linear expressions of binary decision variables. In particular, there are only sum expressions and multiplications with constants but no products between variables. If one feature set is known, i.e., either s' or s'' is fixed, Equation 5 only multiplies variables with constants and is already linear without Equation 6. □

Given a suitable objective function, which we discuss later, linear constraints allow using a broad range of solvers. As an alternative formulation, one could also encode such constraints into propositional logic (SAT) [103].

If the set sizes $|F'|$ and $|F''|$ are constant, e.g., user-defined, Equation 4 implies that the threshold τ has a linear relationship to the maximum number of overlapping features $|F' \cap F''|$. This correspondence eases the interpretation of τ and makes us use the Dice dissimilarity in the following. In contrast, the Jaccard distance exhibits a non-linear relationship between τ and the overlap size, which follows from re-arranging Equation 2 in combination with Definition 1:

$$\begin{aligned} d_{\text{Jacc}}(F', F'') = 1 - \frac{|F' \cap F''|}{|F'| + |F''| - |F' \cap F''|} &\geq \tau \\ \Leftrightarrow |F' \cap F''| &\leq \frac{1-\tau}{2-\tau} \cdot (|F'| + |F''|) \end{aligned} \tag{7}$$

Further, if $|F'| = |F''|$, as in our experiments, the Dice dissimilarity (cf. Equation 4) becomes identical to several other set-dissimilarity measures [26]. The parameter τ then directly expresses which fraction of features in one set needs to differ from the other set and vice versa, which further eases interpretability:

$$d_{\text{Dice}}(F', F'') \geq \tau \Leftrightarrow |F' \cap F''| \leq (1-\tau) \cdot |F'| = (1-\tau) \cdot |F''| \tag{8}$$

Thus, if users are uncertain how to choose τ and $|F'|$ is reasonably small, they can try out all values of $\tau \in \{i/|F'|\}$ with $i \in \{1, \ldots, |F'|\}$. In particular, these $|F'|$ unique values of τ suffice to produce all possible results that one could obtain with an arbitrary $\tau \in (0, 1]$.

	Sequential search		Simultaneous search
	Alternative i	Summed	
Decision variables s	n	$(a+1) \cdot n$	$(a+1) \cdot n$
Linearization variables t	0	0	$\frac{a \cdot (a+1) \cdot n}{2}$
Alternative constraints	i	$\frac{a \cdot (a+1)}{2}$	$\frac{a \cdot (a+1)}{2}$
Linearization constraints	0	0	$\frac{3 \cdot a \cdot (a+1) \cdot n}{2}$

Table 1: Size of the optimization problem by search method, for a alternatives ($a+1$ feature sets overall) and n features.

3.2.2 Multiple Alternatives

If users desire multiple alternative feature sets rather than only one, we can determine these alternatives sequentially or simultaneously. The number of alternatives $a \in \mathbb{N}_0$ is a parameter to be set by the user. The overall number of feature sets is $a+1$ since we deem one feature set the 'original' one. Table 1 compares the sizes of the optimization problems for these two search methods.

Sequential alternatives With sequential search, users obtain several alternatives iteratively, with one feature set per iteration. We constrain this new set to be an alternative to all previously found ones, which are given in the set $\mathbb{F}$:

Definition 2 (Sequential alternative)**.** A feature set F'' is an alternative to a set of feature sets $\mathbb{F}$ (and vice versa) if F'' is a single alternative (cf. Definition 1) to each $F' \in \mathbb{F}$.

One could also think of less strict constraints, e.g., requiring only the average dissimilarity to all previously found feature sets to pass a threshold τ. However, definitions like the latter may allow some feature sets to overlap heavily or even be identical if other feature sets are very dissimilar. Thus, we require pairwise dissimilarity in Definition 2. Combining Equation 1 with Definition 2, we obtain the following optimization problem for each iteration of the search:

$$\begin{aligned} \max_{s} \quad & Q(s, X, y) \\ \text{subject to:} \quad & \forall F' \in \mathbb{F} : \; d(F_s, F') \geq \tau \end{aligned} \tag{9}$$

The objective function remains the same as for a single alternative ($|\mathbb{F}| = 1$), i.e., we only optimize the quality of one feature set at once. Thus, the number of variables in the optimization problem is independent of the number of alternatives a. Instead, we solve the optimization problem repeatedly; each alternative only adds one constraint to the problem. The first, 'original' feature set is the same as in conventional feature selection without constraints for alternatives. As we always compare only one variable feature set to existing, constant feature sets, we also do not need to introduce auxiliary variables as in Equation 6.

Thus, we expect the runtime of sequential search to scale well with the number of alternatives. Further runtime gains may arise if the solver keeps a state between iterations and can warm-start.

However, as the solution space becomes narrower over iterations, feature-set quality can deteriorate with each further alternative. In particular, multiple alternatives from the same sequential search might differ significantly in their quality. As a remedy, users can decide after each iteration if the feature-set quality is already unacceptably low or if another alternative should be found. In particular, users do not need to define the number of alternatives a a priori.

Simultaneous alternatives With simultaneous search, users obtain multiple alternatives at once, so they need to decide on the number of alternatives beforehand. We use pairwise dissimilarity constraints again:

Definition 3 (Simultaneous alternatives)**.** A set of feature sets $\mathbb{F}$ contains simultaneous alternatives if each feature set $F' \in \mathbb{F}$ is a single alternative (cf. Definition 1) to each other set $F'' \in \mathbb{F}$, $F' \neq F''$.

Combining Equation 1 with Definition 3, we obtain the following optimization problem for $a+1$ feature sets:

$$\begin{aligned} \max_{s^{(0)},\ldots,s^{(a)}} \quad & \operatorname*{agg}_{i\in\{0,\ldots,a\}} Q(s^{(i)}, X, y) \\ \text{subject to:} \quad & \forall i_1, i_2 \in \{0,\ldots,a\},\ i_1 \neq i_2 :\ d(F_{s^{(i_1)}}, F_{s^{(i_2)}}) \geq \tau \end{aligned} \tag{10}$$

In contrast to the sequential case (cf. Equation 9), we need to introduce further decision variables and modify the objective function here. The operator $\operatorname{agg}(\cdot)$ defines how to aggregate the feature-set qualities of the alternatives. In our experiments, we consider the sum as well as the minimum to instantiate $\operatorname{agg}(\cdot)$, which we refer to as *sum-aggregation* and *min-aggregation*. The latter explicitly fosters balanced feature-set qualities. Appendix A.1 discusses these two aggregation operators and additional ideas for balancing qualities in detail.

Runtime-wise, we expect simultaneous search to scale worse with the number of alternatives than sequential search, as it tackles one large optimization problem instead of multiple smaller ones. In particular, the number of decision variables increases linearly with the number of alternatives a. Also, for each feature and each pair of alternatives, we need to introduce an auxiliary variable if we want to obtain linear constraints (cf. Equation 6 and Table 1).

In contrast to the greedy procedure of sequential search, simultaneous search optimizes alternatives globally. Thus, the simultaneous procedure should yield the same or higher average feature-set quality for the same number of alternatives. Also, the quality can be more evenly distributed over the alternatives, as opposed to the dropping quality over the course of the sequential procedure. However, increasing the number of alternatives still has a negative effect on the average feature-set quality. Further, as opposed to the sequential procedure, there are no intermediate steps where users could interrupt the search.

3.3 Objective Functions – Finding Alternatives

In this section, we discuss how to find alternative feature sets. In particular, we describe how to solve the optimization problem from Section 3.1 for the different categories of feature-set quality measures from Section 2.2. We distinguish between white-box optimization (cf. Section 3.3.1), black-box optimization (cf. Section 3.3.2), and embedding alternatives (cf. Section 3.3.3).

3.3.1 White-Box Optimization

If the feature-set quality function $Q(s, X, y)$ is sufficiently simple, one can tackle alternative feature selection with a suitable white-box solver. We already showed that our notion of alternative feature sets results in 0-1 integer linear constraints (cf. Proposition 1). We now discuss several feature-selection methods with objectives that admit formulating a 0-1 integer linear problem. Appendix A.2 describes feature-selection methods we did not include in our experiments.

Univariate filter feature selection For univariate filter feature selection, the objective function is linear by default. In particular, these methods decompose the quality of a feature set into the qualities of the individual features:

$$Q_{\text{uni}}(s, X, y) = \sum_{j=1}^{n} q(X_{\cdot j}, y) \cdot s_j \tag{11}$$

Here, $q(\cdot)$ typically is a bivariate dependency measure, e.g., mutual information [55] or the absolute value of Pearson correlation, to quantify the relationship between one feature and the prediction target.

For this objective, Appendix A.3 specifies the complete optimization problem, including the constraints for alternatives. Appendix A.4 describes how to potentially speed up optimization by leveraging the monotonicity of the objective. Appendix A.6 proposes heuristic search methods, while we use exact optimization in our experiments.

Instead of an integer problem, one could formulate a weighted partial maximum satisfiability (MaxSAT) problem [5, 60], i.e., a weighted Max One problem [47]. In particular, Equation 11 is a sum of weighted binary variables, and the constraints for alternatives can be turned into SAT formulas with a cardinality encoding [99] for the sum expressions.

Post-hoc feature importance From the technical perspective, one can also insert values of post-hoc feature-importance scores into Equation 11. For example, one can pre-compute permutation importance [12] or SAGE scores [20] for each feature and use them as $q(X_{\cdot j}, y)$. However, such post-hoc importance scores often evaluate the usefulness of each feature in the presence of other features. Thus, the importance scores of different features are not independent of each other, violating the implicit assumption behind Equation 11. For example, a feature might show high post-hoc importance if another feature is present,

due to feature interaction, but low importance else. Equation 11 cannot express such conditional importance but requires one overall quality value for each feature. Re-calculating feature importance for each possible alternative feature set is infeasible. In practice, one can still use Equation 11 with importance scores only computed on the full dataset X, i.e., with all features being present. While such an approach might not represent importance in feature subsets faithfully, it can serve as a heuristic nevertheless.

FCBF The Fast Correlation-Based Filter (FCBF) [115] bases on the notion of predominance: Each selected feature's correlation with the prediction target must exceed a user-defined threshold as well as the correlation of each other selected feature with the given one. While the original FCBF uses a heuristic search to find predominant features, we propose a formulation as a constrained optimization problem to enable a white-box optimization for alternatives:

$$\begin{aligned} \max_{s} \quad & Q_{\mathrm{FCBF}}(s, X, y) = \sum_{j=1}^{n} q(X_{\cdot j}, y) \cdot s_j \\ \text{subject to:} \quad & \forall j_1, j_2 \in \{1, \ldots, n\},\ j_1 \neq j_2,\ (*) : s_{j_1} + s_{j_2} \leq 1 \\ \text{with } (*): \quad & q(X_{\cdot j_1}, y) \leq q(X_{\cdot j_2}, X_{\cdot j_1}) \end{aligned} \tag{12}$$

We drop the original FCBF's threshold parameter on feature-target correlation and maximize the latter instead, as in the univariate-filter case. This change could produce large feature sets that contain many low-quality features. As a countermeasure, one can constrain the feature-set sizes, as we do in our experiments. Additionally, one could also filter out the features with low target correlation before optimization. Further, we keep FCBF's constraints on feature-feature correlation. In particular, we prevent the simultaneous selection of two features if the correlation between them is at least as high as one of the features' correlation to the target. As the 'with'-condition in Equation 12 does not depend on the decision variables s, one can check whether it holds before optimization and add the corresponding linear constraint on s only if needed.

mRMR Minimal Redundancy Maximum Relevance (mRMR) [85] combines two criteria, i.e., feature relevance and feature redundancy. Relevance corresponds to the dependency between features and prediction target, which should be maximized, as for univariate filters. Redundancy corresponds to the dependency between features, which should be minimized. Using a bivariate dependency measure $q(\cdot)$, the objective is maximizing the following difference between relevance and redundancy:

$$Q_{\mathrm{mRMR}}(s, X, y) = \frac{\sum_{j=1}^{n} q(X_{\cdot j}, y) \cdot s_j}{\sum_{j=1}^{n} s_j} - \frac{\sum_{j_1=1}^{n} \sum_{j_2=1}^{n} q(X_{\cdot j_1}, X_{\cdot j_2}) \cdot s_{j_1} \cdot s_{j_2}}{(\sum_{j=1}^{n} s_j)^2} \tag{13}$$

If one knows the feature-set size $\sum_{j=1}^{n} s_j$ to be a constant k, the denominators of both fractions are constant, so the objective leads to a quadratic-programming

problem [80, 89]. If one additionally replaces each product terms $s_{j_1} \cdot s_{j_2}$ according to Equation 6, the problem becomes linear. However, there is a more efficient linearization [76, 78], which we use in our experiments:

$$\begin{aligned}
\max_{s} \quad & Q_{\text{mRMR}}(s, X, y) = \frac{\sum_{j=1}^{n} q(X_{\cdot j}, y) \cdot s_j}{k} - \frac{\sum_{j=1}^{n} z_j}{k \cdot (k-1)} \\
\text{subject to:} \quad \forall j_1 : \quad & A_{j_1} = \sum_{j_2 \neq j_1} q(X_{\cdot j_1}, X_{\cdot j_2}) \cdot s_{j_2} \\
\forall j : \quad & z_j \geq M \cdot (s_j - 1) + A_j \\
\forall j : \quad & z_j \in \mathbb{R}_{\geq 0} \\
\text{with indices:} \quad & j, j_1, j_2 \in \{1, \ldots, n\}
\end{aligned} \tag{14}$$

Here, A_{j_1} is the sum of all redundancy terms related to the feature with index j_1. Thus, one can use one real-valued auxiliary variable z_j for each feature instead of one new binary variable for each pair of features. Since redundancy should be minimized, z_j assumes the value of A_j with equality if the feature with index j is selected ($s_j = 1$) and is zero else ($s_j = 0$). To that end, M is a large positive value that deactivates the constraint on z_j if $s_j = 0$.

Since Equation 14 assumes the feature-set size $k \in \mathbb{N}$ to be user-defined before optimization, it requires fewer auxiliary variables and constraints than the more general formulation in [76, 78]. Further, following [80], we set the self-redundancy terms $q(X_{\cdot j}, X_{\cdot j})$, to zero and thereby exclude them from the objective. Thus, the redundancy term uses $k \cdot (k-1)$ instead of k^2 for averaging.

3.3.2 Black-Box Optimization

If feature-set quality has no closed-form expression, one has to treat it as a black-box function when searching for alternatives. This situation applies to wrapper feature-selection methods, which use prediction models to assess feature-set quality. One can optimize such black-box functions with search heuristics that systematically iterate over candidate feature sets. However, search heuristics often assume an unconstrained search space and may propose candidate feature sets that are not alternative enough. We see four ways to address this issue:

Enumerating feature sets Instead of using a search heuristic, one may enumerate all feature sets that are alternative enough. E.g., one can iterate over all feature sets and sort out those violating the constraints or use a solver to enumerate all valid alternatives directly. Both approaches are usually very inefficient, as there can be a vast number of alternatives.

Sampling feature sets Instead of considering all possible alternatives, one can also sample a limited number. E.g., one could sample from all feature sets but remove samples that are not alternative enough. However, if the number of valid alternatives is small, this approach might need many samples. One could also sample with the help of a solver. However, uniform sampling from

Algorithm 1: *Greedy Wrapper* for alternative feature selection.

Input: Dataset X with n features, Prediction target y,
Feature-set quality function $Q(\cdot)$,
Constraints for alternatives $Cons$,
Maximum number of iterations max_iters

Output: Set of feature-selection decision vectors $S = \{s^{(0)}, \ldots, s^{(a)}\}$

```
1  S ← Solve(Cons)                                   // Initial alternatives
2  iters ← 1                                  // Number of iterations = solver calls
3  if S = ∅ then return ∅                          // No valid alternatives exist
4  j1 ← 1                                    // Indices of features to be swapped
5  j2 ← j1 + 1
6  while iters < max_iters and j1 < n do
7      S' ← Solve(Cons ∪ {¬s_{j1}^{(i)}, ¬s_{j2}^{(i)} | i ∈ {0, ..., a}})      // Try swap
8      iters ← iters + 1
9      if S' ≠ ∅ and Q(S', X, y) > Q(S, X, y) then // Swap if improved
10         S ← S'
11         j1 ← 1                                  // Reset swap-feature indices
12         j2 ← j1 + 1
13     else if j2 < n then                  // Try next swap; advance one index
14         j2 ← j2 + 1
15     else                                // Try next swap; advance both indices
16         j1 ← j1 + 1
17         j2 ← j1 + 1
18 return S
```

a constrained space is a computationally hard problem, possibly harder than determining if a valid solution exists or not [28].

Multi-objective optimization If one phrases alternative feature selection as a multi-objective problem (cf. Section 3.1), there are no hard constraints anymore, and one could apply a standard multi-objective black-box search procedure. However, we chose to analyze a different problem formulation.

Adapting search One can adapt an existing search heuristic to consider the constraints for alternatives. One idea is to prevent the search from producing feature sets that violate the constraints or at least make the latter less likely, e.g., with a penalty in the objective function. Another idea is to 'repair' feature sets in the search that violate constraints, e.g., replacing them with the most similar feature sets satisfying the constraints. Such solver-assisted search approaches are common in search procedures for software feature models [34, 41, 109]. One could also apply solver-based repair to sampled feature sets.

Greedy wrapper For wrapper feature selection in our experiments, we propose a method that falls into the category *adapting search*. In particular, we adopt a greedy hill-climbing strategy [52] that observes constraints, as displayed in Algorithm 1. First, the algorithm uses a solver to find one solution that is alternative enough, given the current constraints (Line 1). Thus, it has a valid starting point and can always return a solution unless there are no valid solutions at all. Next, it tries 'swapping' two features, i.e., selecting the features if they were deselected or deselecting them if they were selected (Line 7). For simultaneous search, we swap the affected two features in each alternative feature set. This swap might violate cardinality constraints as well as constraints for alternatives. Thus, the algorithm calls the solver again to find a solution s' containing this swap and satisfying the other constraints. If such a solution s' exists and its quality $Q(s', X, y)$ improves the current solution, the algorithm continues from the new solution and tries again to swap the first and second feature (Lines 10–12). Else, it attempts to swap the next pair of features (Lines 13–17). In particular, we only evaluate one solution per swap before moving on rather than enumerating all valid solutions containing the swap.

The algorithm terminates if no swap leads to an improvement or a fixed number of iterations max_iters is reached (Line 6). Due to its heuristic nature, the algorithm might get stuck in local optima rather than yielding the global optimum. In particular, max_iters only is an upper bound on the iteration count since the algorithm can stop earlier. We define the iteration count as the number of calls to the solver, i.e., attempts to generate feature sets. This number also bounds the number of prediction models trained. However, we only train a model for valid solutions, and not all solver calls may yield one.

3.3.3 Embedding Alternatives

If feature selection is embedded into a prediction model, there is no general approach for finding alternative feature sets. Instead, one would need to embed the search for alternatives into model training as well. Thus, we leave the formulation of specific approaches open for future work. E.g., one could adapt the training of decision trees to not split on a feature if the resulting feature set of the tree was too similar to a given feature set. As another example, there are various formal encodings of prediction models, e.g., as SAT formulas [75, 94, 114], where 'training' already uses a solver. In such representations, one may directly add constraints for alternatives.

3.4 Computational Complexity

In this section, we analyze the time complexity of alternative feature selection. In particular, we study the scalability regarding the number of features $n \in \mathbb{N}$, feature-set size $k \in \mathbb{N}$ and number of alternatives $a \in \mathbb{N}_0$. Section 3.4.1 discusses exhaustive search for arbitrary feature-selection methods, while Section 3.4.2 examines univariate feature qualities. Section 3.4.3 summarizes key results.

3.4.1 Exhaustive Search for Arbitrary Feature-Selection Methods

An exhaustive search over the entire search space is the arguably simplest though inefficient approach to finding alternative feature sets. This approach provides an upper bound for the time complexity of a runtime-optimal search algorithm. In this section, we assume unit costs for elementary arithmetic operations like addition, multiplication, and comparison of two numbers.

Conventional feature selection In general, the search space of feature selection grows exponentially with n, even without alternatives. In particular, there are $2^n - 1$ possibilities to form a single non-empty feature set of arbitrary size. For a fixed feature-set size k, there are $\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} \leq n^k$ solution candidates. In an exhaustive search, we iterate over these feature sets:

Proposition 2 (Complexity of exhaustive conventional feature selection)**.** *Exhaustive search for one feature set of size k from n features has a time complexity of $O(n^k)$ without the cost of evaluating the objective function.*

Evaluating the objective means computing the quality of each solution candidate so that we can determine the best feature set in the end. The cost of this step depends on the feature-selection method but should usually be polynomial in n. Even better, since feature-set quality typically only depends on selected features rather than unselected ones, this cost may be polynomial in $k \ll n$.

If we assume $k \ll n,\ k \in O(1)$, i.e., k being a small constant, independent from n, then the complexity in Proposition 2 is polynomial rather than exponential in n. This assumption makes sense for feature selection, where one typically wants to obtain a small feature set from a high-dimensional dataset. However, the exponent k may still render an exhaustive search practically infeasible. In terms of parameterized complexity, the problem resides in class $\mathcal{XP}$ since the runtime term has the form $O(f(k) \cdot n^{g(k)})$ [25], here with parameter k and functions $f(k) = 1$, $g(k) = k$.

Sequential search Like conventional feature selection, sequential search for alternatives (cf. Definition 2) finds a single feature set at once. However, not all size-k feature sets are valid anymore. In particular, the constraints for alternatives put an extra cost on each solution candidate. Constraint checking involves iterating over all existing feature sets and features to compute the dissimilarity between sets (cf. Equation 19). This procedure entails a cost of $O(a \cdot n)$ for each new alternative and $O(a^2 \cdot n)$ for the whole sequential search with a alternatives. Combining this cost with Proposition 2, we obtain the following proposition:

Proposition 3 (Complexity of exhaustive sequential search)**.** *Exhaustive sequential search for $a \in \mathbb{N}$ alternative feature sets of size k from n features has a time complexity of $O(a^2 \cdot n^{k+1})$ without the cost of evaluating the objective function.*

Thus, the runtime resides in the parameterized complexity class $\mathcal{XP}$ with the parameter k and remains polynomial if $k \in O(1)$ and $a \in O(n^c),\ c \in O(1)$.

Simultaneous search Simultaneous search (cf. Definition 3) enlarges the search space since it optimizes $a + 1$ feature sets at once. Thus, an exhaustive search over size-k feature sets iterates over $O(n^{k\cdot(a+1)})$ solution candidates. Including the cost of constraint checking, we arrive at the following proposition:

Proposition 4 (Complexity of exhaustive simultaneous search)**.** *Exhaustive simultaneous search for $a \in \mathbb{N}$ alternative feature sets of size k from n features has a time complexity of $O(a^2 \cdot n^{k\cdot(a+1)+1})$ without the cost of evaluating the objective function.*

The scalability with n is worse than for sequential search since the number of alternatives appears in the exponent now, except for a special case discussed in Appendix A.5.1. Proposition 4 also assumes that the constraints do not use linearization variables (cf. Equations 6 and 20), which would enlarge the search space even further. Finally, the complexity remains polynomial in n if a and k are small and independent from n, i.e., $a \cdot k \in O(1)$:

Proposition 5 (Parameterized complexity of simultaneous search)**.** *Simultaneous search for $a \in \mathbb{N}$ alternative feature sets of size k from n features resides in the parameterized complexity class $\mathcal{XP}$ for the parameter $a \cdot k$.*

3.4.2 Univariate Feature Qualities

Motivation While the assumption $a \cdot k \in O(1)$ ensures polynomial runtime for arbitrary feature-selection methods, the optimization problem can still be hard without this assumption. In the following, we derive complexity results for univariate feature qualities (cf. Equation 11 and Appendix A.3). This feature-selection method has the arguably simplest objective function, i.e., a feature set's quality equals the sum of its constituent features' qualities. This simplicity eases the transformation from and to well-known $\mathcal{NP}$-hard problems. Appendix A.5.2 discusses related work on these problems in detail.

Min-aggregation with complete partitioning We start with three assumptions, which we will drop later: First, we use a dissimilarity threshold of $\tau = 1$, i.e., zero overlap of feature sets. Second, all features must be part of one set. Third, we analyze simultaneous search with min-aggregation (cf. Equation 16). We call the combination of the first two assumptions, which implies $n = (a + 1) \cdot k$, a *complete partitioning*. This scenario differs from the one for which we made polynomial-runtime claims in Section 3.4.1.

A key factor for the hardness of partitioning is the number of solutions: There are $\left\{ {n \atop a} \right\}$ ways to partition a set of n elements into a non-empty subsets, a Stirling number of the second kind [31], which roughly scale like $a^n/a!$ [70], i.e., exponential in n for a fixed a. Even if the subset sizes are fixed, the scalability regarding n remains bad since it bases on a multinomial coefficient.

Our complete-partitioning scenario is a variant of the Multi-Way Number Partitioning problem: Partition a multiset of n integers into a subsets such

that the sums of all subsets are as equal as possible [54]. One problem formulation, called MULTIPROCESSOR SCHEDULING in [30], minimizes the maximum subset sum: The goal is to assign tasks with different lengths to a fixed number of processors such that the maximum processor runtime is minimal. Multiplying task lengths with -1, one can turn the minimax problem of MULTIPROCESSOR SCHEDULING into the maximin formulation of simultaneous search with min-aggregation: The tasks become features, the negative task lengths become univariate feature qualities, and the processors become feature sets. Since MULTIPROCESSOR SCHEDULING is $\mathcal{NP}$-complete, even for just two partitions [30], our problem is $\mathcal{NP}$-complete as well:

Proposition 6 (Complexity of simultaneous search with min-aggregation, complete partitioning, and unconstrained feature-set size)**.** *Assuming univariate feature qualities, a dissimilarity threshold $\tau = 1$, unconstrained feature-set sizes, and all n features have to be selected, simultaneous search for alternative feature sets with min-aggregation is $\mathcal{NP}$-complete.*

Since the assumptions in Proposition 6 denote a special case of alternative feature selection, we directly obtain the following, more general proposition:

Proposition 7 (Complexity of simultaneous search with min-aggregation)**.** *Simultaneous search for alternative feature sets with min-aggregation is $\mathcal{NP}$-hard.*

While Proposition 6 allowed arbitrary sets sizes, there are also existing partitioning problems for constrained k, e.g., called BALANCED NUMBER PARTITIONING or K-PARTITIONING. K-PARTITIONING with a minimax objective is $\mathcal{NP}$-hard [4] and can be transformed into our maximin objective as above:

Proposition 8 (Complexity of simultaneous search with min-aggregation, complete partitioning, and constrained feature-set size)**.** *Assuming univariate feature qualities, a dissimilarity threshold $\tau = 1$, desired feature-set size k, and all n features have to be selected, simultaneous search for alternative feature sets with min-aggregation is $\mathcal{NP}$-complete.*

Min-aggregation with incomplete partitioning We now allow that some features may not be part of any feature set while we keep the assumption of zero feature-set overlap. The problem of finding such an *incomplete partitioning* still is $\mathcal{NP}$-complete in general (cf. Appendix A.5.3 for the proof):

Proposition 9 (Complexity of simultaneous search with min-aggregation, incomplete partitioning, and constrained feature-set size)**.** *Assuming univariate feature qualities, a dissimilarity threshold $\tau = 1$, desired feature-set size k, and* not *all n features have to be selected, simultaneous search for alternative feature sets with min-aggregation is $\mathcal{NP}$-complete.*

Min-aggregation with overlapping feature sets The problem with $\tau < 1$, i.e., set overlap, also is $\mathcal{NP}$-hard in general (cf. Appendix A.5.3 for the proof):

Proposition 10 (Complexity of simultaneous search with min-aggregation, $\tau < 1$, and constrained feature-set size)**.** *Assuming univariate feature qualities, a dissimilarity threshold $\tau < 1$, and desired feature-set size k, simultaneous search for alternative feature sets with min-aggregation is $\mathcal{NP}$-hard.*

Sum-aggregation In contrast to the previous $\mathcal{NP}$-hardness results for min-aggregation, sum-aggregation (cf. Equation 15) with $\tau = 1$ admits polynomial-time algorithms (cf. Appendix A.5.3 for the proof):

Proposition 11 (Complexity of search with sum-aggregation and $\tau = 1$)**.** *Assuming univariate feature qualities and a dissimilarity threshold $\tau = 1$, the search for alternative feature sets with sum-aggregation has a time complexity of $O(n)$ for a complete partitioning of n features and $O(n \cdot \log n)$ for an incomplete partitioning.*

This feasibility result applies to sequential and simultaneous search, an arbitrary number of alternatives a, and arbitrary feature-set sizes. The key reason for polynomial runtime is that sum-aggregation does not require balancing the feature sets' qualities. Thus, $\tau = 1$ allows many solutions with the same objective value. While at least one of these solutions also optimizes the objective with min-aggregation, most do not. Hence, it is not a contradiction that optimizing with min-aggregation is considerably harder.

3.4.3 Summary

We showed that simultaneous search for alternative feature sets is $\mathcal{NP}$-hard in general (cf. Proposition 7). We also placed it in the parameterized complexity class $\mathcal{XP}$ (cf. Proposition 5), having a and k as the parameters that drive the hardness of the problem. For univariate feature qualities and min-aggregation, we obtained more specific $\mathcal{NP}$-hardness results for (1) complete partitioning, i.e., $\tau = 1$ and $(a + 1) \cdot k = n$ (cf. Proposition 8), (2) incomplete partitioning, i.e., $(a + 1) \cdot k < n$ (cf. Proposition 9) and (3) feature set overlap, i.e., $\tau < 1$ (cf. Proposition 10). In contrast, we also inferred polynomial runtime for univariate feature qualities, sum-aggregation, and $\tau = 1$ (cf. Proposition 11).

4 Related Work

In this section, we review related work from the fields of feature selection (cf. Section 4.1), subgroup discovery (cf. Section 4.2), clustering (cf. Section 4.3), subspace clustering and subspace search (cf. Section 4.4), and explainable artificial intelligence (cf. Section 4.5). To the best of our knowledge, searching for optimal alternative feature sets in the sense of this paper is novel. However, there is literature on optimal alternatives outside the field of feature selection. Also, there are works on finding multiple, diverse feature sets.

4.1 Feature Selection

Conventional feature selection Most feature-selection methods only yield one solution [11], though some exceptions exist. Nevertheless, none of the following approaches searches for optimal alternatives in our sense.

[97] proposes a genetic algorithm that iteratively updates a population of multiple feature sets. To foster diversity, the algorithm's fitness criterion does not only consider feature-set quality but also a penalty on feature-set overlap in the population. However, users cannot control the admissible overlap, i.e., there is no parameter comparable to τ. In contrast, the genetic algorithm's parameter for the population size corresponds to the number of alternatives.

[27] employs multi-objective genetic algorithms to obtain prediction models with different complexity and diverse feature sets. However, the two objectives are prediction performance and feature-set size, while diversity only influences the genetic selection step under particular circumstances.

[73] clusters features and forms alternatives by picking one feature from each cluster. However, they do this to reduce the number of features for subsequent model selection and model evaluation, not as a guided search for alternatives.

Ensemble feature selection Ensemble feature selection [92, 96] combines feature-selection results, e.g., obtained by different feature-selection methods or on different samples of the data. Fostering diverse feature sets might be a subgoal to improve prediction performance, but it is usually only an intermediate step. This focus differs from our goal of finding optimal alternatives.

[113] obtains feature sets or rankings on bootstrap samples of the data. Next, an aggregation strategy creates one or multiple diverse feature sets. The authors propose using k-medoid clustering and frequent itemset mining for the latter. While these approaches allow to control the number of feature sets, there is no parameter for their dissimilarity. Also, aggregation builds on bootstrap sampling instead of being allowed to form arbitrary alternatives.

[62] builds an ensemble prediction model from classifiers trained on different feature sets. To this end, a genetic algorithm iteratively evolves a population of feature sets. Diversity is one of multiple fitness criteria, with the Hamming distances quantifying the dissimilarity of feature sets. However, since feature diversity is only one of several objectives, users cannot control it directly.

[35] computes feature relevance separately for each class and then combines the top features. This procedure can yield alternatives but does not enforce dissimilarity. Also, the number of alternatives is fixed to the number of classes.

Statistically equivalent feature sets Approaches for statistically equivalent feature sets [11, 56] use statistical tests to determine features or feature sets that are equivalent for predictions. E.g., a feature may be independent of the target given another feature. A search algorithm conducts multiple such tests and outputs equivalent feature sets or a corresponding feature grouping.

Our notion of alternatives differs from equivalent feature sets in several aspects. In particular, building optimal alternatives from equivalent feature sets

is not straightforward. Depending on how the statistical tests are configured, there can be an arbitrary number of equivalent feature sets without explicit quality-based ordering. Instead, we always provide a fixed number of alternatives. Also, our alternatives need not have equivalent quality but should be optimal under constraints. Further, our dissimilarity threshold allows controlling overlap between feature sets instead of eliminating all redundancies.

Constrained feature selection We define alternatives via constraints on feature sets. There already is work on other kinds of constraints in feature selection, e.g., for feature cost [83], feature groups [116], or domain knowledge [6, 32]. These approaches are orthogonal to our work, as such constraints do not explicitly foster optimal alternatives. At most, they might implicitly lead to alternative solutions [6]. Further, most of the approaches are tied to particular constraint types, while our integer-programming formulation also supports such constraints besides the ones for alternatives. [6] is an exception in that regard since it models feature selection as a Satisfiability Modulo Theories (SMT) optimization problem, which admits our constraints for alternatives as well.

4.2 Subgroup Discovery

[59] presents six strategies to foster diversity in subgroup set discovery, which searches for interesting regions in the data space, i.e., combinations of conditions on feature values, rather than only selecting features. Three strategies yield a fixed number of alternatives, and the other three a variable number. The strategies become part of beam search, i.e., a heuristic search procedure, while we mainly consider exact optimization. Also, the criteria for alternatives differ from ours. The strategy *fixed-size description-based selection* prunes subgroups with the same quality as previously found ones if they differ by at most one feature-value condition. In contrast, we require dissimilarity independent from the quality, have a flexible dissimilarity threshold, and support simultaneous besides sequential search for alternatives. Another strategy, *variable-size description-based selection*, limits the total number of subgroups a feature may occur in but does not constrain subgroup overlap per se. The four remaining strategies in [59] have no obvious counterpart in our feature-selection scenario.

4.3 Clustering

Finding alternative solutions has been addressed extensively in the field of clustering. [9] gives a taxonomy and describes algorithms for alternative clustering. Our problem definition in Sections 3.1 and 3.2 is, on a high level, inspired by the one in [9]: Find multiple solutions that maximize quality while minimizing similarity. [9] also distinguishes between singular/multiple alternatives and sequential/simultaneous search. They mention constraint-based search for alternatives as one of several solution paradigms. Further, feature selection can help to find alternative clusterings [101]. Nevertheless, the problem definition for alternatives in clustering and feature selection is fundamentally different.

First, the notion of dissimilarity differs, as we want to find differently composed feature sets while alternative clustering targets at different assignments of data objects to clusters. Second, our objective function, i.e., feature-set quality, relates to a supervised prediction scenario while clustering is unsupervised.

Two exemplary approaches for alternative clustering are *COALA* [7] and *MAXIMUS* [8]. COALA [7] imposes *cannot-link constraints* on pairs of data objects rather than constraining features: Data objects from the same cluster in the original clustering should be assigned to different clusters in the alternative clustering. In each step of its iterative clustering procedure, COALA compares the quality of an action observing the constraints to another one violating them. Based on a threshold on the quality ratio, either action is taken. MAXIMUS [8] employs an integer program to formulate dissimilarity between clusterings. In particular, it wants to maximize the dissimilarity of the feature-value distributions in clusters between the clusterings. The output of the integer program leads to constraints for a subsequent clustering procedure.

4.4 Subspace Clustering and Subspace Search

Finding multiple useful feature sets plays a role in subspace clustering [42, 72] and subspace search [29, 79, 102]. These approaches strive to improve the results of data-mining algorithms by using subspaces, i.e., feature sets, rather than the full space, i.e., all features. While some subspace approaches only consider individual subspaces, others explicitly try to remove redundancy between subspaces [72, 79] or foster subspace diversity [29, 102]. In particular, [42] surveys subspace-clustering approaches yielding multiple results and discusses the redundancy aspect. However, subspace clustering and -search approaches differ from alternative feature selection in at least one of the following aspects:

First, the objective differs, i.e., definitions of subspace quality deviate from feature-set quality in our scenario. Second, definitions of subspace redundancy may consider dissimilarity between projections of the entire data, i.e., data objects with feature values, into subspaces, while our notion of dissimilarity purely bases on binary feature-selection decisions. Third, controlling dissimilarity in subspace approaches is often less user-friendly than with our parameter τ. E.g., dissimilarity might be a regularization term in the objective rather than a hard constraint, or there might not be an explicit control parameter at all.

4.5 Explainable Artificial Intelligence (XAI)

In the field of XAI, alternative explanations might provide additional insights into predictions, enable users to develop and test different hypotheses, appeal to different kinds of users, and foster trust in the predictions [50, 108]. In contrast, obtaining significantly different explanations for the same prediction might raise doubts about how meaningful the explanations are [43]. Finding diverse explanations had been studied for various explainers, e.g., for counterfactuals [21, 44, 67, 71, 91, 105], criticisms [49], and semifactual explanations [2]. There are

several approaches to foster diversity, e.g., ensembling different kinds of explanations [98], considering multiple local minima [105], using a search algorithm that maintains diversity [21], extending the optimization objective [2, 49, 71], or introducing constraints [44, 67, 91]. The last option is similar to the way we enforce alternatives. Of the various mentioned approaches, only [2, 67, 71] introduce a parameter to control the diversity of solutions. Of these three works, only [67] offers a user-friendly dissimilarity threshold in $[0, 1]$, while the other two approaches employ a regularization parameter in the objective.

Despite similarities, all the previously mentioned XAI techniques tackle different problems than alternative feature selection. In particular, they provide local explanations, i.e., target at prediction outcomes for individual data objects and build on feature values. In contrast, we are interested in the global prediction quality of feature sets. For example, counterfactual explanations [33, 100, 104] alter feature *values as little as possible* to produce an alternative prediction *outcome*. In contrast, alternative feature sets might alter the feature *selection significantly* while trying to maintain the original prediction *quality*.

5 Experimental Design

In this section, we describe our experimental design. We give a brief overview of its goal and components (cf. Section 5.1) before elaborating on the components in detail. In particular, we describe evaluation metrics (cf. Section 5.2), methods (cf. Section 5.3), datasets (cf. Section 5.4), and implementation (cf. Section 5.5).

5.1 Overview

We conduct experiments with 30 binary-classification datasets. Our evaluation focuses on the trade-off between feature-set quality and obtaining alternative feature sets. We compare five feature-selection methods, representing different notions of feature-set quality. Also, we train prediction models with the resulting feature sets and analyze prediction performance. To find alternatives, we consider simultaneous as well as sequential search. We systematically vary the number of alternatives and the dissimilarity threshold for alternatives.

5.2 Evaluation Metrics

Feature-set quality We evaluate feature-set quality with two metrics. First, we report the *objective value* $Q(s, X, y)$ of the feature-selection methods, which guided the search for alternatives. Second, we train prediction models with the found feature sets. We report *prediction performance* in terms of the Matthews correlation coefficient (MCC) [64]. This coefficient is insensitive to class imbalance, reaches its maximum of 1 for perfect predictions, and is 0 for random guessing. We conduct stratified five-fold cross-validation to analyze how well feature selection and prediction models generalize. The search for alternatives and model training are limited to the training data.

Runtime We consider two metrics related to runtime.

First, we analyze the *optimization time*. For white-box feature-selection methods, we measure the total runtime of solver calls. We exclude the time for computing feature qualities and feature dependencies for the objective since one can compute these values once per dataset and then re-use them in each solver call. For *Greedy Wrapper*, we measure the runtime of the entire black-box optimization procedure involving multiple solver calls and model trainings.

Second, we examine the *optimization status*, which can take four values. If the solver finished before reaching a timeout, it either found an *optimal* solution or proved the problem *infeasible*, i.e., no solution exists. If the solver reached its timeout, it either found a *feasible* solution whose optimality it could not prove or found no valid solution though one might exist, so the problem is *not solved*.

5.3 Methods

We compare several approaches for making predictions (cf. Section 5.3.1), feature selection (cf. Section 5.3.2), and searching alternatives (cf. Section 5.3.3).

5.3.1 Prediction

As prediction models, we use decision trees [13] and random forests with 100 trees [12]. Both these models admit learning complex, non-linear dependencies from the data. We leave the hyperparameters of the models at their defaults, except for using information gain instead of Gini impurity as the split criterion, to be consistent with our parametrization of filter feature-selection methods.

Note that tree models also carry out feature selection themselves, i.e., they are embedded approaches. Thus, they might not use all features from the alternative feature sets. However, this is not a problem for our study. We are interested in which performance the models achieve if they are limited to certain feature sets, not if and how they use each feature from these sets.

5.3.2 Feature Selection (Objective Functions)

We search for alternatives under different notions of feature-set quality as the objective function. We choose five well-known feature-selection methods that are easy to parameterize and cover the different categories from Section 2.2 except *embedded*, as explained in Section 3.3.3. One method (*Greedy Wrapper*) requires black-box optimization, while the other four are white-box.

With each feature-selection method, we select $k \in \{5, 10\}$ features, thereby obtaining small feature sets. We enforce the desired k with a simple constraint in optimization, using the feature-set-size expression from Equation 5.

Filter feature selection We evaluate three filter methods, all using mutual information [55] as the dependency measure $q(\cdot)$. This measure allows to capture arbitrary dependencies rather than, e.g., just linear correlations. *MI* denotes a univariate filter (cf. Equation 11), while *FCBF* (cf. Equation 12) and *mRMR*

(cf. Equation 14) are multivariate. Since mutual information has no fixed upper bound, we normalize per dataset and cross-validation fold to improve the comparability of feature-set quality. For *FCBF* and *MI*, we normalize the individual features' qualities such that selecting all features yields a quality of 1 and selecting no feature yields a quality of 0. For *mRMR*, we min-max-normalize all mutual-information values to $[0, 1]$, so the overall objective is in $[-1, 1]$.

Wrapper feature selection As a wrapper method, we employ the hill-climbing strategy *Greedy Wrapper* from Algorithm 1. We set max_iters to 1000. To evaluate feature-set quality within the wrapper, we apply a stratified 80:20 holdout split and train decision trees. $Q(s, X, y)$ corresponds to the prediction performance in terms of MCC on the 20% validation part.

Post-hoc feature importance As a post-hoc importance measure, we use model-based feature importance provided by *scikit-learn*. Again, we use a decision tree as the model. There, importance expresses a feature's contribution towards optimizing the split criterion of the tree, for which we choose information gain. These importances are normalized to sum up to 1 by default. We plug the importances into Equation 11, i.e., treat them like univariate filter scores. The interpretation is different, though, since the scores originate from trees trained with all features rather than assessing features in isolation.

5.3.3 Alternatives (Constraints)

Competitors We only evaluate approaches for searching alternatives that we proposed in this article. As discussed in Section 4, approaches from related work pursue different objective functions, operate with different notions of alternatives, and may only work for particular feature-selection methods. All these points prevent a meaningful comparison of these approaches to ours. E.g., a feature set considered alternative in related work might violate our constraints for alternatives. Further, within our own approaches, we can still put the feature-set quality into perspective by comparing alternatives to each other.

Search parametrization We employ *sequential* (cf. Equation 9) and *simultaneous* (cf. Equation 10) search for alternatives. For the latter, we use sum-aggregation (cf. Equation 15) and min-aggregation (cf. Equation 16) in the objective. We evaluate $a \in \{1, \dots, 10\}$ alternatives for sequential search and $a \in \{1, \dots, 5\}$ for simultaneous search due to the higher runtime of the latter. For the dissimilarity threshold τ, we analyze all possible sizes of the feature-set overlap in the Dice dissimilarity (cf. Equations 3 and 8). Thus, for $k = 5$, we consider $\tau \in \{0.2, 0.4, 0.6, 0.8, 1\}$, corresponding to an overlap of four to zero features. For $k = 10$ we consider $\tau \in \{0.1, 0.2, \dots, 1\}$. We exclude $\tau = 0$, which would allow returning duplicate feature sets.

Optimization All searches for alternatives rely on solvers. With *Greedy Wrapper* as the feature-selection method, the search procedure is heuristic and might not cover the entire search space. There, the solver only assists in finding valid solutions but does not optimize. For the white-box feature-selection methods, the solver exactly solves the underlying optimization problems. Thus, given sufficient solving time, these alternatives are globally optimal.

Timeout We employ a solver timeout to make a large-scale evaluation feasible and to account for the high variance of solver runtime, even for optimization problems of the same size. In particular, we grant each solver call 60 s multiplied by the number of feature sets. Thus, sequential search conducts multiple solver calls with 60 s timeout, while simultaneous search conducts one solver call with proportionally more time. The summed timeout for a fixed number of alternatives is the same for both search methods. For 84% of the feature sets in our evaluation, the solver finished before the timeout.

5.4 Datasets

We evaluate alternative feature selection on the Penn Machine Learning Benchmarks (PMLB) [82, 90]. To harmonize evaluation, we only consider binary-classification datasets, though alternative feature selection also works for regression and multi-class problems. We exclude datasets with less than 100 data objects since they might entail a high uncertainty when assessing feature-set quality. Otherwise, the number of data objects should not systematically impact the feature-set quality and is unimportant for our evaluation. Also, we exclude datasets with less than 15 features to leave some room for alternatives. Next, we exclude one dataset with 1000 features, which would dominate the overall runtime of the experiments. Finally, we manually exclude datasets that seem duplicated or modified versions of other datasets from the benchmark.

Consequently, we obtain 30 datasets with 106 to 9822 data objects and 15 to 168 features. The datasets contain no missing values. Categorical features have an ordinal encoding by default. Table 2 lists these datasets.

5.5 Implementation and Execution

We implemented our experimental pipeline in Python 3.8, using *scikit-learn* [84] for machine learning and the solver *SCIP* [10] via the package *OR-Tools* [86] for optimization. A requirements file in our code specifies the versions of all packages. The experimental pipeline parallelizes over datasets, cross-validation folds, and feature-selection methods, while solver calls and model training are single-threaded. We ran the pipeline on a server with 128 GB RAM and an *AMD EPYC 7551* CPU, having 32 physical cores and a base clock of 2.0 GHz. The parallelized pipeline run took 255 hours, i.e., about 10.6 days.

Dataset	m	n
backache	180	32
chess	3196	36
churn	5000	20
clean1	476	168
clean2	6598	168
coil2000	9822	85
credit_a	690	15
credit_g	1000	20
dis	3772	29
G_Epistasis_2_Way_20atts_0.1H_EDM_1_1	1600	20
G_Epistasis_2_Way_20atts_0.4H_EDM_1_1	1600	20
G_Epistasis_3_Way_20atts_0.2H_EDM_1_1	1600	20
G_Heterogeneity_20atts_1600_Het_0.4_0.2_50_EDM_2_001	1600	20
G_Heterogeneity_20atts_1600_Het_0.4_0.2_75_EDM_2_001	1600	20
hepatitis	155	19
Hill_Valley_with_noise	1212	100
horse_colic	368	22
house_votes_84	435	16
hypothyroid	3163	25
ionosphere	351	34
molecular_biology_promoters	106	57
mushroom	8124	22
ring	7400	20
sonar	208	60
spambase	4601	57
spect	267	22
spectf	349	44
tokyo1	959	44
twonorm	7400	20
wdbc	569	30

Table 2: Datasets from PMLB used in our experiments. m denotes the number of instances and n the number of features. Dataset names starting with 'G_' actually start with 'GAMETES_'; we truncated them to reduce the table's width.

6 Evaluation

In this section, we evaluate our experiments. In particular, we discuss the parametrization for searching alternatives: the search method (cf. Section 6.1), number of alternatives a (cf. Section 6.2), and dissimilarity threshold τ (cf. Section 6.3). Section 6.4 summarizes key findings. Additionally, Appendix A.7 contains results for further dimensions of our experimental design.

6.1 Search Methods for Alternatives

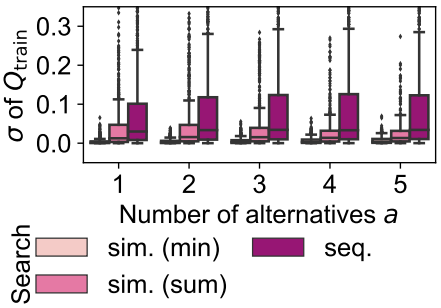
Variance in feature-set quality As expected, the search method influences how much the training-set objective value Q varies between alternatives found within each search run. Figure 1a visualizes this result for *MI* as the feature-selection method and $k = 5$. In particular, the quality of multiple alternatives found by sequential search usually varies more than for simultaneous search. For simultaneous search, min-aggregation yields considerably more homogeneous feature-set quality than sum-aggregation. These findings apply to all white-box feature-selection methods but not the heuristic *Greedy Wrapper*.

As Figures 1c and 1e show, the variance of feature-set quality differs considerably less between the search methods on the test set, for the objective value as well as prediction performance. In particular, alternatives found by simultaneous search do not have considerably more homogeneous test feature-set quality than for sequential search. This effect might result from overfitting: Even if training feature-set quality is similar, some alternatives might generalize better, i.e., lose less quality on the test set than others. Thus, the variance in test feature-set quality caused by overfitting could alleviate the effect on variance caused by the search method.

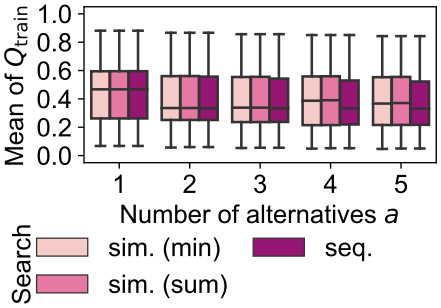
Average value of feature-set quality While obtaining alternatives of homogeneous quality can be one goal of simultaneous search, the main selling point compared to sequential search would be alternatives of higher average quality. However, we found that simultaneous search is not clearly better than sequential search in that regard. In particular, Figure 1b compares the distribution of the mean training-set objective in search runs with *MI* as feature-selection method and $k = 5$. We observe that all search methods yield very similar distributions of feature-set quality. The other four feature-selection methods also do not show a general quality advantage of simultaneous search. At most, simultaneous search tends to develop a slight advantage with a growing number of alternatives for *MI*, as visible in Figure 1b, and *Model Gain*.

The test-set objective value in Figure 1d and the test-set prediction performance in Figure 1f also exhibit the negligible quality difference between the search methods. As Figure 2a displays, the variation in prediction performance caused by other dimensions of the experimental design, e.g., dataset, dissimilarity threshold τ, etc., exceeds the variation due to the search methods.

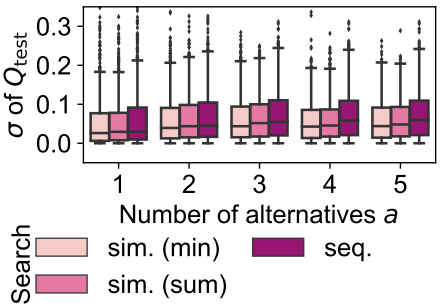
Finally, Figure 2b displays the difference in feature-set quality between sequential and simultaneous search compared on each search setting separately,



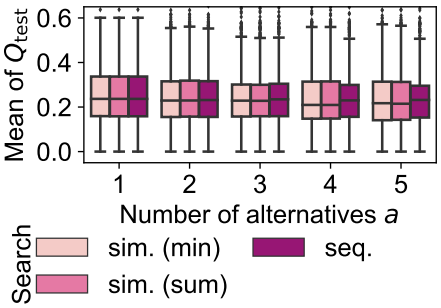
(a) Standard deviation of training-set objective value within search runs.



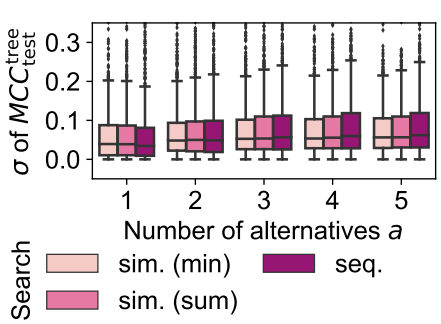
(b) Mean of training-set objective value within search runs.



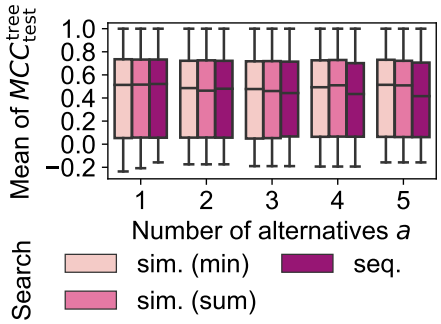
(c) Standard deviation of test-set objective value within search runs.



(d) Mean of test-set objective value within search runs.

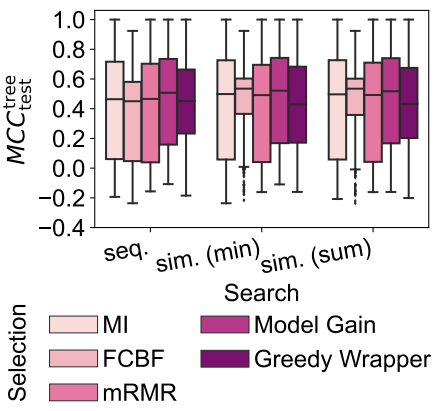


(e) Standard deviation of test-set prediction performance within search runs.

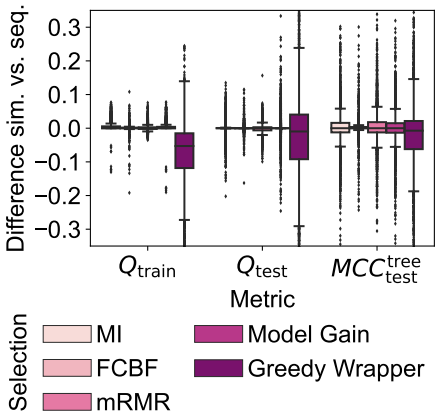


(f) Mean of test-set prediction performance within search runs.

Figure 1: Feature-set quality over the number of alternatives a, by search method for alternatives and evaluation metric. Results with *MI* as feature-selection method and $k = 5$. Y-axes are truncated to improve readability.



(a) Test-set prediction performance.



(b) Difference in quality between simultaneous (sum-aggregation) and sequential search by evaluation metric. Y-axis is truncated to improve readability.

Figure 2: Feature-set quality by feature-selection method and search method for alternatives. Results with $k = 5$ and $a \in \{1, 2, 3, 4, 5\}$.

i.e., each combination of dataset, dissimilarity threshold τ, etc. The figure again shows little variation in quality between the search methods except for *Greedy Wrapper* feature selection. In particular, the quality difference is usually close to zero, apart from a few outliers. Additionally, the figure highlights that outliers can occur in both directions: While simultaneous search can yield better feature sets in some scenarios, sequential search can be better in others.

Optimization status One reason why simultaneous search fails to consistently beat sequential search quality-wise is that search results can be suboptimal. For *Greedy Wrapper*, the search is heuristic per se and does not cover the entire search space. For all feature-selection methods, the solver can time out. Table 3 shows that simultaneous search has a higher likelihood of timeouts than sequential search, likely due to the larger size of the optimization problem (cf. Table 1). In particular, for up to five alternatives and $k = 5$, all sequential searches for *FCBF*, *MI*, and *Model Gain* finished within the timeout, i.e., yielded the optimal feature set or ascertained infeasibility, while *mRMR* had about 9% timeouts. In contrast, for simultaneous search with sum-aggregation, all feature-selection methods experience timeouts: Roughly 2-3% of the searches for *FCBF*, *MI*, and *Model Gain*, and 67% of the searches for *mRMR* found a feasible solution but could not prove optimality. Such timeout-affected simultaneous solutions can be worse than optimal sequential solutions. The optimization status *not solved*, i.e., not finding a feasible solution without proving infeasibility, did not occur in the displayed results. Min-aggregation instead of sum-aggregation in simultaneous search exhibits more timeouts for *MI* and

Selection	Search	Optimization status		
		Infeasible	Feasible	Optimal
FCBF	seq.	66.39%	0.00%	33.61%
FCBF	sim. (min)	73.07%	1.73%	25.20%
FCBF	sim. (sum)	73.07%	2.19%	24.75%
MI	seq.	1.97%	0.00%	98.03%
MI	sim. (min)	4.67%	9.60%	85.73%
MI	sim. (sum)	4.67%	3.17%	92.16%
Model Gain	seq.	1.97%	0.00%	98.03%
Model Gain	sim. (min)	4.67%	5.55%	89.79%
Model Gain	sim. (sum)	4.67%	1.92%	93.41%
mRMR	seq.	1.95%	8.67%	89.38%
mRMR	sim. (min)	4.67%	49.04%	46.29%
mRMR	sim. (sum)	4.67%	67.39%	27.95%

Table 3: Frequency of optimization statuses (cf. Section 5.2) by feature-selection method and search method for alternatives. Results with $k = 5$, $a \in \{1, 2, 3, 4, 5\}$, and excluding *Greedy Wrapper*, which uses the solver for satisfiability checking rather than optimizing. Each row adds up to 100%.

Model Gain but less for *FCBF* and *mRMR*. Still, sequential search incurs fewer timeouts for all these four feature-selection methods.

Finally, note that the fraction of timeouts strongly depends on the number of alternatives a, as Table 4 displays: For simultaneous search with $k = 5$ and sum-aggregation, roughly 8% of the white-box searches timed out for $a = 1$ but 20% for $a = 3$ and 30% for $a = 5$. While we grant simultaneous searches proportionally more time for multiple alternatives, the observed increase in timeouts suggests that runtime increases super-proportionally, as we analyze next.

Optimization time The actual optimization times also speak in favor of sequential search. As Table 5 shows, the mean optimization time of sequential search is lower for all five feature-selection methods. In particular, the difference between sequential and simultaneous search is up to three orders of magnitude for the four white-box feature-selection methods. Further, *FCBF*, *MI*, and *Model Gain* experience a dramatic increase in optimization time with the number of alternatives a in simultaneous search, as Table 6 displays. In contrast, the runtime increase is considerably less for sequential search, which shows an approximately linear trend with the number of alternatives.

Based on all results described in this section, we focus on sequential search in the following. In particular, it was significantly faster than simultaneous search while yielding similar feature-set quality.

Another interesting question for practitioners is how the runtime relates to n, the number of features in the dataset. One would expect a positive correlation since the optimization problem's instance size increases with n. Roughly speak-

a	Optimization status		
	Infeasible	Feasible	Optimal
1	16.10%	7.57%	76.33%
2	17.50%	13.43%	69.07%
3	20.00%	20.40%	59.60%
4	27.00%	21.47%	51.53%
5	28.23%	30.47%	41.30%

Table 4: Frequency of optimization statuses (cf. Section 5.2) by number of alternatives a. Results from simultaneous search with sum-aggregation, $k = 5$, and excluding *Greedy Wrapper*. Each row adds up to 100%.

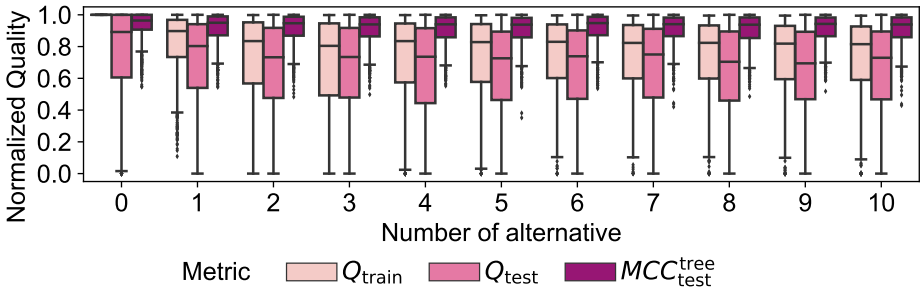
Selection	Optimization time		
	Seq.	Sim. (min)	Sim. (sum)
FCBF	0.22 s	11.91 s	13.09 s
Greedy Wrapper	54.23 s	61.10 s	63.45 s
MI	0.03 s	48.25 s	25.39 s
Model Gain	0.03 s	30.91 s	19.98 s
mRMR	34.12 s	157.87 s	189.76 s

Table 5: Mean optimization time by feature-selection method and search method for alternatives. Results with $k = 5$ and $a \in \{1, 2, 3, 4, 5\}$.

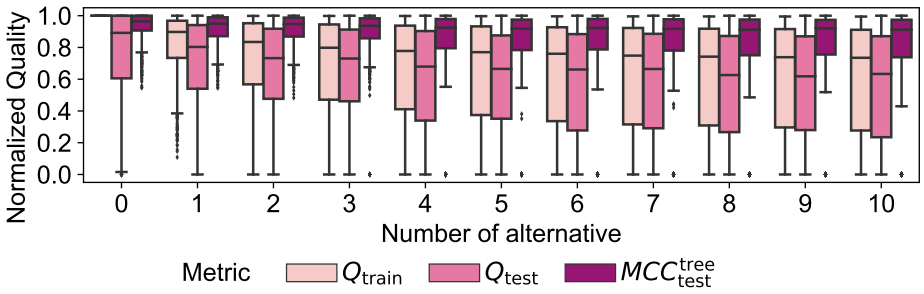
ing, this trend appears in our experimental data indeed. However, the observed trend is rather noisy, particularly for simultaneous search, and some higher-dimensional datasets even show lower average runtimes than lower-dimensional datasets. This result indicates that several other factors than n influence runtime. Besides factors related to the datasets and experimental design, the heuristics used by the solver may also cause the runtime to fluctuate considerably.

6.2 Number of Alternatives a

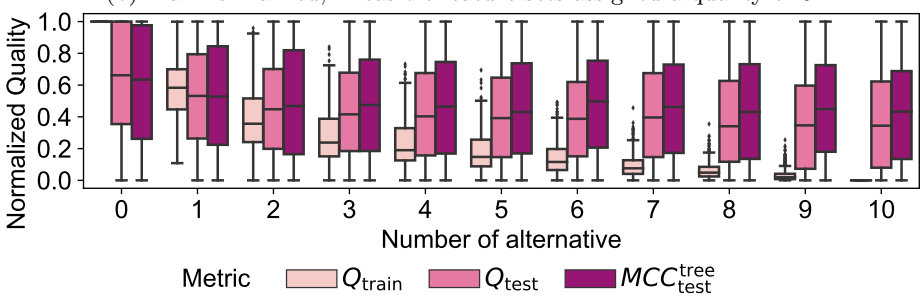
Feature-set quality For sequential search, the training-set objective value has to decrease with the number of alternatives, at least for the feature-selection criteria optimized exactly. In particular, each found feature set constrains the optimization problem further. Figures 3a and 3c illustrate this trend for *MI*-based feature selection. Since feature-set quality varies between datasets (cf. Appendix A.7.1), we additionally normalize feature-set quality here. In particular, we analyze the relative development of feature-set quality within each search run for alternatives. First, we shift the range of all evaluation metrics to $[0, 1]$ since prediction performance and the objectives of *Greedy Wrapper* and *mRMR* have the range $[-1, 1]$ without this shift. Second, we max-normalize feature-set quality for each search of alternatives, i.e., the highest feature-set quality in



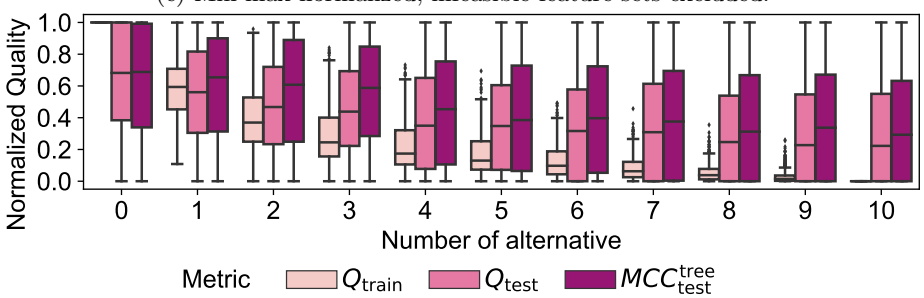
(a) Max-normalized, infeasible feature sets excluded.



(b) Max-normalized, infeasible feature sets assigned a quality of 0.



(c) Min-max-normalized, infeasible feature sets excluded.



(d) Min-max-normalized, infeasible feature sets assigned a quality of 0.

Figure 3: Feature-set quality, normalized per search run for alternatives, over the number of alternatives, by evaluation metric and normalization method. Results from sequential search with *MI* as feature-selection method and $k = 5$.

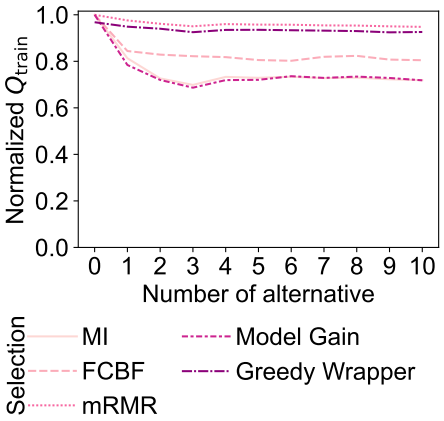
a	Optimization time				
	FCBF	Wrapper	MI	Model Gain	mRMR
1	0.52 s	25.94 s	0.03 s	0.02 s	44.99 s
2	0.95 s	39.44 s	0.09 s	0.08 s	118.80 s
3	3.26 s	56.52 s	0.31 s	0.27 s	208.90 s
4	14.02 s	86.13 s	3.84 s	3.59 s	258.40 s
5	46.71 s	109.20 s	122.69 s	95.94 s	317.69 s

Table 6: Mean optimization time by number of alternatives and feature-selection method. Results from simultaneous search with sum-aggregation and $k = 5$.

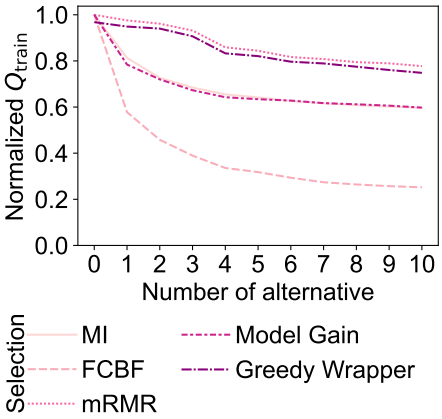
the search run is set to 1, and the other qualities are scaled accordingly. Figure 3a shows that multiple alternatives may have a similar quality, as the median training-set objective value remains relatively stable over the alternatives and is above 0.8 even for the tenth alternative. For comparison, Figure 3c uses min-max normalization, i.e., the worst of the alternatives gets 0 as objective. This figure makes the decrease in quality over the alternatives more visible. In particular, this figure highlights that the training-set objective value decreases most from the original feature set to the first alternative but less beyond.

Additionally, Figures 3a and 3c show that the test-set objective value also drops most to the first alternative. However, this decrease is less prominent than on the training set, and there is no clear trend beyond the first few alternatives. In particular, alternatives can even have a higher test-set objective value than the original feature set due to overfitting. Similar findings hold for test-set prediction performance. Overall, these results indicate that alternative feature sets fulfill their purpose of being different solutions with similar quality.

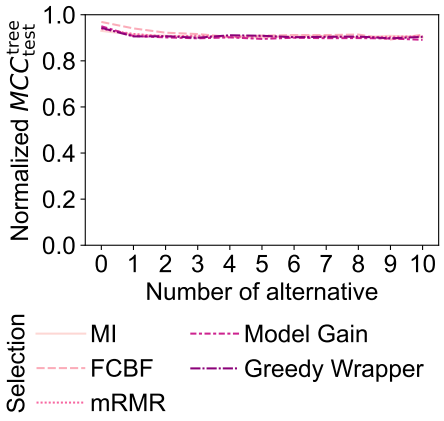
Optimization status The prior observations refer to the quality of the found feature sets. However, the more alternatives are desired, the likelier an infeasible optimization problem is (cf. Table 4). For example, *MI*-based feature selection in sequential search always finds an original feature set. However, with $k = 5$, the problem is infeasible in 2% of the cases for the third alternative, 12% for the fifth, and 17% for the tenth. Increasing the feature-set size k or having lower dataset dimensionality n naturally causes more infeasible solutions, as fewer features become available for alternatives. Thus, even if the quality of found feature sets remains relatively stable for more alternatives, valid alternatives may simply not exist. Figures 3b and 3d show the same data as Figures 3a and 3c but with the quality of infeasible feature sets set to zero, i.e., the theoretical minimum after we shifted the value ranges of evaluation metrics. In these figures, the downward trend of feature-set quality over the alternatives becomes slightly more prominent, particularly for many alternatives. This trend also depends on the dissimilarity threshold τ, which we analyze in the next section.



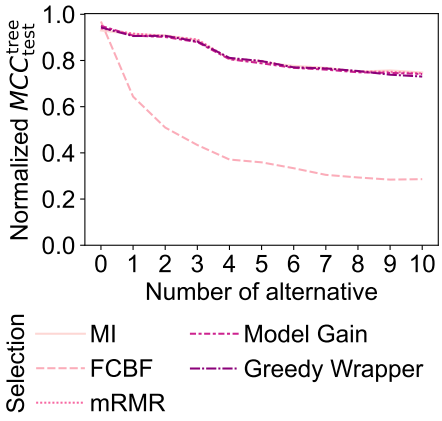
(a) Training-set objective value. Infeasible feature sets excluded.



(b) Training-set objective value. Infeasible feature sets assigned a quality of 0.



(c) Test-set prediction performance. Infeasible feature sets excluded.



(d) Test-set prediction performance. Infeasible feature sets assigned a quality of 0.

Figure 4: Mean of feature-set quality, max-normalized per search run for alternatives, over the number of alternatives, by feature-selection method and evaluation metric. Results from sequential search with $k = 5$.

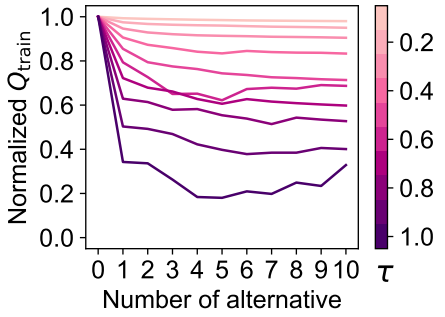
Influence of feature-selection method While we discussed *MI* before, the decrease in objective value over the number of alternatives occurs for all feature-selection methods in our experiments, as Figure 4a displays. The strength of the decrease varies between the feature selection methods. For example, *Greedy Wrapper* and *mRMR* show little effect of increasing the number of alternatives, while *MI* and *Model Gain* exhibit the strongest effect. As Figure 4b displays, the quality decrease becomes more prominent if one sets the quality of infeasible feature sets to zero. Further, for the test-set prediction performance shown in Figure 4c, no feature-selection method exhibits a strong decrease over the number of alternatives, unless we account for infeasible feature sets (cf. Figure 4d).

6.3 Dissimilarity Threshold τ

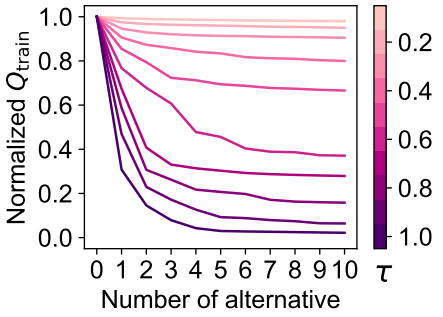
Feature-set quality As Figure 5a shows for *MI* as the feature-selection method, the decrease in the objective value Q over the number of alternatives strongly depends on the dissimilarity threshold τ. We use results with $k = 10$ instead of $k = 5$ here to show more distinct values of τ. For a low dissimilarity threshold, e.g., $\tau = 0.1$, the objective value barely drops over the number of alternatives. In contrast, the objective value decreases significantly for a high dissimilarity threshold, e.g., $\tau = 1$. This trend is expected since a higher τ constrains the feature selection more. As Figure 5c displays, this phenomenon also holds for the test-set objective value, though the dependency on τ is lower there. The effect of τ on prediction performance exhibits an even less clear trend, as visualized in Figure 5e. This result underlines our previous observations that the objective value is only partially indicative of prediction performance.

Optimization status Similar to our analysis for the number of alternatives (cf. Section 6.2), one needs to consider that setting τ too high can make the optimization problem infeasible. In particular, a higher dissimilarity threshold increases the likelihood that no feature set is alternative enough. Figure 6 visualizes the fraction of valid feature sets over the number of alternatives and dissimilarity threshold τ. Figures 5b, 5d, and 5f account for infeasible feature sets by setting their feature-set quality to zero. Compared to Figures 5a, 5c, and 5e, the decrease in feature-set quality is noticeably stronger. In contrast, if only considering valid feature sets, the mean quality can increase over the number of alternatives, as visible in Figure 5a for $\tau = 1.0$ or in Figure 4a for *MI* and *Model Gain*. This counterintuitive phenomenon can occur because some datasets run out of valid feature sets sooner than others, so the average quality may be determined for different sets of datasets at each number of alternatives.

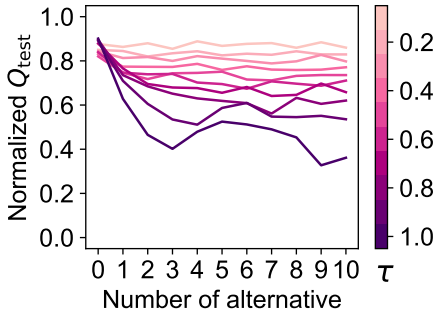
Influence of feature-selection method The impact of τ on feature-set quality varies between feature-selection methods, as Figure 7a shows. Besides *MI*, the objective value of *Model Gain* strongly depends on τ as well. In contrast, the remaining three feature-selection methods exhibit little influence of τ on feature-set quality unless one also accounts for infeasible feature sets (cf. Figure 7b). For *Greedy Wrapper*, this outcome may be explained by the heuristic,



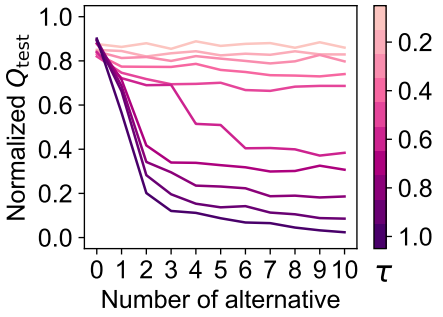
(a) Training-set objective value, max-normalized. Infeasible feature sets excluded.



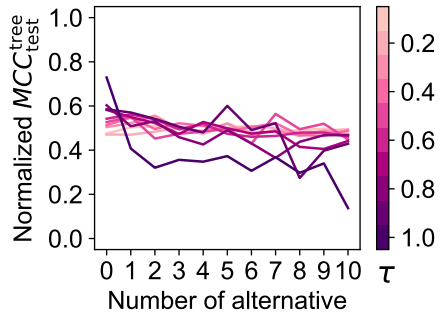
(b) Training-set objective value, max-normalized. Infeasible feature sets assigned a quality of 0.



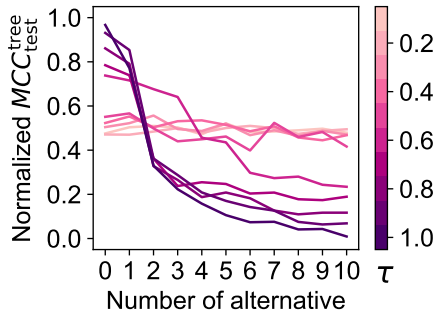
(c) Test-set objective value, max-normalized. Infeasible feature sets excluded.



(d) Test-set objective value, max-normalized. Infeasible feature sets assigned a quality of 0.

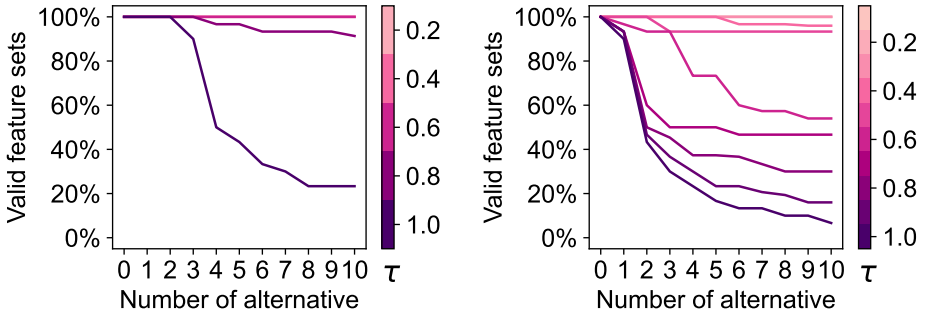


(e) Test-set prediction performance, min-max-normalized. Infeasible feature sets excluded.



(f) Test-set prediction performance, min-max-normalized. Infeasible feature sets assigned a quality of 0.

Figure 5: Mean of feature-set quality, normalized per search run for alternatives, over the number of alternatives and dissimilarity threshold τ, by evaluation metric and normalization method. Results from sequential search with *MI* as feature-selection method and $k = 10$.



(a) Feature-set size $k = 5$.

(b) Feature-set size $k = 10$.

Figure 6: Fraction of optimization runs yielding a valid feature set over the number of alternatives and dissimilarity threshold τ, by feature-set size k. Results from sequential search with *MI* as feature-selection method.

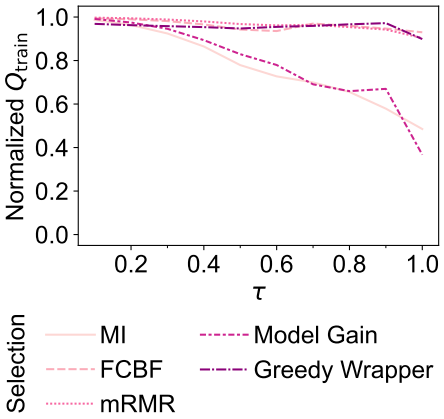
inexact search procedure. For *FCBF*, the additional constraints on feature-feature correlation (cf. Equation 12) may alleviate the effect of τ. For *mRMR*, the low influence of τ matches the low influence of the number of alternatives. For this feature-selection method, alternatives tend to vary little in their objective value. Finally, the test-set prediction performance does not vary considerably over τ for any feature-selection method, as Figure 7c displays. Only considering infeasible feature sets results in decreased prediction performance (cf. Figure 7d).

6.4 Summary

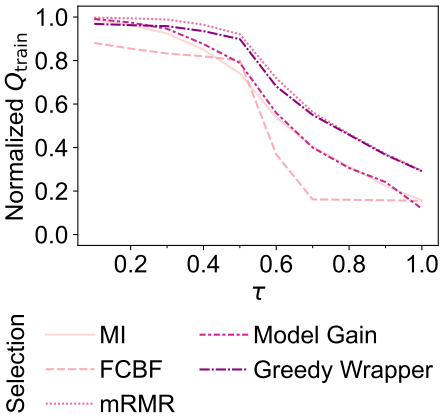
Datasets (cf. Appendix A.7.1) Generally, feature-set quality strongly depended on the dataset. Thus, an analysis of alternative feature sets should be dataset-specific or appropriately normalize quality, as we did.

Feature-set quality metrics (cf. Appendix A.7.2) Different notions of feature-set quality exhibited different trends in our experiments, so one should choose a notion of feature-set quality carefully. In particular, the objective function of feature-selection methods might disagree with the prediction performance of the corresponding feature sets. Further, we observed overfitting, i.e., a gap between training-set quality and test-set quality, also for simple objective functions, though to a lesser extent than for prediction performance.

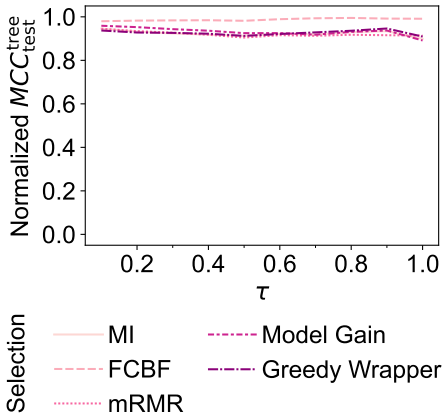
Feature-selection methods (cf. Appendix A.7.3) Among the feature-selection methods, *Model Gain* resulted in the best prediction performance on average, though the simple univariate *MI* also turned out competitive. *Greedy Wrapper* and *mRMR* required high optimization times, while our constraint-based version of *FCBF* yielded many infeasible solutions. Finally, selecting $k = 10$ instead of $k = 5$ features yielded only a slight improvement in prediction



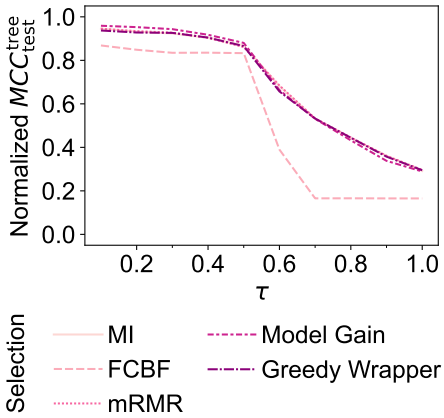
(a) Training-set objective value. Infeasible feature sets excluded.



(b) Training-set objective value. Infeasible feature sets assigned a quality of 0.



(c) Test-set prediction performance. Infeasible feature sets excluded.



(d) Test-set prediction performance. Infeasible feature sets assigned a quality of 0.

Figure 7: Mean of feature-set quality, max-normalized per search run for alternatives, over the dissimilarity threshold τ, by feature-selection method and evaluation metric. Results from sequential search with $k = 10$.

performance for all feature-selection methods, so one might stick to smaller feature-set sizes if such a setting benefits interpretability for users.

Search methods for alternatives (cf. Section 6.1) Simultaneous search, particularly with min-aggregation, considerably reduced the variance of the training-set objective value over alternatives compared to sequential search, as we desired. However, results were less clear on the test set and when considering prediction performance to measure feature-set quality. Further, the average quality of alternatives was similar to sequential search. In addition, the latter was considerably faster and led to less solver timeouts, particularly when increasing the number of alternatives. Also, sequential search allows users to stop searching after each alternative instead of requiring the number of alternatives to be specified beforehand. Thus, we recommend using sequential search.

Number of alternatives a (cf. Section 6.2) Feature-set quality decreased most from the original feature set to the first alternative but less beyond. The strength of this decrease depended on the feature-selection method. There usually were several alternatives of similar quality, if such valid alternatives existed at all. In particular, the frequency of infeasible solutions increased with a due to more constraints. Finally, the quality decrease was more prominent on the training than on the test set.

Dissimilarity threshold τ (cf. Section 6.3) A higher dissimilarity threshold caused a stronger decrease in feature-set quality in terms of objective value for the feature-selection methods *MI* and *Model Gain*. This result shows that users can control a trade-off between quality and dissimilarity. However, results regarding prediction performance and for the other three feature-selection methods were less clear. In any case, a higher τ naturally caused more infeasible solutions, which users should be aware of.

7 Conclusions and Future Work

In this section, we summarize our work (cf. Section 7.1) and give an outlook on potential future work (cf. Section 7.2).

7.1 Conclusions

Feature-selection methods are a valuable tool to foster interpretable predictions. Conventional feature-selection methods typically yield only one feature set. However, users may be interested in obtaining multiple, sufficiently diverse feature sets of high quality. Such alternative feature sets may provide alternative explanations for predictions from the data.

In this article, we defined alternative feature selection as an optimization problem. We formalized alternatives via constraints that are independent of

the feature-selection method, can be combined with other constraints on feature sets, and allow users to control diversity according to their needs. We analyzed the complexity of this optimization problem and proved $\mathcal{NP}$-hardness, even for simple notions of feature-set quality. Further, we discussed how to integrate different categories of conventional feature-selection methods. Finally, we evaluated alternative feature selection with 30 classification datasets and five feature-selection methods. We compared two search methods for alternatives and varied the number of alternatives as well as the threshold for alternatives.

7.2 Future Work

Feature selection (objective function) One could search for alternatives with other feature-selection methods than the five we analyzed. In particular, we implemented only one procedure to find alternatives for wrapper feature selection (cf. Section 3.3.2). Embedded feature selection, which we did not evaluate, would also need adapted search procedures for alternatives (cf. Section 3.3.3).

Alternatives (constraints) One could vary the definition of alternatives, e.g., the set-dissimilarity measure (cf. Section 3.2.1), the quality aggregation for simultaneous alternatives (cf. Appendix A.1), or the overall optimization problem (cf. Section 3.1). While we made general and straightforward decisions for each of these points, particular applications might demand other formalizations of alternatives. E.g., one could use soft instead of hard constraints.

Computational complexity Appendix A.5.4 discusses how one could extend our complexity analysis of alternative feature selection (cf. Section 3.4).

Runtime Our experiments (cf. Section 6.1) and theoretical analyses (cf. Section 3.2.2) revealed that simultaneous search scales poorly with the number of alternatives. One could conceive a more efficient problem formulation. Further, one could limit the solver runtime and take the intermediate results once the timeout is reached. We already used a fixed timeout in our experiments, but studying the exact influence of timeouts on feature-set quality is an open topic. Next, one could use a different solver, e.g., one for non-linear optimization, so the auxiliary variables from Equation 6 become superfluous. Finally, one could employ a heuristic rather than an exact search method (cf. Appendix A.6).

Datasets Our evaluation used datasets from various domains (cf. Section 5.4). While we could uncover several general trends, the existence and quality of alternatives naturally depend on the dataset. Thus, practitioners could use our generic search methods for alternatives in domain-specific case studies.

Acknowledgments This work was supported by the Ministry of Science, Research and the Arts Baden-Württemberg, project *Algorithm Engineering for the Scalability Challenge (AESC).*

A Appendix

In this section, we provide supplementary materials. Section A.1 discusses aggregation operators for the objective of simultaneous search (cf. Equation 10). Section A.2 discusses additional objective functions for multivariate filter feature selection (cf. Section 3.3.1). Section A.3 provides complete definitions of the alternative-feature-selection problem (cf. Section 3.2) for the univariate objective (cf. Equation 11). Section A.4 proposes how to speed up optimization for the univariate objective (cf. Equation 11). Section A.5 complements the complexity analysis (cf. Section 3.4). Section A.6 proposes search heuristics for the univariate objective (cf. Equation 11). Section A.7 contains additional evaluation results (cf. Section 6).

A.1 Aggregation Operators for Simultaneous Search

In this section, we discuss operators to aggregate the feature-set quality of multiple alternatives in the objective of simultaneous search (cf. Equation 10).

Sum-aggregation The arguably simplest way to aggregate the qualities of multiple feature sets is to sum them up, which we call *sum-aggregation*:

$$\max_{s^{(0)},\ldots,s^{(a)}} \sum_{i=0}^{a} Q(s^{(i)}, X, y) \tag{15}$$

While this objective fosters a high average quality of feature sets, it does not guarantee that the alternatives have similar quality:

Example 1 (Sum-aggregation)**.** Consider $n = 6$ features with univariate feature qualities (cf. Equation 11) $q = (9, 8, 7, 3, 2, 1)$, feature-set size $k = 3$, number of alternatives $a = 2$, and dissimilarity threshold $\tau = 0.5$, which permits an overlap of one feature between sets here. Sequential search yields the selection $s^{(0)} = (1, 1, 1, 0, 0, 0)$, $s^{(1)} = (1, 0, 0, 1, 1, 0)$, and $s^{(2)} = (0, 1, 0, 1, 0, 1)$, with a summed quality of $24 + 14 + 12 = 50$. One simultaneous-search solution consists of the feature sets $s^{(0)} = (1, 1, 0, 1, 0, 0)$, $s^{(1)} = (1, 0, 1, 0, 1, 0)$, and $s^{(2)} = (0, 1, 1, 0, 0, 1)$, with a summed quality of $20 + 18 + 16 = 54$. Another simultaneous-search solution is $s^{(0)} = (1, 1, 0, 0, 0, 1)$, $s^{(1)} = (1, 0, 1, 0, 1, 0)$, and $s^{(2)} = (0, 1, 1, 1, 0, 0)$, with a summed quality of $18 + 18 + 18 = 54$.

This example allows several insights. First, sequential search yields worse quality than simultaneous search here, i.e., 50 vs. 54. Second, the feature-set qualities of the sequential solution, i.e., 24, 14, and 12, differ significantly. Third, simultaneous search can yield multiple solutions whose feature-set quality is differently balanced. Here, the feature-set qualities in the second simultaneous-search solution, i.e., 18, 18, and 18, are more balanced than in the first, i.e., 20, 18, and 16. However, both solutions are equally optimal for sum-aggregation.

Min-aggregation To actively foster balanced feature-set qualities in simultaneous search, we propose *min-aggregation* in the objective:

$$\max_{s^{(0)},\ldots,s^{(a)}} \min_{i\in\{0,\ldots,a\}} Q(s^{(i)}, X, y) \tag{16}$$

In the terminology of social choice theory, this objective uses an egalitarian rule instead of a utilitarian one [74]. Note that optimizing the objective with either sum-aggregation or min-aggregation does not necessarily optimize the other. We already showed a solution optimizing sum-aggregation but not min-aggregation (cf. Example 1). In the following, we demonstrate the other direction:

Example 2 (Min-aggregation)**.** Consider $n = 6$ features with univariate feature qualities (cf. Equation 11) $q = (11, 10, 6, 5, 4, 1)$, feature-set size $k = 3$, number of alternatives $a = 1$, and dissimilarity threshold $\tau = 0.5$, which permits an overlap of one feature between sets here. One solution optimizing the objective with min-aggregation is $s^{(0)} = (1, 1, 0, 0, 1, 0)$ and $s^{(1)} = (1, 0, 1, 1, 0, 0)$, with a summed quality of $25 + 22 = 47$. Another solution is $s^{(0)} = (1, 1, 0, 0, 0, 1)$ and $s^{(1)} = (1, 0, 1, 1, 0, 0)$, with a summed quality of $22 + 22 = 44$.

While both solutions have the same minimum quality, only the first solution optimizes the objective with sum-aggregation. In particular, min-aggregation permits reducing the quality of sets above the minimum of all sets.

From the technical perspective, Equation 16 has the disadvantage of being non-linear regarding the decision variables $s^{(0)}, \ldots, s^{(a)}$. However, we can linearize it with one constraint per feature set and an auxiliary variable $Q_{\min}$:

$$\begin{aligned} \max_{s^{(0)},\ldots,s^{(a)}} \quad & & Q_{\min} \\ \text{subject to:} \quad & \forall i \in \{0, \ldots, a\} : & Q_{\min} \leq Q(s^{(i)}, X, y) \\ & & Q_{\min} \in \mathbb{R} \end{aligned} \tag{17}$$

As we maximize $Q_{\min}$, this variable will implicitly assume the actual minimum value of $Q(s^{(i)}, X, y)$ with equality since the solution would not be optimal otherwise. This situation relieves us from introducing further auxiliary variables that are usually necessary when linearizing maximum or minimum expressions [69].

Further approaches for balancing quality Min-aggregation provides no control or guarantee of how much the feature-set qualities will actually differ between alternatives since it only incentives high quality for all sets. One can alleviate this issue by adapting the objective or constraints. First, related work on number partitioning also uses other objectives for balancing [54, 57] (cf. Section A.5.2). E.g., one could minimize the difference between maximum and minimum feature-set quality. Second, one could use sum-aggregation but constrain the minimum or maximum quality of sets, or the difference between the qualities. However, such constraint-based approaches introduce one or several parameters bounding feature-set quality, which are difficult to determine a priori. Third, one could treat balancing qualities as another objective besides

maximizing the summed quality. One can then optimize two objectives simultaneously, filtering results for Pareto-optimal solutions or optimize a weighted combination of the two objectives. In both cases, users may need to define an acceptable trade-off between the objectives. It is an open question if a solution always exists that jointly optimizes min- and sum-aggregation. If yes, then optimizing a weighted combination of the two objectives would also optimize each of them on its own, assuming positive weights.

A.2 Further Objectives for Multivariate Filter Methods

While Section 3.3.1 already addressed FCBF and mRMR as multivariate filter feature-selection methods, we discuss the objectives of CFS and Relief here.

CFS Correlation-based Feature Selection (CFS) [37, 38] follows a similar principle as mRMR but uses the ratio instead of the difference between a relevance term and a redundancy term for feature-set quality. Using a bivariate dependency measure $q(\cdot)$ to quantify correlation, the objective is as follows:

$$Q_{\text{CFS}}(s, X, y) = \frac{\sum_{j=1}^{n} q(X_{\cdot j}, y) \cdot s_j}{\sqrt{\sum_{j=1}^{n} s_j + \sum_{j_1=1}^{n} \sum_{\substack{j_2=1 \\ j_2 \neq j_1}}^{n} q(X_{\cdot j_1}, X_{\cdot j_2}) \cdot s_{j_1} \cdot s_{j_2}}} \tag{18}$$

One can square this objective to remove the square root in the denominator [78]. Nevertheless, the objective remains non-linear in the decision variables s since it involves a fraction and multiplications between variables. However, one can linearize the objective with additional variables and constraints [77, 78], allowing to formulate alternative feature selection for CFS as a linear problem.

Relief Relief [51, 88] builds on the idea that data objects with a similar value of the prediction target should have similar feature values, but data objects that differ in their target should differ in their feature values. Relief assigns a score to each feature by sampling data objects and quantifying the difference in feature values and target values compared to their nearest neighbors. We deem Relief to be multivariate since the nearest-neighbor computations involve all features instead of considering them independently. However, the resulting feature scores can directly be put into the univariate objective (cf. Equation 11) to obtain a linear problem. One can also use Relief scores in CFS to consider feature redundancy [37, 38], which the default Relief does not.

A.3 Complete Specifications of the Optimization Problem for the Univariate Objective

In this section, we provide complete specifications of the alternative-feature-selection problem for sequential and simultaneous search. In particular, we

combine all relevant definitions and equations from Section 3. We use the objective of univariate filter feature selection (cf. Equation 11). The corresponding feature qualities $q(\cdot)$ are constants in the optimization problem. We use the Dice dissimilarity (cf. Equation 8) to measure feature-set dissimilarity for alternatives. The dissimilarity threshold $\tau \in [0, 1]$ is a user-defined constant. Further, we assume fixed, user-defined feature-set sizes $k \in \mathbb{N}$.

Sequential alternatives In the sequential case, only one feature set F_s is variable in the optimization problem, while the existing feature sets $F_{\bar{s}} \in \mathbb{F}$ with their selection vectors $\bar{s}$ are constants.

$$\begin{aligned} \max_{s} \quad & Q_{\text{uni}}(s, X, y) = \sum_{j=1}^{n} q(X_{\cdot j}, y) \cdot s_j \\ \text{subject to:} \quad \forall F_{\bar{s}} \in \mathbb{F} : \quad & \sum_{j=1}^{n} s_j \cdot \bar{s}_j \leq (1 - \tau) \cdot k \\ & \sum_{j=1}^{n} s_j = k \\ & s \in \{0, 1\}^n \end{aligned} \tag{19}$$

Simultaneous alternatives In the simultaneous case, all feature sets are variable. $a \in \mathbb{N}_0$ denotes the number of alternatives, which corresponds to the number of feature sets minus one. Next, we introduce auxiliary variables to linearize products between variables (cf. Equation 6). Finally, we use sum-aggregation (cf. Equation 15) in the objective here.

$$\begin{aligned} \max_{s^{(0)}, \ldots, s^{(a)}} \quad & \sum_i Q_{\text{uni}}(s^{(i)}, X, y) = \sum_i \sum_j q(X_{\cdot j}, y) \cdot s_j^{(i)} \\ \text{subject to:} \quad \forall i_1 \; \forall i_2 : \quad & \sum_j t_j^{(i_1, i_2)} \leq (1 - \tau) \cdot k \\ \forall i_1 \; \forall i_2 \; \forall j : \quad & t_j^{(i_1, i_2)} \leq s_j^{(i_1)} \\ \forall i_1 \; \forall i_2 \; \forall j : \quad & t_j^{(i_1, i_2)} \leq s_j^{(i_2)} \\ \forall i_1 \; \forall i_2 \; \forall j : \quad & 1 + t_j^{(i_1, i_2)} \geq s_j^{(i_1)} + s_j^{(i_2)} \\ \forall i : \quad & \sum_j s_j^{(i)} = k \\ \forall i : \quad & s^{(i)} \in \{0, 1\}^n \\ \forall i_1 \; \forall i_2 : \quad & t^{(i_1, i_2)} \in \{0, 1\}^n \\ \text{with indices:} \quad & i \in \{0, \ldots, a\} \\ & i_1 \in \{1, \ldots, a\} \\ & i_2 \in \{0, \ldots, i_1 - 1\} \\ & j \in \{1, \ldots, n\} \end{aligned} \tag{20}$$

A.4 Pre-Selection for the Univariate Objective

In this section, we describe how to potentially speed up the optimization of the univariate objective (cf. Equation 11) by *pre-selection* if the user-defined feature-set sizes k and the number of alternatives a are small.

The univariate objective is monotonic in the features' qualities $q(X_{\cdot j}, y)$ and the selection decisions s_j. In particular, the objective cannot decrease when selecting more features or replacing a feature with another of higher quality for a fixed feature-set size. Sum-aggregation (cf. Equation 15) and min-aggregation (cf. Equation 16) for simultaneous search are monotonic as well.

Thus, assuming $(a+1) \cdot k < n$, it suffices to use the $(a+1) \cdot k$ highest feature qualities when searching for an optimal solution out of $a+1$ feature sets. Due to monotonicity, the remaining feature qualities cannot improve the objective, so one can drop them before optimization. We call this step *pre-selection*. While there might also be optimal solutions using the dropped features, their objective value cannot be higher than with pre-selection. For example, such solutions can arise in case of multiple identical qualities or for min-aggregation in the objective (cf. Example 2). Also, the optimal solution might not contain all pre-selected features, i.e., pre-selection over-approximates the set of selected features.

One can conduct pre-selection before using a solver or any other search mechanism, e.g., exhaustive search. The latter generally has polynomial runtime regarding n assuming small, constant a and k, i.e., $a \cdot k \in O(1)$ (cf. Section 3.4.1). With pre-selection, the pure search cost would even become independent from n, i.e., $O(1)$ under that assumption. However, one would need to determine the highest feature qualities first, e.g., by sorting all qualities in $O(n \cdot \log n)$ or iteratively determining the maximum quality in $O((a+1) \cdot k \cdot n)$.

A.5 Computational Complexity

In this section, we provide details for our analysis of computational complexity (cf. Section 3.4). In particular, we discuss a special case of exhaustive simultaneous search (cf. Section A.5.1), outline related work (cf. Section A.5.2), provide proofs (cf. Section A.5.3), and describe future work (cf. Section A.5.4).

A.5.1 A Special Case of Exhaustive Simultaneous Search

The complexity of exhaustive simultaneous search is lower than in Proposition 4 for the special case $0 < \tau \cdot k \leq 1$, i.e., if feature sets need to differ in only one feature. There, each feature set is an alternative to each other unless both sets are identical. Thus, each set of $a+1$ distinct feature sets constitutes a valid solution, and further constraint checking is unnecessary. Hence, instead of iterating over sets of feature sets, one can iterate over individual feature sets and maintain a buffer containing the $a+1$ feature sets with the highest quality. For each feature set iterated over, one needs to determine if its quality is higher than the lowest feature-set quality in the buffer and replace it if yes. This procedure has a runtime of $O((a+1) \cdot n^k)$ without the cost of evaluating the objective. I.e.,

unlike in Proposition 4, the number of alternatives a is not part of the exponent anymore, and the cost corresponds to the search for one feature set times the cost of updating the buffer. For large a, one can implement the buffer as a heap, thereby reducing the linear factor regarding a to a logarithmic one.

A.5.2 Related Work

In this section, we discuss related work on $\mathcal{NP}$-hard problems that resemble alternative feature-selection with univariate feature qualities (cf. Equation 11), providing background for Section 3.4.2.

Integer programming The univariate objective and several other feature-selection methods allow us to phrase alternative feature selection as a 0-1 integer linear program (cf. Section 3.3.1). Integer Programming is $\mathcal{NP}$-complete in general, even for binary decision variables [30, 45]. Thus, alternative feature selection with a white-box objective suitable for Integer Programming resides in $\mathcal{NP}$. However, it could still be easier since alternative feature selection only uses particular constraint types instead of expressing arbitrary integer linear problems. Vice versa, the membership in $\mathcal{NP}$ based on Integer Programming assumes a particular encoding of alternative feature selection, i.e., each constraint is stored separately and counts towards the problem's input size. If we instead define the input size only as the number of features n or the total encoding length of the objective function plus parameters a, k, and τ, the problem could be harder than $\mathcal{NP}$, e.g., for a high number of alternatives. In particular, increasing the number of alternatives would increase the encoding length logarithmically but the cost of constraint checking quadratically.

Multi-way number partitioning / multiprocessor scheduling The literature provides different formulations of Multi-Way Number Partitioning and Multiprocessor Scheduling. In particular, different objectives formalize the notion of balanced subset sums and can lead to different optimal solutions [54, 57]. The maximin formulation we use for min-aggregation in simultaneous search is one such notion.

There are several exact algorithms to solve Multi-Way Number Partitioning, e.g., using branch-and-bound approaches that might have exponential runtime [39, 95, 107]. For a fixed number of partitions, the problem is weakly $\mathcal{NP}$-complete since it admits pseudo-polynomial algorithms [30, 53]. Such algorithms run in polynomial time if the input numbers are bounded to a particular size known in advance. Since our feature qualities typically are real numbers, one would need to scale and discretize them to apply such an algorithm. Also, for an arbitrary number of partitions, the problem is strongly $\mathcal{NP}$-complete, so no pseudo-polynomial algorithm can exist unless $\mathcal{P} = \mathcal{NP}$ [30].

However, $\mathcal{NP}$-completeness does not exclude the existence of approximation routines that run in polynomial time and have a guaranteed quality relative to the optimal solution. For example, [1, 24, 112] present such algorithms for

the maximin formulation of Multi-Way Number Partitioning, which corresponds to our objective with min-aggregation. In particular, [1, 112] describe polynomial-time approximation schemes (PTAS), which can provide a solution arbitrarily close to the optimum. However, the runtime depends on the desired approximation ratio and can grow exponentially the more precision is desired. Unless $\mathcal{P} = \mathcal{NP}$, the strong $\mathcal{NP}$-completeness of the problem prevents the existence of a fully polynomial-time approximation scheme (FPTAS), which would only polynomially depend on the precision of approximation [1, 112]. However, an FPTAS does exist for each fixed number of partitions [93]. Further, besides approximations, the problem also has polynomial-time exact algorithms if certain parameters of the problem are fixed, e.g., the number of unique numbers to be partitioned or the largest number [66]. Thus, the problem is fixed-parameter tractable ($\mathcal{FPT}$) for an appropriate definition of 'parameter'.

Balanced number partitioning / k-partitioning While the previous approaches considered sets of arbitrary sizes, there are number-partitioning problems with constrained k as well, e.g., called Balanced Number Partitioning or K-Partitioning. The problem formulations differ in their objective and cardinality constraints, e.g., if equalities or inequalities are used.

For the minimax objective, [4, 65, 117] propose heuristic algorithms, some with approximation guarantees. [4] also provides a bound of the objective value relative to the unconstrained case. Further, there is a PTAS for each fixed set size k [65]. Finally, the problem exhibits a polynomial-time exact algorithm for $k = 2$ [22, 23] and an FPTAS for $k = n/2$ [111].

One can also loosen the cardinality constraints by requiring $\leq k$ instead of $= k$. Further, the cardinality k might vary between partitions. This generalized problem is strongly $\mathcal{NP}$-hard but has heuristics running in polynomial time [46]. In particular, [17] provides an efficient PTAS (EPTAS).

As another problem formulation, [18, 40, 58] use a maximin objective as we do. This objective was rarely addressed in combination with cardinality constraints in the literature [58]. Also, all these three references use $\leq k$ constraints instead of $= k$. Again, this problem is strongly $\mathcal{NP}$-hard [40], but [18, 40, 58] propose approximation algorithms, partly with quality guarantees.

Other partitioning problems There are other $\mathcal{NP}$-complete problems that partition elements into non-overlapping subsets [30]. E.g., Partition [45] asks if one can partition a set of elements with positive integer weights into two subsets with the same subset sum. 3-Partition [30] demands a partitioning into three-element subsets with an identical, predefined subset sum of the elements' positive integer weights. In contrast to these two problems, we do not require alternative feature sets to have the same quality.

Bin covering Bin Covering [3] distributes elements with individual weights into bins such that the number of bins is maximal and the summed weights in

each bin surpass a predefined limit. [57] noted a relationship between Multi-Way Number Partitioning and Bin Covering, which may improve solution approaches for either problem [106, 107]. In our case, we could maximize the number of alternatives such that each feature set's quality exceeds a threshold.

Multiple knapsack Simultaneous search with sum-aggregation, $\tau = 1$, and univariate feature qualities is a special case of the Multiple Knapsack problem [16]. The latter involves knapsacks, i.e., sets with individual capacities, and elements with individual weights and profits. The goal is to assign elements to knapsacks such that the summed profit of selected elements is maximal. Each element can be assigned to at most one knapsack, and the weights of all elements in the knapsack must not violate its capacity. This problem is strongly $\mathcal{NP}$-complete in general, though it exhibits a PTAS [16]. However, our problem is a special case where the feature qualities act as profits, the feature-set sizes are capacities, and each feature has a weight of 1. These uniform weights enable the polynomial-runtime result stated in Proposition 11.

A.5.3 Proofs

In this Section, we provide proofs for propositions from Section 3.4.2.

Proof of Proposition 9

Proof. Let an arbitrary problem instance I of the complete-partitioning problem be given and the feature-set size k be fixed. We add one feature f' to I and keep a, k, and τ as before, obtaining an instance I' of the incomplete-partitioning problem since one feature will not be selected. We choose the quality q' of f' to be lower than the quality of all other features in I. Since the univariate objective with min-aggregation is monotonically increasing, selecting feature f' in the solution of I' does not have any benefit since f' would replace a feature with higher quality. If f' is not selected, then this solution of I' also solves I. However, if the qualities of the resulting alternatives are not equal, f' might be chosen in a set that does not have the minimum quality of all sets since only the latter determines the overall objective value (cf. Example 2). In that case, we replace f' with the remaining feature that was not selected instead; the objective value remains the same, and the solution becomes valid for I. Thus, in any case, we can easily transform a solution for I' to a solution for I.

This argument shows that an algorithm for incomplete partitioning can solve arbitrary complete-partitioning problem instances with negligible computational overhead. Thus, a polynomial-time algorithm for incomplete partitioning could also solve complete partitioning polynomially. However, the latter problem type is $\mathcal{NP}$-complete (cf. Proposition 8), so incomplete partitioning has to be $\mathcal{NP}$-hard. Since checking a solution for incomplete partitioning needs only polynomial time, we obtain membership in $\mathcal{NP}$ and thereby $\mathcal{NP}$-completeness. □

Proof of Proposition 10

Proof. Let an arbitrary problem instance I of the complete-partitioning problem be given and the feature-set size k be fixed. We create another instance I' by adding a new feature f' and increasing the feature-set size to $k' = k + 1$. Further, we set $\tau' = (k' - 1)/k'$, thereby allowing an overlap of at most one feature between feature sets. Also, we choose f' to have a considerably higher quality q' than all other features. The goal is to force the selection of f' in all feature sets such that any other solution would be worse, no matter which other features are selected. One possible choice is $q' = \sum_{j=1}^{n} q_j + \varepsilon$, with $\varepsilon \in \mathbb{R}_{>0}$ being a small positive number, or, if the qualities are integers, $\varepsilon = 1$. This quality q' of f' is higher than of any feature set not containing it. Thus, a solution for I' contains f' in each feature set while the remaining features are part of exactly one feature set. Hence, we remove f' to get feature sets of size $k = k' - 1$ that constitute an optimal solution for the original problem instance I.

This transformation shows how an algorithm for instances with $\tau < 1$ can help solve arbitrary problem instances with $\tau = 1$. Given the $\mathcal{NP}$-completeness of the latter problem, we obtain $\mathcal{NP}$-hardness of the former. □

Adding the proposed f' with a high quality q' enlarges the size of the problem instance. However, the transformation from I to I' still runs in polynomial time and increases the input size by at most a fixed factor. In particular, encoding a problem instance involves n feature qualities and the values of a, k, and τ. Assuming the feature qualities in I have an average encoding size of $c \in \mathbb{R}$, the overall quality encoding has the size $c \cdot n$. As q' roughly equals the sum of all feature qualities, its encoding size is upper-bounded by $c \cdot n$ if we disregard ϵ. The change of k and τ is negligible for the encoding size. In consequence, the input size of I' is at most roughly double the size of I. If we explicitly stored all the constraints instead of only the relevant parameters, we would obtain a similar result: Besides adding q' to the objective, all constraints would accommodate one new feature, independent of its quality, increasing their encoding size from $O(n)$ to $O(n + 1)$, i.e., less than double.

One can extend the reduction above from $\tau' = (k' - 1)/k'$ to all other $\tau > 0$. In particular, for a fixed feature set-size k, there is only a finite number of τ values leading to different set overlaps, i.e., $\tau = \{1/k, \ldots, (k - 1)/k\}$. The highest overlap except $\tau = 0$ requires creating an instance I' with $\tau' = 1/k$ from an instance with $\tau = 1$. For this purpose, $k^2 - k$ features need to be added since $\tau' = k/k' = k/(k + k^2 - k) = 1/k$. I.e., k out of $k' = k^2$ features need to form a complete partitioning, while the remaining $k^2 - k$ features occur in each feature set and will be removed after solving I'. The maximum number of features to be added is polynomial in k and thereby also polynomial in n.

Proof of Proposition 11

Proof. For a complete partitioning, we must use each of the n features exactly once. How we distribute the features among sets does not change the objective value, which is the sum of all n qualities in any case. We only need to

ensure that each feature set satisfies cardinality constraints if the latter exist. Thus, 'searching' for alternatives amounts to iterating over the features once to assigning them to the feature sets. Hence, the time complexity is $O(n)$.

For an incomplete partitioning, we use the monotonicity of the univariate objective with sum-aggregation (cf. Section A.4) and order the features decreasingly by their individual quality. Next, we pick features without replacement until we have the desired number of alternatives with the desired feature-set sizes. Again, assigning features to sets does not matter for the objective value. Due to the quality-based sorting, the time complexity is $O(n \cdot \log n)$. If only a small fraction of features is used, one might slightly improve complexity by iteratively picking the maximum instead of sorting all qualities. □

A.5.4 Future Work

In this section, we outline future work on alternative feature selection from the complexity-theory perspective, supplementing the Sections 3.4 and 7.2.

Scenarios of alternative feature selection Our prior complexity analyses focused on special cases of alternative feature selection. E.g., while we obtained $\mathcal{NP}$-hardness for min-aggregation with feature-set overlap (cf. Proposition 10), an analysis of sum-aggregation with overlap is open, even for sequential search. Sum-aggregation admits polynomial runtime for $\tau = 1$ (cf. Proposition 11), but this result might not extend to $\tau < 1$. In particular, $\tau < 1$ increases the number of solution candidates, which could negatively affect the runtime.

Further, our complexity analyses mostly assumed univariate feature qualities. Other feature-selection methods can reside in different complexity classes.

Complexity classes For analyzing other scenarios of alternative feature selection, several questions spring to mind. First, one could establish a complexity result like $\mathcal{NP}$-hardness or membership in $\mathcal{P}$. In the former case, there might be pseudo-polynomial approaches or (F)PTAS. As a first step in that direction, we show membership in complexity class $\mathcal{APX}$ under certain conditions (cf. Proposition 13), i.e., there are polynomial-time algorithms yielding constant-factor approximations. One might attempt to tighten the quality bounds we derived. Further, there might be efficient exact or approximate algorithms for certain types of problem instances, e.g., satisfying additional assumptions regarding feature-set quality or the parameters k, a, and τ. Finally, while we placed alternative feature selection in class $\mathcal{XP}$ (cf. Proposition 5), one might prove membership or hardness for more specific parameterized complexity classes.

Related problem formulations We only focused on the optimization problem of alternative feature selection until now. Another interesting question is how many alternatives exist for a given n, k, and τ, regardless of their quality. Also, given the number of alternatives as well, it would be interesting to have an exact or approximate estimate for the number of valid solutions for alternative feature selection, i.e., sets of feature sets. While both these estimates are

straightforward for $\tau = 1$, allowing arbitrary τ poses a larger challenge. Finally, one could re-formulate alternative feature selection similar to Bin Covering (cf. Section A.5.2) and analyze this problem in detail.

A.6 Heuristic Search for the Univariate Objective

In this section, we propose heuristic search methods for the univariate objective (cf. Equation 11 and Section A.3), complementing the exact, solver-based search methods that we evaluate in our experiments (cf. Section 6.1). The proposed heuristics may be faster than exact optimization at the expense of lower feature-set quality. In particular, we describe *Greedy Replacement Search* (cf. Section A.6.1), *Greedy Balancing Search* (cf. Section A.6.2), and *Greedy Depth Search* (cf. Section A.6.3). The second search method is simultaneous, while the other two are sequential. All three heuristics leverage that the univariate objective sums up the individual qualities q_j of selected features and does not consider interactions between features.

A.6.1 Greedy Replacement Search

Greedy Replacement Search is our first heuristic for alternative feature selection with the univariate objective. This heuristic conducts a sequential search.

Algorithm Algorithm 2 outlines *Greedy Replacement Search*. We start by sorting the features decreasingly based on their qualities q_j (Line 1). For a fixed feature-set size k, a dissimilarity threshold τ, and using the Dice dissimilarity (cf. Equation 3), one subset with $\lfloor (1-\tau) \cdot k \rfloor$ features can be contained in all alternatives without violating the dissimilarity threshold (cf. Equation 8). Thus, our algorithms indeed selects the $\lfloor (1-\tau) \cdot k \rfloor$ features with highest quality in each alternative $s^{(\cdot)}$ (Lines 2–7). We fill the remaining spots in the sets by iterating over the alternatives and remaining features (Lines 8–15). For each alternative, we select the $\lceil \tau \cdot k \rceil$ highest-quality features not used in any prior alternative, thereby satisfying the dissimilarity threshold. We continue this procedure until we reach the desired number of alternatives a or until there are not enough unused features to form further alternatives (Line 9).

Example 3 (Algorithm of *Greedy Replacement Search*)**.** With $n = 10$ features, feature-set size $k = 5$, and $\tau = 0.4$, each feature set must differ by $\lceil \tau \cdot k \rceil = 2$ features from the other feature sets. The original feature set $s^{(0)}$ consists of the top $k = 5$ features regarding quality q_j. The first alternative $s^{(1)}$ consists of the top $\lfloor (1-\tau) \cdot k \rfloor = 3$ features plus the sixth- and seventh-best feature. The second alternative $s^{(2)}$ consists of the top three features plus the eighth- and ninth-best one. The algorithm has to stop at $i = 2$ since there are not enough unused features to form further alternatives in the same manner.

In general, i-th alternative consists of the top $\lfloor (1-\tau) \cdot k \rfloor$ features plus the features $k + (i-1) \cdot \lceil \tau \cdot k \rceil + 1$ to $k + i \cdot \lceil \tau \cdot k \rceil$ in descending quality order.

Algorithm 2: *Greedy Replacement Search* for alternative feature sets.

Input: Univariate feature qualities q_j with $j \in \{1, \ldots, n\}$,
Feature-set size k,
Number of alternatives a,
Dissimilarity threshold τ

Output: List of feature-selection decision vectors $s^{(\cdot)}$

```
1  indices ← sort_indices(q, order=descending)   // Order by qualities
2  s ← {0}^n                                     // Initial selection for all alternatives
3  feature_position ← 1                          // Index of index of current feature
4  while feature_position ≤ ⌊(1 − τ) · k⌋ do
5  |  j ← indices[feature_position]              // Index feature by quality
6  |  s_j ← 1
7  |  feature_position ← feature_position + 1
8  i ← 0                                         // Number of current alternative
9  while i ≤ a and i ≤ (n−k)/⌈τ·k⌉ do
10 |  s^(i) ← s                                  // Select top ⌊(1 − τ) · k⌋ features
11 |  for _ ← 1 to ⌈τ · k⌉ do                    // Select remaining ⌈τ · k⌉ features
12 |  |  j ← indices[feature_position]
13 |  |  s_j^(i) ← 1
14 |  |  feature_position ← feature_position + 1
15 |  i ← i + 1
16 return s^(0), ..., s^(i)
```

Complexity Sorting the qualities of n features (Line 1) has a complexity of $O(n \cdot \log n)$. Next, the algorithm iterates over the features and processes each feature at most once. In particular, after selecting a feature in an alternative, *feature_position* increases by 1. The maximum value of this variable depends on a and k (Line 9) but cannot exceed the total number of features n. For each *feature_position*, the algorithm accesses the arrays *indices* and $s^{(i)}$ (Lines 11–14). Further, each alternative $s^{(i)}$ gets initialized as the selection s of the top $\lfloor (1 - \tau) \cdot k \rfloor$ features (Line 10), which the algorithm only needs to determine once before the main loop (Lines 2–7). Each of these array operations runs in $O(n)$ or faster. Combining the cost per *feature_position* with the number of *feature_position*s, the overall time complexity is $O(n^2)$, i.e., polynomial in n.

Quality While not optimizing exactly, *Greedy Replacement Search* still offers an approximation guarantee relative to exact search methods:

Proposition 12 (Approximation quality of *Greedy Replacement Search*)**.** *Assume non-negative univariate feature qualities of n features, $a \in \mathbb{N}_0$ alternatives, a dissimilarity threshold τ, desired feature-set size k, and $k + a \cdot \lceil \tau \cdot k \rceil \leq n$. Under these conditions,* Greedy Replacement Search *reaches at least a fraction*

of $\frac{\lfloor(1-\tau)\cdot k\rfloor}{k}$ *of the optimal objective values of the optimization problems for (1) sequential search, (2) simultaneous search with sum-aggregation, and (3) simultaneous search with min-aggregation.*

Proof. In the univariate objective, the quality of a feature set is the sum of the qualities of the contained features. *Greedy Replacement Search* includes the $\lfloor(1-\tau)\cdot k\rfloor$ highest-quality features in each alternative of size k, while the remaining $\lceil\tau\cdot k\rceil$ features may have an arbitrary quality. In comparison, the single, i.e., unconstrained, optimal feature set of size k contains the top k features, which are the union of the top $\lfloor(1-\tau)\cdot k\rfloor$ features and the next-best $\lceil\tau\cdot k\rceil$ features. Due to quality sorting, each of the next-best $\lceil\tau\cdot k\rceil$ features has at most the quality of each of the top $\lfloor(1-\tau)\cdot k\rfloor$ features. Hence, assuming non-negative qualities, each alternative yielded by *Greedy Replacement Search* has at least a quality of $\lfloor(1-\tau)\cdot k\rfloor/k$ relative to the single optimal feature set of size k. Next, the single optimal feature set of size k upper-bounds the quality of any individual feature set of size k found by any search method. Thus, the bound also applies to the minimum and sum of qualities over feature sets. □

In particular, *Greedy Replacement Search* yields a constant-factor approximation for the three optimization problems (cf. Equation 9 and 10) mentioned in Proposition 12. The condition $k+a\cdot\lceil\tau\cdot k\rceil\leq n$ describes scenarios where *Greedy Replacement Search* can yield all desired alternatives, i.e., does not run out of unused features. As the heuristic has polynomial runtime, alternative feature selection lies in the complexity class $\mathcal{APX}$ [48] under the specified conditions:

Proposition 13 (Approximation complexity of alternative feature selection)**.** *Assume non-negative univariate feature qualities of n features, $a\in\mathbb{N}_0$ alternatives, a dissimilarity threshold τ, desired feature-set size k, and $k+a\cdot\lceil\tau\cdot k\rceil\leq n$. Under these conditions, the optimization problems of alternative feature selection with (1) sequential search, (2) simultaneous search with sum-aggregation, and (3) simultaneous search with min-aggregation reside in the complexity class $\mathcal{APX}$.*

For $\tau=1$, *Greedy Replacement Search* even yields the same objective values as sequential search and simultaneous search with sum-aggregation since it becomes identical to a procedure we outlined in our complexity analysis earlier (cf. Proposition 11). In contrast, the following example shows that the heuristic can be worse than exact sequential search for as few as $a=2$ alternatives:

Example 4 (Quality of *Greedy Replacement Search* vs. exact search)**.** Consider $n=6$ features with univariate feature qualities $q=(9,8,7,3,2,1)$, feature-set size $k=2$, number of alternatives $a=2$, and dissimilarity threshold $\tau=0.5$, which permits an overlap of one feature between sets here. Sequential search and simultaneous search, for min- and sum-aggregation, yield the selection $s^{(0)}=(1,1,0,0,0,0)$, $s^{(1)}=(1,0,1,0,0,0)$, and $s^{(2)}=(0,1,1,0,0,0)$, with a summed quality of $17+16+15=48$. *Greedy Replacement Search* yields the selection $s^{(0)}=(1,1,0,0,0,0)$, $s^{(1)}=(1,0,1,0,0,0)$, and $s^{(2)}=(1,0,0,1,0,0)$, with a summed quality of $17+16+12=45$.

While the first two feature sets are identical between exact and heuristic search, the quality of $s^{(2)}$ is lower for the heuristic (12 vs. 15). In particular, by always selecting the top $\lfloor (1-\tau) \cdot k \rfloor$ features, the heuristic misses out on feature sets only involving the next-best features.

For min-aggregation in the objective, $a = 1$ alternative already suffices such that the heuristic may be worse than exact search:

Example 5 (Quality of *Greedy Replacement Search* vs. min-aggregation)**.** Consider $n = 6$ features with univariate feature qualities $q = (9, 8, 7, 3, 2, 1)$, feature-set size $k = 3$, number of alternatives $a = 1$, and dissimilarity threshold $\tau = 0.5$, which permits an overlap of one feature between sets here. Simultaneous search with min-aggregation yields the selection $s^{(0)} = (1, 1, 0, 0, 1, 0)$ and $s^{(1)} = (1, 0, 1, 1, 0, 0)$, with a quality of $\min\{19, 19\} = 19$. *Greedy Replacement Search* and sequential search yield the selection $s^{(0)} = (1, 1, 1, 0, 0, 0)$ and $s^{(1)} = (1, 0, 0, 1, 1, 0)$, with a quality of $\min\{24, 14\} = 14$. Simultaneous search with sum-aggregation may yield either of these two solutions or the selection $s^{(0)} = (1, 1, 0, 1, 0, 0)$ and $s^{(1)} = (1, 0, 1, 0, 1, 0)$ with the same summed quality.

In particular, *Greedy Replacement Search* does not balance feature-set qualities since it is a sequential search method. We alleviate this issue with the heuristic *Greedy Balancing Search* (cf. Section A.6.2).

Limitations Proposition 12 and Examples 4, 5 already showed the potential quality loss of the heuristic compared to an exact search for alternatives. Further, *Greedy Replacement Search* only works as long as some features have not been part of any feature set yet, i.e., $k + a \cdot \lceil \tau \cdot k \rceil \leq n$. Once the heuristic runs out of unused features, one would need to switch the search method. Thus, to obtain a high number of alternatives a, the following conditions are beneficial for the heuristic: The number of features n should be high, the feature-set size k show be low, and the dissimilarity threshold τ should be low. These conditions align well with typical feature-selection scenarios where $k \ll n$.

Another drawback is that *Greedy Replacement Search* assumes a very simple structure of the optimization problem. If the objective function becomes more complex than a sum of univariate qualities, quality-based feature ordering may be impossible or suboptimal. Further, *Greedy Replacement Search* cannot accommodate additional constraints on feature sets, e.g., based on domain knowledge. Finally, the heuristic assumes the same size k for all feature sets.

Given the limitations of *Greedy Replacement Search* and the low optimization time for exact sequential search with the univariate objective (cf. Table 5), we do not evaluate this heuristic in our experiments in Section 6.

A.6.2 Greedy Balancing Search

Greed Balancing Search modifies *Greedy Replacement Search* to obtain more balanced feature-set qualities with a simultaneous search procedure.

Algorithm 3: *Greedy Balancing Search* for alternative feature sets.

Input: Univariate feature qualities q_j with $j \in \{1, \ldots, n\}$,
Feature-set size k,
Number of alternatives a,
Dissimilarity threshold τ

Output: List of feature-selection decision vectors $s^{(0)}, \ldots, s^{(a)}$

if $\lceil \tau \cdot k \rceil \cdot a + k > n$ **then**
  return $\emptyset$
$indices \leftarrow$ sort_indices(q, order=descending) `// Order by qualities`
for $i \leftarrow 0$ **to** a **do** `// Initial selection for all alternatives`
  $s^{(i)} \leftarrow \{0\}^n$
$feature\_position \leftarrow 1$ `// Index of index of current feature`
while $feature\_position \leq \lfloor (1 - \tau) \cdot k \rfloor$ **do** `// Select top features`
  $j \leftarrow indices[feature\_position]$ `// Index feature by quality`
  for $i \leftarrow 0$ **to** a **do** `// Same features in all alternatives`
    $s_j^{(i)} \leftarrow 1$
  $feature\_position \leftarrow feature\_position + 1$
for $i \leftarrow 0$ **to** a **do**
  $Q^{(i)} \leftarrow 0$ `// Relative quality of each alternative`
while $feature\_position \leq \lceil \tau \cdot k \rceil \cdot a + k$ **do** `// Fill all positions`
  $Q_{\min} \leftarrow \infty$ `// Find alternative with lowest quality`
  $i_{\min} \leftarrow -1$
  for $i \leftarrow 0$ **to** a **do**
    if $Q^{(i)} < Q_{min}$ ***and*** $\sum_{j=1}^{n} s_j^{(i)} < k$ **then** `// Check cardinality`
      $Q_{\min} \leftarrow Q^{(i)}$
      $i_{\min} \leftarrow i$
  $j \leftarrow indices[feature\_position]$ `// Index feature by quality`
  $s_j^{(i_{\min})} \leftarrow 1$ `// Add to lowest-quality, non-full alternative`
  $Q^{(i_{\min})} \leftarrow Q^{(i_{\min})} + q_j$ `// Update quality of that alternative`
  $feature\_position \leftarrow feature\_position + 1$
return $s^{(0)}, \ldots, s^{(a)}$

Algorithm Algorithm 3 outlines *Greedy Balancing Search*. First, we check whether the algorithm should terminate early, i.e., whether the number of features n is not high enough to satisfy the desired user parameters k, a, and τ (Line 1). Next, we select the first $\lfloor (1-\tau) \cdot k \rfloor$ features in each alternative like in *Greedy Replacement Search* (cf. Algorithm 2), i.e., we pick the features with the highest quality q_j (Lines 3–11).

For the remaining spots in the alternatives, we use a Longest Processing Time (LPT) heuristic (Lines 12–24). Such heuristics are common for Multiprocessor Scheduling and Balanced Number Partitioning problems [4, 18, 58] (cf. Section A.5.2). In particular, we continue iterating over features by decreasing quality. We assign each feature to the alternative that currently has the lowest summed quality $Q^{(i)}$ and whose size k has not been reached yet. We continue this procedure until all alternatives have reached size k (Line 14).

Example 6 (Algorithm of *Greedy Balancing Search*)**.** Consider $n = 6$ features with univariate feature qualities $q = (9, 8, 7, 3, 2, 1)$, feature-set size $k = 4$, number of alternatives $a = 1$, and dissimilarity threshold $\tau = 0.5$, which permits an overlap of two features between sets here. The features with qualities 9 and 8 become part of both feature sets, $s^{(0)}$ and $s^{(1)}$ (Lines 3–11). At this point, both alternatives have the same relative quality $Q^{(0)} = Q^{(1)} = 0$, i.e., $Q^{(i)}$ in the algorithm ignores the quality of the shared features. Now the LPT heuristic becomes active (Lines 12–24). The feature with quality 7 is added to $s^{(0)}$, which causes $Q^{(0)} > Q^{(1)}$ (i.e., $7 > 0$). Thus, the feature with quality 3 is added to $s^{(1)}$. As $Q^{(0)} > Q^{(1)}$ (i.e., $7 > 3$) still holds, the feature with quality 2 becomes part of $s^{(1)}$ as well. Because $s^{(1)}$ has reached size $k = 4$, the feature with quality 1 is added to $s^{(0)}$, even if the latter still has a higher quality (i.e., $7 > 5$). Now both alternatives have reached their desired size and $n = 6 = \lceil 0.5 \cdot 4 \rceil \cdot 1 + 4 = \lceil \tau \cdot k \rceil \cdot a + k$ (Line 14). Thus, the algorithm terminates. The solution consists of $s^{(0)} = (1, 1, 1, 0, 0, 1)$ and $s^{(1)} = (1, 1, 0, 1, 1, 0)$.

Complexity Like *Greedy Replacement Search*, *Greedy Balancing Search* has an upfront cost of $O(n \cdot \log n)$ for sorting feature qualities (Line 3) and then iterates over $O(n)$ $feature_position$s. For each $feature\_position$, the algorithm iterates over a alternatives and conducts a fixed number of array operations in $O(n)$. Thus, the overall complexity of *Greedy Balancing Search* is $O(a \cdot n^2)$.

Quality *Greed Balancing Search* selects the same features as *Greedy Replacement Search* and only changes their assignment to the feature sets. Thus, the summed feature-set quality remains the same, while the minimum feature-set quality may be higher due to balancing. Hence, the quality guarantee of *Greedy Replacement Search* (cf. Proposition 12) holds here as well:

Proposition 14 (Approximation quality of *Greedy Balancing Search*)**.** *Assume non-negative univariate feature qualities of n features, $a \in \mathbb{N}_0$ alternatives, a dissimilarity threshold τ, desired feature-set size k, and $k + a \cdot \lceil \tau \cdot k \rceil \leq n$. Under these conditions,* Greedy Balancing Search *reaches at least a fraction of*

$\frac{\lfloor(1-\tau)\cdot k\rfloor}{k}$ *of the optimal objective values of the optimization problems for (1) sequential search, (2) simultaneous search with sum-aggregation, and (3) simultaneous search with min-aggregation.*

For the objective with min-aggregation, *Greedy Balancing Search* can even be better than exact sequential search, as Example 5 shows, where the heuristic would yield the same solution as simultaneous search with min-aggregation. However, the heuristic can also be worse than sequential and simultaneous search, as Example 4 shows, where *Greedy Balancing Search* would yield the same solution as *Greedy Replacement Search*.

Limitations *Greedy Balancing Search* shares several limitations with *Greedy Replacement Search*, e.g., it may be worse than exact search, assumes univariate feature qualities, and does not work if the number of features n is too low relative to k, a, and τ. In the latter case, *Greedy Balancing Search* yields no solution due to its simultaneous nature, while *Greedy Replacement Search* yields at least some alternatives. However, if running out of features is not an issue, *Greedy Balancing Search* has the advantage of more balanced feature-set qualities.

A.6.3 Greedy Depth Search

Greedy Depth Search is a sequential search heuristic that generalizes *Greedy Replacement Search* and allows to obtain more than $\frac{n-k}{\lceil\tau\cdot k\rceil}$ alternatives.

Algorithm Algorithm 4 outlines *Greedy Depth Search*. As in the other two heuristics, we start by sorting the features decreasingly according to their qualities q_j (Line 1). However, instead of keeping the same $\lfloor(1-\tau)\cdot k\rfloor$ features in each alternative and only replacing the remaining ones, we now allow all features to be replaced. In particular, we may exhaustively iterate over all feature sets, depending on the number of alternatives a. Thus, we maintain not only one feature position as before but a length-k array of the feature positions for the current feature set (Lines 2–4). This array represents feature indices regarding the sorted qualities and is sorted increasingly, which prevents evaluating the same feature set, only with different feature order, multiple times.

In the main loop of the algorithm, we find alternatives sequentially (Lines 7–24). For each potential alternative, we select the features based on the position array (Lines 8–11). We check the resulting feature set against the constraints for alternatives (Line 12) and only store it if it is valid. This check was unnecessary in the other two heuristics, which only formed valid alternatives by design.

Next, we update the feature positions for the next potential alternative (Lines 14–24). First, we try to replace the lowest-quality feature in the current feature set by advancing one position in the sorted qualities. This step may not be possible, as the feature set may already contain the feature with the overall lowest quality, i.e., position n in the array of sorted qualities (Line 17). In this case, we try to replace the second-lowest-quality feature in the current feature set by advancing its position. If this action is impossible as well, we iterate

Algorithm 4: *Greedy Depth Search* for alternative feature sets.

Input: Univariate feature qualities q_j with $j \in \{1, \ldots, n\}$,
Feature-set size k,
Number of alternatives a,
Dissimilarity threshold τ

Output: List of feature-selection decision vectors $s^{(\cdot)}$

```
indices ← sort_indices(q, order=descending)  // Order by qualities
feature_positions ← {0}^k        // Indices of indices of features
for p ← 1 to k do                    // Start with top k features
    feature_positions[p] ← p      // Ordered by qualities as well
i ← 0                             // Number of current alternative
has_next_solution ← true
while i ≤ a and has_next_solution do
    s^(i) ← {0}^n
    for p ← 1 to k do  // Select k features, indexed by quality
        j ← indices[feature_positions[p]]
        s_j^(i) ← 1
    if is_valid_alternative(s^(i), {s^(0), ..., s^(i−1)}) then
        i ← i + 1      // Else, s^(i) overwritten in next iteration
    p ← k      // Update feature positions, starting with last
    while p ≥ 1 do
        position ← feature_positions[p]
        if position < n + p − k then  // Position can be increased
            for Δ_p ← 0 to k − p do // Also update later positions
                feature_positions[p + Δ_p] ← position + Δ_p + 1
            p ← −1                        // Position update finished
        else                          // Position cannot be increased
            p ← p − 1 // Also update at least one prior position
    if p = 0 then // Updating positions further would violate n
        has_next_solution ← false
return s^(0), ..., s^(i)
```

further over positions in the current feature set by increasing quality (Line 22). Once we find a feature position that we can increase, we also advance all subsequent, i.e., lower-quality, positions accordingly. Hence, the feature positions remain sorted by decreasing quality (Lines 18–19).

We repeat the main loop until we reach the desired number of alternatives a or until we cannot update any feature position without exceeding the number of features n, i.e., we cannot form another alternative (Lines 7 and 23).

Example 7 (Algorithm of *Greedy Depth Search*)**.** Consider $n = 6$ features with univariate feature qualities $q = (9, 8, 7, 3, 2, 1)$, feature-set size $k = 4$, number of alternatives $a = 1$, and dissimilarity threshold $\tau = 0.5$, which permits an overlap of two features between sets here. Note that the features are already ordered by quality here, i.e., $indices = (1, 2, 3, 4, 5, 6)$ (Line 1). Next, the algorithm initializes $feature\_positions = (1, 2, 3, 4)$ (Line 2–4). $s^{(0)}$ contains these k features, i.e., $s^{(0)} = (1, 1, 1, 1, 0, 0)$. Given that there are no other alternatives yet, this feature set is valid (Line 12)) and the algorithm moves on to $i = 1$.

For forming $s^{(1)}$, the position-update step (Lines 14–24) first tries to only replace the lowest-quality feature in the alternative, i.e., $feature\_positions = (1, 2, 3, 5)$ and $feature\_positions = (1, 2, 3, 6)$. However, neither of these feature sets constitutes a valid alternative regarding $s^{(0)}$. Thus, the algorithm attempts to replace the feature with the second-lowest quality as well, evaluating $feature\_positions = (1, 2, 4, 5)$ and $feature\_positions = (1, 2, 4, 6)$. However, the overlap with $s^{(0)}$ is still too large. The next value is $feature\_positions = (1, 2, 5, 6)$, which yields the valid alternative $s^{(1)} = (1, 1, 0, 0, 1, 1)$.

Greedy Replacement Search would terminate now since the options for replacing the $\lceil \tau \cdot k \rceil = 2$ lowest-quality features are exhausted. In contrast, *Greedy Depth Search* attempts to replace the third-lowest-quality feature, starting with $feature\_positions = (1, 3, 4, 5)$. This feature set is not a valid alternative, and neither are the subsequent sets with $feature\_positions = (1, 3, 4, 6)$, $feature\_positions = (1, 3, 5, 6)$, etc. After more iterations, the algorithm also replaces the highest-quality feature, starting with $feature\_positions = (2, 3, 4, 5)$. Eventually, the algorithm reaches $feature\_positions = (3, 4, 5, 6)$, which yields the valid alternative $s^{(2)} = (0, 0, 1, 1, 1, 1)$. After obtaining $s^{(2)}$, there is no valid update of the feature positions left (Line 23). Thus, the algorithm terminates.

Complexity The runtime behavior differs from the other two heuristics. In particular, *Greedy Replacement Search* has the same runtime cost between subsequent alternatives since it directly creates valid alternatives by design. In contrast, *Greedy Depth Search* iterates over all possible feature sets, and the runtime between valid alternatives may vary. For each values of $feature\_positions$, the algorithm creates a feature selection in $O(k \cdot n)$ (Lines 8–11), checks constraints in $O(a \cdot n)$ (Line 12), and updates the position array in $O(k^2)$ (Lines 14–24). However, there are $O(n^k)$ potential $feature\_positions$, and *Greedy Depth Search* may exhaustively iterate over them. This cost is comparable to exhaustive conventional feature selection (cf. Proposition 2) and exhaustive sequential search

(cf. Proposition 3). Unlike the latter, the search does not restart for each alternative, i.e., it only considers each feature set once instead of $a + 1$ times.

On the positive side, *Greedy Depth Search* can yield more alternatives than *Greedy Replacement Search* with its $O(n^2)$ cost or *Greedy Balancing Search* with its $O(a \cdot n^2)$ cost. Nevertheless, in scenarios where the latter two are applicable, i.e., $k + a \cdot \lceil \tau \cdot k \rceil \leq n$, they have a lower cost than *Greedy Depth Search*. In particular, *Greedy Depth Search* needs $O(n^{\lceil \tau \cdot k \rceil})$ iterations to cover the options for replacing the worst $\lceil \tau \cdot k \rceil$ features in size-k feature sets, which is the search space of the other two heuristics. In particular, the cost disadvantage relative to the other two heuristics grows with the dissimilarity threshold τ. As a remedy, one may use *Greedy Replacement Search* for as many alternatives as possible and then continue with *Greedy Depth Search*, initializing the $feature\_positions$ (Line 2–4) based on the results of the former heuristic.

Quality *Greedy Depth Search* initially yields the same solutions as *Greedy Replacement Search*. Thus, *Greedy Depth Search* also yields a constant-factor approximation of the optimal solution in case $k + a \cdot \lceil \tau \cdot k \rceil \leq n$ (cf. Proposition 12). The quality analysis becomes more involved for further alternatives since these do not contain all top $\lfloor (1 - \tau) \cdot k \rfloor$ features anymore, on which our proof of Proposition 12 builds. Thus, we leave this analysis open for future work. The quality of alternatives may not even be monotonically decreasing anymore, as the following example shows:

Example 8 (Non-monotonic quality of *Greedy Depth Search*)**.** Consider $n = 4$ features with univariate feature qualities $q = (9, 8, 7, 1)$, feature-set size $k = 2$, number of alternatives $a = 3$, and dissimilarity threshold $\tau = 0.5$, which permits an overlap of one feature between sets here. *Greedy Depth Search* yields the the selection $s^{(0)} = (1, 1, 0, 0)$, $s^{(1)} = (1, 0, 1, 0)$, $s^{(2)} = (1, 0, 0, 1)$, and $s^{(3)} = (0, 1, 1, 0)$, with the corresponding feature-set qualities 17, 16, 10, and 15.

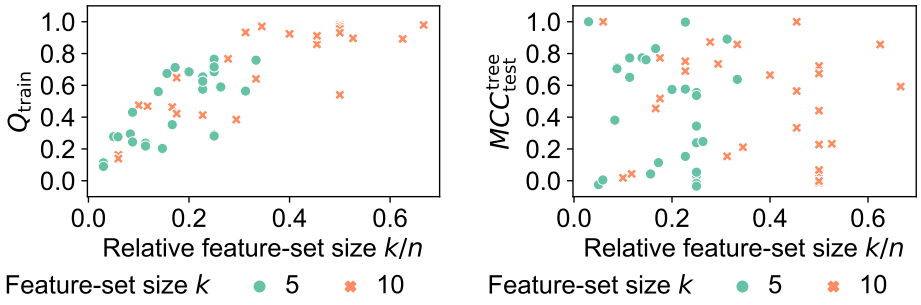
Limitations Like *Greedy Balancing Search* and *Greedy Replacement Search*, *Greedy Depth Search* assumes univariate feature qualities and may be worse than exact search. As a sequential procedure, it does not balance the alternatives' qualities. It may yield more alternatives than the former two heuristics but has a higher and more variable runtime.

A.7 Evaluation

In this section, we evaluate experimental results not covered in Section 6. In particular, we cover three experimental dimensions not stemming from the search for alternatives: datasets (cf. Section A.7.1), feature-set-quality metrics (cf. Section A.7.2), and feature-selection methods (cf. Section A.7.3).

A.7.1 Datasets

Naturally, feature-set quality depends on the dataset, and several effects could occur. For example, the distribution of feature-set quality in a dataset may be



(a) Training-set objective value.

(b) Test-set prediction performance.

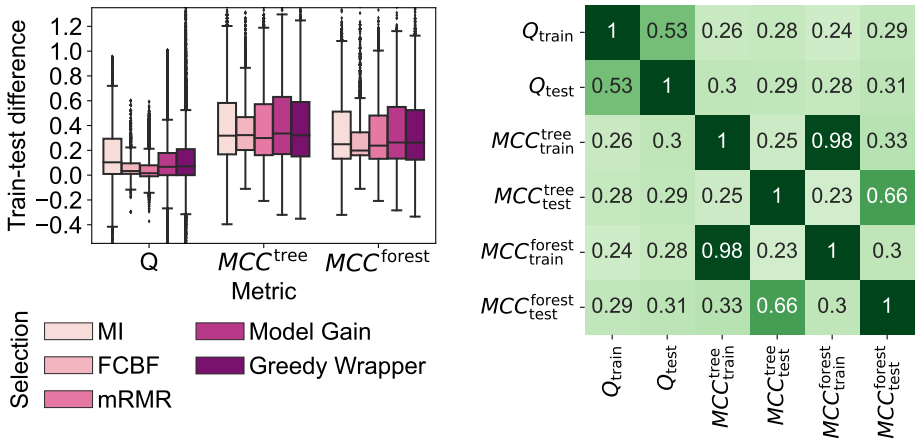
Figure 8: Feature-set quality in datasets over feature-set size k relative to dimensionality n, by feature-set size k and evaluation metric. Results from the original feature sets of sequential search with *MI* as feature-selection method.

relatively uniform or relatively skewed. Further, datasets with more features n give way to more alternative feature sets. At the same time, the feature quality can be spread over more features than for lower-dimensional datasets, making it harder to compose a small high-quality feature set. Indeed, our experiments show a broad variation of feature-set quality over the datasets. Figure 8 depicts the relationship between datasets and the quality of the original, i.e., unconstrained, feature set in sequential search. To account for the varying dataset dimensionality, we put the ratio between feature-set size k and dimensionality n on the x-axis, which measures relative feature-set sizes. As Figure 8a displays, the objective of the univariate feature-selection method *MI* approximately increases linearly with k/n. However, there still is variation exclusively caused by the dataset rather than its dimensionality. Further, the quality of a prediction model, i.e., decision trees, does not exhibit any trend but varies strongly between datasets, as Figure 8b visualizes. This variation justifies our normalization of feature-set quality when analyzing alternatives in Sections 6.2 and 6.3.

A.7.2 Feature-Set Quality Metrics

Prediction models and overfitting As one can expect, random forests have a higher average prediction performance than decision trees. Further, both model types exhibit overfitting, i.e., there is a gap between training-set and test-set performance. In particular, over all experimental settings, both model types have a mean training-set MCC around 0.85-0.86 (median: 1.0). In contrast, decision trees have a mean MCC of 0.47 (median: 0.53) on the test set, while random forests have a slightly higher mean MCC of 0.52 (median: 0.61). I.e., prediction performance is significantly worse on the test set than on the training set. The existence of overfitting makes sense as we do not regularize, i.e., limit the growth of the trees or prune them after training.

As another comparison, Figure 9a shows the distribution of the difference between training and test feature-set quality, again over all experimental set-



(a) Training-test difference in feature-set quality by feature-selection method. Y-axis is truncated to improve readability.

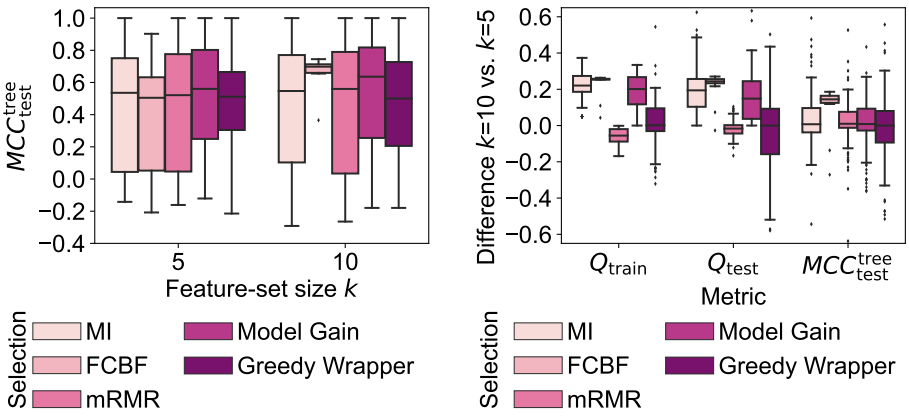
(b) Correlation between evaluation metrics, averaged over datasets, cross-validation folds, and feature-selection methods.

Figure 9: Feature-set quality by evaluation metric. Results from all search runs.

tings. Once more, we observe that training feature-set quality is usually higher, i.e., the difference shown in the figure is greater than zero. However, this phenomenon does not invalidate our analysis of how feature-set quality develops over alternatives. The optimization objective Q, which Figure 9a also depicts, shows overfitting for all feature-selection methods as well, though to a lesser extent than prediction performance. Thus, Section 6 considers the training and test set for the objective value, but only the test set for prediction performance.

Correlation between evaluation metrics Figure 9b shows the Spearman correlation between different evaluation metrics over all experimental settings: First, we compute the correlation between metrics for each combination of dataset, cross-validation fold, and feature-selection method. Second, we average the correlation values over these three experimental dimensions. This two-step procedure accounts for the different objectives of feature-selection methods and the normalization of quality per dataset and cross-validation fold in some objectives (cf. Section 5.3.2). The plot shows that the performance of decision trees and random forests is highly correlated. Thus, we only report MCC of decision trees in Section 6, which are the simpler model type and always consider all features during training rather than randomly sampling them.

Figure 9b also shows that the correlation between training and test feature-set quality is only moderate for the optimization objective Q and weak for prediction performance in terms of MCC. This result might be caused by overfitting, whose strength may depend on the experimental settings. Further, the correlation between optimization objective Q and prediction MCC is only weak to moderate as well. I.e., the objective of feature selection is only partially in-



(a) Test-set prediction performance.

(b) Difference in feature-set quality between $k = 10$ and $k = 5$ by evaluation metric. Y-axis is truncated to improve readability.

Figure 10: Feature-set quality by feature-selection method and feature-set size k. Results from the original feature sets of sequential search.

dicative of prediction performance since the former might use a simplified quality criterion. Among the five feature-selection methods, *Greedy Wrapper* has the highest correlation between training-set objective value and test-set MCC, with a value of 0.48. Since this feature-selection method uses prediction performance in its objective, a comparatively high correlation is expected. On the other end of the spectrum, *mRMR* exhibits a correlation of -0.05 between training-set objective value and test-set MCC. This filter method penalizes the correlation between features in its objective. However, redundant features may not hurt prediction performance in decision trees, even if they do not improve it.

A.7.3 Feature-Selection Methods

Prediction performance As the five feature-selection methods employ different objective functions Q, comparing absolute objective values between them does not make sense. However, we can analyze the prediction performance of the obtained feature sets. Figure 10a compares a decision tree's test-set MCC on the original feature sets of sequential search between feature-selection methods. On average, *Model Gain* is the best feature-selection method: The mean test-set MCC of decision trees is 0.53 for *Model Gain*, 0.49 for *Greedy Wrapper*, 0.47 for *MI*, 0.46 for *mRMR*, 0.43 for *FCBF*. In particular, the univariate, model-free method *MI* keeps up surprisingly well with more sophisticated methods. Thus, the analyses of alternative feature sets in Section 6 focus on *MI* while still discussing the remaining feature-selection methods. The overall best feature-selection method, *Model Gain*, uses the same objective function as *MI*

but obtains its feature qualities from a prediction model rather than a bivariate dependency measure, which might boost its performance.

While *Greedy Wrapper* uses actual prediction performance to assess feature-set quality, its heuristic nature might prevent better results: This method only evaluates a fraction of all feature sets, while the other feature-selection methods optimize globally. In particular, *Greedy Wrapper* performed 629 iterations on average (median: 561) to determine the original feature sets of sequential search. However, the number of possible feature sets is much higher, e.g., already $2^{15} = 32768$ for the lowest-dimensional datasets in our evaluation (cf. Table 2).

FCBF's results may be taken with a grain of salt: Over all experimental settings, 89% of feature sets for *FCBF* were infeasible, i.e., no solution satisfied the constraints. In contrast, this figure only is 18% for *MI*. Even the original feature set in sequential search is infeasible in 71% of the cases for *FCBF* but never for the other feature-selection methods. In particular, the combination of feature-correlation constraints in our formulation of *FCBF* (c.f. Equation 12) with a feature-set-cardinality constraint, i.e., enforcing a feature-set size k, may make the problem infeasible, especially if k gets larger.

Influence of feature-set size k As expected, larger feature sets usually exhibit a higher feature-set quality than smaller feature sets in our experiments. However, the increase in quality with k is not proportional, and there might even be a decrease. As Figure 10b shows for the original feature sets of sequential search, *MI* and *Model Gain* exhibit an increase of the training-set objective value Q_{train} from $k = 5$ to $k = 10$, i.e., the difference depicted in Figure 10b is positive. As these objectives are monotonic in the set of selected features, a decrease in the training-set objective value is impossible. In contrast, the heuristic *Greedy Wrapper* does not necessarily benefit from more features. The latter insight also applies to *mRMR*, which normalizes its objective with the number of selected features and penalizes feature redundancy. For *FBCF*, the fraction of feasible feature sets changes considerably from $k = 5$ to $k = 10$, so one cannot directly compare the overall quality between these two settings. As Figure 10b also displays, the benefit of larger feature sets is even less clear for prediction performance. In particular, all feature-selection methods except *FCBF* show a median difference in test-set MCC close to zero when comparing $k = 5$ to $k = 10$. Thus, Section 6 focuses on smaller feature sets, i.e., $k = 5$.

References

[1] Noga Alon et al. "Approximation schemes for scheduling on parallel machines". In: *J. Sched.* 1.1 (1998), pp. 55–66. DOI: 10.1002/(SICI)1099-1425(199806)1:1<55::AID-JOS2>3.0.CO;2-J.

[2] André Artelt and Barbara Hammer. *"Even if ..." – Diverse Semifactual Explanations of Reject*. arXiv:2207.01898 [cs.LG]. 2022. URL: https://arxiv.org/abs/2207.01898.

[3] Susan F. Assmann et al. “On a Dual Version of the One-Dimensional Bin Packing Problem”. In: *J. Algorithms* 5.4 (1984), pp. 502–525. DOI: 10.1016/0196-6774(84)90004-X.

[4] Luitpold Babel, Hans Kellerer, and Vladimir Kotov. “The k-Partitioning Problem”. In: *Math. Methods Oper. Res.* 47.1 (1998), pp. 59–82. DOI: 10.1007/BF01193837.

[5] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. “Maximum Satisfiability”. In: *Handbook of Satisfiability*. 2nd ed. IOS Press, 2021. Chap. 24, pp. 929–991. DOI: 10.3233/FAIA201008.

[6] Jakob Bach et al. “An Empirical Evaluation of Constrained Feature Selection”. In: *SN Comput. Sci.* 3.6 (2022), pp. 1–25. DOI: 10.1007/s42979-022-01338-z.

[7] Eric Bae and James Bailey. “COALA: A Novel Approach for the Extraction of an Alternate Clustering of High Quality and High Dissimilarity”. In: *Proc. ICDM*. Hong Kong, China, 2006, pp. 53–62. DOI: 10.1109/ICDM.2006.37.

[8] Eric Bae, James Bailey, and Guozhu Dong. “A clustering comparison measure using density profiles and its application to the discovery of alternate clusterings”. In: *Data Min. Knowl. Disc.* 21.3 (2010), pp. 427–471. DOI: 10.1007/s10618-009-0164-z.

[9] James Bailey. “Alternative Clustering Analysis: A Review”. In: *Data Clustering: Algorithms and Applications*. 1st ed. CRC Press, 2014. Chap. 21, pp. 535–550. DOI: 10.1201/9781315373515.

[10] Ksenia Bestuzheva et al. *The SCIP Optimization Suite 8.0*. Tech. rep. Zuse Institute Berlin, Germany, 2021. URL: http://nbn-resolving.de/urn:nbn:de:0297-zib-85309.

[11] Giorgos Borboudakis and Ioannis Tsamardinos. “Extending greedy feature selection algorithms to multiple solutions”. In: *Data Min. Knowl. Disc.* 35.4 (2021), pp. 1393–1434. DOI: 10.1007/s10618-020-00731-7.

[12] Leo Breiman. “Random Forests”. In: *Mach. Learn.* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[13] Leo Breiman et al. *Classification and Regression Trees*. 1st ed. Wadsworth, 1984. DOI: 10.1201/9781315139470.

[14] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. “Machine Learning Interpretability: A Survey on Methods and Metrics”. In: *Electronics* 8.8 (2019). DOI: 10.3390/electronics8080832.

[15] Girish Chandrashekar and Ferat Sahin. “A survey on feature selection methods”. In: *Comput. Electr. Eng.* 40.1 (2014), pp. 16–28. DOI: 10.1016/j.compeleceng.2013.11.024.

[16] Chandra Chekuri and Sanjeev Khanna. “A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem”. In: *SIAM J. Comput.* 35.3 (2005), pp. 713–728. DOI: 10.1137/S0097539700382820.

[17] Lin Chen et al. “An Efficient PTAS for Parallel Machine Scheduling with Capacity Constraints”. In: *Proc. COCOA*. Hong Kong, China, 2016, pp. 608–623. DOI: 10.1007/978-3-319-48749-6_44.

[18] Shi Ping Chen, Yong He, and Guohui Lin. "3-Partitioning Problems for Maximizing the Minimum Load". In: *J. Comb. Optim.* 6 (2002), pp. 67–80. DOI: 10.1023/A:1013370208101.

[19] Seung-Seok Choi, Sung-Hyuk Cha, and Charles C. Tappert. "A Survey of Binary Similarity and Distance Measures". In: *J. Syst. Cybern. Inf.* 8.1 (2010), pp. 43–48. URL: http://www.iiisci.org/Journal/pdv/sci/pdfs/GS315JG.pdf.

[20] Ian Covert, Scott M. Lundberg, and Su-In Lee. "Understanding Global Feature Contributions With Additive Importance Measures". In: *Proc. NeurIPS*. Virtual conference, 2020, pp. 17212–17223. URL: https://proceedings.neurips.cc/paper/2020/file/c7bf0b7c1a86d5eb3be2c722cf2cf746-Paper.pdf.

[21] Susanne Dandl et al. "Multi-Objective Counterfactual Explanations". In: *Proc. PPSN*. Leiden, The Netherlands, 2020, pp. 448–469. DOI: 10.1007/978-3-030-58112-1_31.

[22] Mauro Dell'Amico, Manuel Iori, and Silvano Martello. "Heuristic Algorithms and Scatter Search for the Cardinality Constrained $P||\mathrm{C}_{\max}$ Problem". In: *J. Heuristics* 10 (2004), pp. 169–204. DOI: 10.1023/B:HEUR.0000026266.07036.da.

[23] Mauro Dell'Amico and Silvano Martello. "Bounds for the cardinality constrained $P||\mathrm{C}_{\max}$ problem". In: *J. Sched.* 4.3 (2001), pp. 123–138. DOI: 10.1002/jos.68.

[24] Bryan L. Deuermeyer, Donald K. Friesen, and Michael A. Langston. "Scheduling to Maximize the Minimum Processor Finish Time in a Multiprocessor System". In: *SIAM J. Algebraic Discrete Methods* 3.2 (1982), pp. 190–196. DOI: 10.1137/0603019.

[25] Rodney G. Downey, Michael R. Fellows, and Ulrike Stege. "Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability". In: *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*. Štiřín Castle, Czech Republic, 1997, pp. 49–99. DOI: https://doi.org/10.1090/dimacs/049/04.

[26] Leo Egghe. "New Relations Between Similarity Measures for Vectors Based on Vector Norms". In: *J. Am. Soc. Inf. Sci. Technol.* 60.2 (2009), pp. 232–239. DOI: 10.1002/asi.20949.

[27] Christos Emmanouilidis et al. "Selecting Features in Neurofuzzy Modelling by Multiobjective Genetic Algorithms". In: *Proc. ICANN*. Edinburgh, United Kingdom, 1999, pp. 749–754. DOI: 10.1049/cp:19991201.

[28] Stefano Ermon, Carla Gomes, and Bart Selman. "Uniform Solution Sampling Using a Constraint Solver As an Oracle". In: *Proc. UAI*. Catalina Island, CA, USA, 2012, pp. 255–264. URL: https://www.auai.org/uai2012/papers/160.pdf.

[29] Edouard Fouché, Florian Kalinke, and Klemens Böhm. "Efficient subspace search in data streams". In: *Inf. Syst.* 97 (2021). DOI: 10.1016/j.is.2020.101705.

[30] Michael R. Garey and David S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. 24th ed. W. H. Freeman and Company, 2003. URL: https://www.worldcat.org/title/440655898.

[31] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. 2nd ed. Addison-Wesley, 1994. URL: https://www.worldcat.org/title/1085703509.

[32] William Christopher Groves. "Toward Automating and Systematizing the Use of Domain Knowledge in Feature Selection". PhD thesis. University of Minnesota, 2015. URL: https://hdl.handle.net/11299/175444.

[33] Riccardo Guidotti. "Counterfactual explanations and how to find them: literature review and benchmarking". In: *Data Min. Knowl. Disc.* (2022), pp. 1–55. DOI: 10.1007/s10618-022-00831-6.

[34] Jianmei Guo and Kai Shi. "To Preserve or Not to Preserve Invalid Solutions in Search-Based Software Engineering: A Case Study in Software Product Lines". In: *Pro. ICSE*. Gothenburg, Sweden, 2018, pp. 1027–1038. DOI: 10.1145/3180155.3180163.

[35] D. S. Guru et al. "An alternative framework for univariate filter based feature selection for text categorization". In: *Pattern Recognit. Lett.* 103 (2018), pp. 23–31. DOI: 10.1016/j.patrec.2017.12.025.

[36] Isabelle Guyon and André Elisseeff. "An Introduction to Variable and Feature Selection". In: *J. Mach. Learn. Res.* 3.Mar (2003), pp. 1157–1182. URL: https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf.

[37] Mark A. Hall. "Correlation-based Feature Selection for Machine Learning". PhD thesis. University of Waikato, Hamilton, New Zealand, 1999. URL: https://www.cs.waikato.ac.nz/~ml/publications/1999/99MH-Thesis.pdf.

[38] Mark A. Hall. *Correlation-based Feature Selection of Discrete and Numeric Class Machine Learning*. Tech. rep. University of Waikato, Hamilton, New Zealand, 2000. URL: https://hdl.handle.net/10289/1024.

[39] Mohamed Haouari and Mahdi Jemmali. "Maximizing the minimum completion time on parallel machines". In: *4OR* 6 (2008), pp. 375–392. DOI: 10.1007/s10288-007-0053-5.

[40] Yong He et al. "k-Partitioning Problems for Maximizing the Minimum Load". In: *Comput. Math. Appl.* 46.10-11 (2003), pp. 1671–1681. DOI: 10.1016/S0898-1221(03)90201-X.

[41] Christopher Henard et al. "Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines". In: *Proc. ICSE*. Florence, Italy, 2015, pp. 517–528. DOI: 10.1109/ICSE.2015.69.

[42] Juhua Hu and Jian Pei. "Subspace multi-clustering: a review". In: *Knowl. Inf. Sys.* 56.2 (2018), pp. 257–284. DOI: 10.1007/s10115-017-1110-9.

[43] Sarthak Jain and Byron C. Wallace. "Attention is not Explanation". In: *Proc. NAACL-HLT*. Minneapolis, MN, USA, 2019, pp. 3543–3556. DOI: 10.18653/v1/N19-1357.

[44] Amir-Hossein Karimi et al. "Model-Agnostic Counterfactual Explanations for Consequential Decisions". In: *Proc. AISTATS*. Virtual conference, 2020, pp. 895–905. URL: https://proceedings.mlr.press/v108/karimi20a.html.

[45] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. Plenum Press, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9.

[46] Hans Kellerer and Vladimir Kotov. "A 3/2-approximation algorithm for k_i-partitioning". In: *Oper. Res. Lett.* 39.5 (2011), pp. 359–362. DOI: 10.1016/j.orl.2011.06.005.

[47] Sanjeev Khanna, Madhu Sudan, and David P. Williamson. "A Complete Classification of the Approximability of Maximization Problems Derived from Boolean Constraint Satisfaction". In: *Proc. STOC*. El Paso, TX, USA, 1997, pp. 11–20. DOI: 10.1145/258533.258538.

[48] Sanjeev Khanna et al. "On Syntactic Versus Vomputational Views of Approximability". In: *SIAM J. Comput.* 28.1 (1998), pp. 164–191. DOI: 10.1137/S0097539795286612.

[49] Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. "Examples are not Enough, Learn to Criticize! Criticism for Interpretability". In: *Proc. NIPS*. Barcelona, Spain, 2016. URL: https://proceedings.neurips.cc/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf.

[50] Mi-Young Kim et al. "A Multi-Component Framework for the Analysis and Design of Explainable Artificial Intelligence". In: *Mach. Learn. Knowl. Extr.* 3.4 (2021), pp. 900–921. DOI: 10.3390/make3040045.

[51] Kenji Kira and Larry A. Rendell. "The Feature Selection Problem: Traditional Methods and a New Algorithm". In: *Proc. ML*. Aberdeen, Scotland, UK, 1992, pp. 129–134. URL: https://www.aaai.org/Papers/AAAI/1992/AAAI92-020.pdf.

[52] Ron Kohavi and George H. John. "Wrappers for feature subset selection". In: *Artif. Intell.* 97.1-2 (1997), pp. 273–324. DOI: 10.1016/S0004-3702(97)00043-X.

[53] Richard E. Korf. "Multi-Way Number Partitioning". In: *Proc. IJCAI*. Pasadena, CA, USA, 2009, pp. 538–543. URL: https://www.ijcai.org/Proceedings/09/Papers/096.pdf.

[54] Richard E. Korf. "Objective Functions for Multi-Way Number Partitioning". In: *Proc. SoCS*. Atlanta, GA, USA, 2010, pp. 71–72. DOI: https://doi.org/10.1609/socs.v1i1.18172.

[55] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information". In: *Phys. Rev. E* 69.6 (2004). DOI: 10.1103/PhysRevE.69.066138.

[56] Vincenzo Lagani et al. "Feature Selection with the R Package MXM: Discovering Statistically Equivalent Feature Subsets". In: *J. Stat. Software* 80.7 (2017), pp. 1–25. DOI: 10.18637/jss.v080.i07.

[57] Alexander Lawrinenko. "Identical Parallel Machine Scheduling Problems: Structural patterns, bounding techniques and solution procedures". PhD thesis. Friedrich-Schiller-Universität Jena, 2017. URL: https://nbn-resolving.org/urn:nbn:de:gbv:27-dbt-20170427-0956483.

[58] Alexander Lawrinenko, Stefan Schwerdfeger, and Rico Walter. "Reduction criteria, upper bounds, and a dynamic programming based heuristic for the max–min k_i-partitioning problem". In: *J. Heuristics* 24 (2018), pp. 173–203. DOI: 10.1007/s10732-017-9362-9.

[59] Matthijs van Leeuwen and Arno Knobbe. "Diverse subgroup set discovery". In: *Data Min. Knowl. Disc.* 25.2 (2012), pp. 208–242. DOI: 10.1007/s10618-012-0273-y.

[60] Chu Min Li and Felip Manya. "MaxSAT, Hard and Soft Constraints". In: *Handbook of Satisfiability*. 2nd ed. IOS Press, 2021. Chap. 23, pp. 903–927. DOI: 10.3233/FAIA201007.

[61] Jundong Li et al. "Feature Selection: A Data Perspective". In: *ACM Comput. Surv.* 50.6 (2017), pp. 1–45. DOI: 10.1145/3136625.

[62] Kai Liu and Jin Tian. "Subspace Learning with an Archive-Based Genetic Algorithm". In: *Proc. IEEM*. Bangkok, Thailand, 2018, pp. 181–188. DOI: 10.1007/978-981-13-3402-3_20.

[63] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Proc. NIPS*. Long Beach, CA, USA, 2017. URL: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[64] Brian W. Matthews. "Comparison of the predicted and observed secondary structure of T4 phage lysozyme". In: *Biochim. Biophys. Acta - Protein Struct.* 405.2 (1975), pp. 442–451. DOI: 10.1016/0005-2795(75)90109-9.

[65] Wil Michiels et al. "Computer-assisted proof of performance ratios for the Differencing Method". In: *Discrete Optim.* 9.1 (2012), pp. 1–16. DOI: 10.1016/j.disopt.2011.10.001.

[66] Matthias Mnich and René van Bevern. "Parameterized complexity of machine scheduling: 15 open problems". In: *Comput. Oper. Res.* 100 (2018), pp. 254–261. DOI: 10.1016/j.cor.2018.07.020.

[67] Kiarash Mohammadi et al. "Scaling Guarantees for Nearest Counterfactual Explanations". In: *Proc. AIES*. Virtual conference, 2021, pp. 177–187. DOI: 10.1145/3461702.3462514.

[68] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. "Interpretable Machine Learning – A Brief History, State-of-the-Art and Challenges". In: *Proc. XKDD*. Ghent, Belgium, 2020, pp. 417–431. DOI: 10.1007/978-3-030-65965-3_28.

[69] MOSEK ApS. *MOSEK Modeling Cookbook : Mixed integer optimzation*. Accessed: 2022-10-18. 2022. URL: https://docs.mosek.com/modeling-cookbook/mio.html.

[70] L. Moser and M. Wyman. "Stirling numbers of the second kind". In: *Duke Math. J.* 25.1 (1958), pp. 29–43. DOI: 10.1215/S0012-7094-58-02504-3.

[71] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. "Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations". In: *Proc. FAT**. Barcelona, Spain, 2020, pp. 607–617. DOI: 10.1145/3351095.3372850.

[72] Emmanuel Müller et al. "Relevant Subspace Clustering: Mining the Most Interesting Non-Redundant Concepts in High Dimensional Data". In: *Proc. ICDM*. Miami Beach, FL, USA, 2009, pp. 377–386. DOI: 10.1109/ICDM.2009.10.

[73] Inga M. Müller. "Feature selection for energy system modeling: Identification of relevant time series information". In: *Energy AI* 4 (2021). DOI: 10.1016/j.egyai.2021.100057.

[74] Roger B. Myerson. "Utilitarianism, Egalitarianism, and the Timing Effect in Social Choice Problems". In: *Econometrica* 49.4 (1981), pp. 883–897. DOI: 10.2307/1912508.

[75] Nina Narodytska et al. "Learning Optimal Decision Trees with SAT". In: *Proc. IJCAI*. Stockholm, Sweden, 2018, pp. 1362–1368. DOI: 10.24963/ijcai.2018/189.

[76] Hai Nguyen, Katrin Franke, and Slobodan Petrović. "Optimizing a Class of Feature Selection Measures". In: *Proc. DISCML*. Vancouver, BC, Canada, 2009. URL: https://www.researchgate.net/publication/231175763.

[77] Hai Thanh Nguyen, Katrin Franke, and Slobodan Petrović. "Improving Effectiveness of Intrusion Detection by Correlation Feature Selection". In: *Proc. ARES*. Krakow, Poland, 2010, pp. 17–24. DOI: 10.1109/ARES.2010.70.

[78] Hai Thanh Nguyen, Katrin Franke, and Slobodan Petrović. "Towards a Generic Feature-Selection Measure for Intrusion Detection". In: *Proc. ICPR*. Istanbul, Turkey, 2010, pp. 1529–1532. DOI: 10.1109/ICPR.2010.378.

[79] Hoang Vu Nguyen, Emmanuel Müller, and Klemens Böhm. "4S: Scalable Subspace Search Scheme Overcoming Traditional Apriori Processing". In: *Proc. Big Data*. Santa Clara, CA, USA, 2013, pp. 359–367. DOI: 10.1109/BigData.2013.6691596.

[80] Xuan Vinh Nguyen et al. "Effective Global Approaches for Mutual Information Based Feature Selection". In: *Proc. KDD*. New York, NY, USA, 2014, pp. 512–521. DOI: 10.1145/2623330.2623611.

[81] Uchechukwu F. Njoku et al. "Wrapper Methods for Multi-Objective Feature Selection". In: *Proc. EDBT*. Ioannina, Greece, 2023, pp. 697–709. DOI: 10.48786/edbt.2023.58.

[82] Randal S. Olson et al. "PMLB: a large benchmark suite for machine learning evaluation and comparison". In: *Biodata Min.* 10 (2017). DOI: 10.1186/s13040-017-0154-4.

[83] Pavel Paclík et al. "On Feature Selection with Measurement Cost and Grouped Features". In: *Proc. SSPR /SPR*. Windsor, ON, Canada, 2002, pp. 461–469. DOI: 10.1007/3-540-70659-3_48.

[84] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *J. Mach. Learn. Res.* 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[85] Hanchuan Peng, Fuhui Long, and Chris Ding. "Feature Selection Based on Mutual Information Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27.8 (2005), pp. 1226–1238. DOI: 10.1109/TPAMI.2005.159.

[86] Laurent Perron and Vincent Furnon. *OR-Tools*. Accessed: 2022-10-18. Google, 2022. URL: https://developers.google.com/optimization/.

[87] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?" Explaining the Predictions of Any Classifier". In: *Proc. KDD*. San Francisco, CA, USA, 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778.

[88] Marko Robnik-Šikonja and Igor Kononenko. "An adaptation of Relief for attribute estimation in regression". In: *Proc. ICML*. Nashville, TN, USA, 1997, pp. 296–304. URL: https://www.researchgate.net/publication/2635627.

[89] Irene Rodriguez-Lujan et al. “Quadratic Programming Feature Selection”. In: *J. Mach. Learn. Res.* 11.49 (2010), pp. 1491–1516. URL: http://jmlr.org/papers/v11/rodriguez-lujan10a.html.

[90] Joseph D. Romano et al. *PMLB v1.0: An open source dataset collection for benchmarking machine learning methods*. arXiv:2012.00058v3 [cs.LG]. 2021. URL: https://arxiv.org/abs/2012.00058v3.

[91] Chris Russell. “Efficient Search for Diverse Coherent Explanations”. In: *Pro. FAT**. Atlanta, GA, USA, 2019, pp. 20–28. DOI: 10.1145/3287560.3287569.

[92] Yvan Saeys, Thomas Abeel, and Yves Van de Peer. “Robust Feature Selection Using Ensemble Feature Selection Techniques”. In: *Proc. ECML PKDD*. Antwerp, Belgium, 2008, pp. 313–325. DOI: 10.1007/978-3-540-87481-2_21.

[93] Sartaj K. Sahni. “Algorithms for Scheduling Independent Tasks”. In: *J. ACM* 23.1 (1976), pp. 116–127. DOI: 10.1145/321921.321934.

[94] André Schidler and Stefan Szeider. “SAT-based Decision Tree Learning for Large Data Sets”. In: *Proc. AAAI*. Virtual conference, 2021, pp. 3904–3912. DOI: 10.1609/aaai.v35i5.16509.

[95] Ethan L. Schreiber, Richard E. Korf, and Michael D. Moffitt. “Optimal Multi-Way Number Partitioning”. In: *J. ACM* 65.4 (2018), pp. 1–61. DOI: 10.1145/3184400.

[96] Borja Seijo-Pardo et al. “Ensemble feature selection: Homogeneous and heterogeneous approaches”. In: *Knowl.-Based Syst.* 118 (2017), pp. 124–139. DOI: 10.1016/j.knosys.2016.11.017.

[97] Umair F. Siddiqi, Sadiq M. Sait, and Okyay Kaynak. “Genetic Algorithm for the Mutual Information-Based Feature Selection in Univariate Time Series Data”. In: *IEEE Access* 8 (2020), pp. 9597–9609. DOI: 10.1109/ACCESS.2020.2964803.

[98] Wilson Silva, Kelwin Fernandes, and Jaime S. Cardoso. “How to produce complementary explanations using an Ensemble model”. In: *Proc. IJCNN*. Budapest, Hungary, 2019. DOI: 10.1109/IJCNN.2019.8852409.

[99] Carsten Sinz. “Towards an Optimal CNF Encoding of Boolean Cardinality Constraints”. In: *Proc. CP*. Sitges, Spain, 2005, pp. 827–831. DOI: 10.1007/11564751_73.

[100] Ilia Stepin et al. “A Survey of Contrastive and Counterfactual Explanation Generation Methods for Explainable Artificial Intelligence”. In: *IEEE Access* 9 (2021), pp. 11974–12001. DOI: 10.1109/ACCESS.2021.3051315.

[101] Vinh Thanh Tao and JongHyeok Lee. “A Novel Approach for Finding Alternative Clusterings Using Feature Selection”. In: *Proc. DASFAA*. Busan, South Korea, 2012, pp. 482–493. DOI: 10.1007/978-3-642-29038-1_35.

[102] Holger Trittenbach and Klemens Böhm. “Dimension-based subspace search for outlier detection”. In: *Int. J. Data Sci. Anal.* 7.2 (2019), pp. 87–101. DOI: 10.1007/s41060-018-0137-7.

[103] Felix Ulrich-Oltean, Peter Nightingale, and James Alfred Walker. “Selecting SAT Encodings for Pseudo-Boolean and Linear Integer Constraints”. In: *Proc. CP*. Haifa, Israel, 2022, 38:1–38:17. DOI: 10.4230/LIPIcs.CP.2022.38.

[104] Sahil Verma, John Dickerson, and Keegan Hines. *Counterfactual Explanations for Machine Learning: A Review*. arXiv:2010.10596 [cs.LG]. 2020. URL: https://arxiv.org/abs/2010.10596.

[105] Sandra Wachter, Brent Mittelstadt, and Chris Russell. "Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR". In: *Harv. J. Law Technol.* 31.2 (2017), pp. 841–887. URL: https://jolt.law.harvard.edu/assets/articlePDFs/v31/Counterfactual-Explanations-without-Opening-the-Black-Box-Sandra-Wachter-et-al.pdf.

[106] Rico Walter and Alexander Lawrinenko. "Lower bounds and algorithms for the minimum cardinality bin covering problem". In: *Eur. J. Oper. Res.* 256.2 (2017), pp. 392–403. DOI: 10.1016/j.ejor.2016.06.068.

[107] Rico Walter, Martin Wirth, and Alexander Lawrinenko. "Improved approaches to the exact solution of the machine covering problem". In: *J. Sched.* 20 (2017), pp. 147–164. DOI: 10.1007/s10951-016-0477-x.

[108] Danding Wang et al. "Designing Theory-Driven User-Centric Explainable AI". In: *Proc. CHI*. Glasgow, UK, 2019. DOI: 10.1145/3290605.3300831.

[109] Jules White et al. "Automated diagnosis of feature model configurations". In: *J. Syst. Software* 83.7 (2010), pp. 1094–1107. DOI: 10.1016/j.jss.2010.02.017.

[110] Wallace Alvin Wilson. "On Semi-Metric Spaces". In: *Am. J. Math.* 53.2 (1931), pp. 361–373. DOI: 10.2307/2370790.

[111] Gerhard J. Woeginger. "A comment on scheduling two parallel machines with capacity constraints". In: *Discrete Optim.* 2.3 (2005), pp. 269–272. DOI: 10.1016/j.disopt.2005.06.005.

[112] Gerhard J. Woeginger. "A polynomial-time approximation scheme for maximizing the minimum machine completion time". In: *Oper. Res. Lett.* 20.4 (1997), pp. 149–154. DOI: 10.1016/S0167-6377(96)00055-7.

[113] Adam Woznica, Phong Nguyen, and Alexandros Kalousis. "Model Mining for Robust Feature Selection". In: *Proc. KDD*. Beijing, China, 2012, pp. 913–921. DOI: 10.1145/2339530.2339674.

[114] Jinqiang Yu et al. "Learning Optimal Decision Sets and Lists with SAT". In: *J. Artif. Intell. Res.* 72 (2021), pp. 1251–1279. DOI: 10.1613/jair.1.12719.

[115] Lei Yu and Huan Liu. "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution". In: *Proc. ICML*. Washington DC, USA, 2003, pp. 856–863. URL: https://aaai.org/Papers/ICML/2003/ICML03-111.pdf.

[116] Ming Yuan and Yi Lin. "Model selection and estimation in regression with grouped variables". In: *J. R. Stat. Soc.: Ser. B (Stat. Methodol.)* 68.1 (2006), pp. 49–67. DOI: 10.1111/j.1467-9868.2005.00532.x.

[117] Jilian Zhang, Kyriakos Mouratidis, and HweeHwa Pang. "Heuristic Algorithms for Balanced Multi-Way Number Partitioning". In: *Proc. IJCAI*. Barcelona, Spain, 2011, pp. 693–698. DOI: 10.5591/978-1-57735-516-8/IJCAI11-122.