International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-024-00527-8

REGULAR PAPER

Alternative feature selection with user control

Jakob Bach[1] · **Klemens Böhm**[1]

Received: 26 September 2023 / Accepted: 19 February 2024

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 does not suffice in certain 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 in comprehensive experiments with 30 datasets representing binary-classification problems. We observe that alternative feature sets may indeed have high prediction quality, and we analyze 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 the dimensionality of datasets, they lower the computational cost and the memory requirements of prediction models. Prediction models also tend to generalize better without irrelevant and spurious predictors. While some model types can implicitly select relevant features, others cannot. In addition, simpler prediction models improve interpretability [42].

Most conventional feature-selection methods return one feature set only [6]. 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 [34, 78].

For example, in a dataset describing physical experiments, feature selection may help to discover relationships between physical quantities. In particular, highly predictive feature sets indicate which input quantities are strongly related to the output quantity. Domain experts may use these feature sets to formulate hypotheses on physical laws. However, if multiple alternative sets of similar quality exist, further analyses and experiments may be necessary to reveal the true underlying physical mechanism. Only knowing one predictive feature set and using it as the only explanation is misleading in such a situation.

Problem statement
This article[1] 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 Sect. 3.2. This problem entails an interesting trade-off: Depending on how different the alternatives should be, one may have to compromise on quality. In particular, a stronger dissimilarity requirement may require selecting more low-quality features

✉ Jakob Bach
jakob.bach@kit.edu

Klemens Böhm
klemens.boehm@kit.edu

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

[1] This article is a polished and shortened version of a preprint available on *arXiv* [3]. Experimental results are the same.

Published online: 26 March 2024

in the alternatives. Users should be able to control the dissimilarity and hence this trade-off.

Related work

In the field of feature selection, only a few methods yield multiple, diverse feature sets [6]. In particular, there are techniques for ensemble feature selection [69, 71] and statistically equivalent feature subsets [39]. These approaches do not guarantee the diversity of the feature sets, nor do they let users control diversity. Obtaining multiple, diverse solutions also is an objective in other fields, e.g., for clustering [4, 29, 51], subgroup discovery [76], subspace search [74], and explainable AI [1, 33, 68]. 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 Sect. 6.

Contributions

Our contribution is fourfold.

(1) 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. As a wide variety of such methods exists [10, 42], users can choose a method according to their needs/preferences. This approach also allows integrating other constraints on feature sets, e.g., to capture domain knowledge [2, 21]. Finally, this approach gives users control over the search for alternatives with two parameters, i.e., the number of alternatives and a dissimilarity threshold.
(2) We analyze the computational complexity of this problem. We show $\mathcal{NP}$-hardness, for a simple notion of feature-set quality already. On the other hand, the runtime is polynomial if the sizes of the feature sets and the number of alternatives are fixed.
(3) We study how to solve this optimization problem. Specifically, we describe how to integrate conventional feature-selection methods into the objective function. The resulting optimization problems are either white-box, which one can tackle with an integer-programming optimizer, or black-box, for which we propose a constraint-aware heuristic search procedure.
(4) We evaluate alternative feature selection with comprehensive experiments. We use 30 datasets representing binary-classification problems from the Penn Machine Learning Benchmarks (PMLB) [60, 67] and five feature-selection methods. We focus on the feature-set quality of alternatives relative to the user parameters. We publish all our code[2] and experimental data[3] online.

Experimental results

We observe that several factors influence the quality of alternatives, namely the dataset, the feature-selection method, the notion of feature-set quality, and user parameters controlling the search for alternatives. We have come up with several observations and recommendations, such as: (a) Quality-wise, the prediction performance of feature sets may not correlate with the quality assigned by feature-selection methods. In particular, seemingly bad alternatives regarding the latter might still be good regarding the former. (b) As feature-set quality tends to decrease with an increasing number of alternatives and with an increasing dissimilarity threshold, these parameters let users indeed control the trade-off between the diversity of feature sets and their quality. If the parameter values are too strict, no valid alternatives might even exist. (c) Computationally, a sequential search for alternatives has turned out to be significantly faster than searching for multiple alternatives simultaneously while yielding a similar quality.

Outline

Section 2 introduces notation and fundamentals. Section 3 describes and analyzes alternative feature selection. Section 4 outlines our experimental design. Section 5 presents the experimental results. Section 6 reviews related work. Section 7 concludes. Appendix A contains technical details and some of the proofs.

2 Fundamentals

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

2.1 Notation

$X \in \mathbb{R}^{m \times n}$ is a dataset represented as a matrix. Each row is a data object, and each column is a feature. $\tilde{F} = \{f_1, \ldots, f_n\}$ is the set of feature names. We assume that categorical features have been encoded numerically, 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. Feature selection 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 $F_s = \{f_j \mid s_j = 1\} \subseteq \tilde{F}$. To simplify notation, we drop the subscript s in definitions where we do not explicitly refer to the value of s but only the set F. The function $Q(s, X, y)$ denotes the feature-set quality, which, without loss of generality, should be maximized.

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

[3] https://doi.org/10.35097/1975.

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 [10, 42, 59] for comprehensive studies and surveys of feature selection. Also, note that we assume a supervised feature-selection scenario, i.e., feature-set quality depending on a prediction target y. In principle, our definitions of alternatives also apply to an unsupervised scenario. Since the prediction target only appears in the function $Q(s, X, y)$, one could replace $Q(s, X, y)$ with $Q(s, X)$, i.e., an unsupervised notion of quality.

A conventional categorization of feature-selection methods distinguishes between filter, wrapper, and embedded methods [24]. *Filter methods* evaluate feature-set quality without prediction models. Univariate filters assess each feature independently, e.g., using the correlation of the feature with the prediction target. Multivariate filters, like CFS [25, 26], FCBF [81], and mRMR [63], assess feature sets as a whole, also considering interactions like redundancies between features. *Wrapper methods* [35] search over feature sets with a black-box optimization strategy, e.g., genetic algorithms, and assess feature-set quality with prediction models trained on candidate feature sets. *Embedded methods* train prediction models with built-in feature selection, e.g., decision trees [7] or random forests [8]. Thus, the criterion for feature-set quality is model-specific.

Besides conventional feature selection, *post hoc feature importance methods* play a major role in the field of machine-learning interpretability [9, 48]. These methods assess feature importance after training a prediction model and can be instance-specific, like LIME [65] or SHAP [44], or describe the entire dataset, like permutation importance [8] or SAGE [12].

3 Alternative feature selection

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

3.1 Optimization problem

Alternative feature selection has two goals. First, the quality of alternatives should be high. Second, alternative feature sets should differ from each other, i.e., contain different features. There are different ways to combine two goals in an optimization problem: First, one can consider both goals as objectives in a multi-objective problem. Second, one can treat one goal as the objective and enforce the other with constraints. Third, one can define constraints for both goals, searching for feasible solutions instead of optimizing. We stick to the second formulation and maximize feature-set quality subject to feature sets being alternative. This formulation allows us to keep the original objective function of feature selection. Thus, users do not need to specify a threshold on feature-set quality but control how alternative the feature sets must be instead. We obtain the following optimization problem (Eq. 1) to find 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 limit the feature-set size $|F_s|$ to a user-defined value $k \in \mathbb{N}$, which adds further, simple constraints.

3.2 Constraints—defining alternatives

We now formalize alternative feature sets via constraints, considering single (Sect. 3.2.1) and multiple alternatives (Sect. 3.2.2). All our definitions are independent of the feature-selection method.

3.2.1 Single alternative

We consider a feature set an alternative to another set if it differs sufficiently. We express this notion with a set-dissimilarity measure [11, 15]. These measures typically quantify how strongly two sets F', F'' overlap and relate this to their sizes. For instance, a well-known set-dissimilarity measure is the Jaccard distance, which is defined as follows (Eq. 2) for the feature sets F' and F'':

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

In this article, we use a dissimilarity measure based on the Dice coefficient, as defined by Eq. (3):

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

Other measures are also possible. Our definitions of alternatives only assume non-negativity, i.e., $d(F', F'') \geq 0$, and symmetry, i.e., $d(F', F'') = d(F'', F')$. We define single alternatives as follows:

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 dissimilar alternative feature sets should be. The larger τ, the more feature-set quality may drop. However, depending on the dataset, the same value of τ may have a stronger or weaker effect on feature-set quality. For example, certain datasets may encompass numerous features with similar levels of utility, thereby allowing for numerous alternatives of comparable quality. In contrast, predictions on other datasets may hinge on a few key features. Further, 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 user parameter. For normalized dissimilarity measures as in Eqs. 2 and 3, the interpretation of τ is user-friendly: Setting $\tau = 0$ allows identical alternatives, while $\tau = 1$ forbids any overlap. Users can also experiment with various values, such as performing a binary search across the range of τ.

When implementing Definition 1, the following proposition gives way to using a broad range of solvers to tackle the related optimization problem:

Proposition 1 (Linearity of constraints) *Using the Dice dissimilarity (Eq. 3), one can express alternative feature sets (Definition 1) with 0–1 integer linear constraints.*

Proof We re-arrange terms in the Dice dissimilarity (Eq. 3) to get rid of the quotient of feature-set sizes, obtaining Eq. (4):

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

Next, we express set sizes in terms of the feature-selection vector s, obtaining Eq. (5):

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

Finally, we replace each product $s'_j \cdot s''_j$ with an auxiliary variable t_j, bound by additional constraints, to linearize it [49], obtaining Eq. (6):

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

Only sums of variables and products with constants remain, so the constraints are linear. Further, if one feature set is known, i.e., either s' or s'' is fixed to particular values, Eq. (5) already is linear without Eq. (6). □

As an alternative formulation, one could also encode the 0–1 integer linear constraints into propositional logic (SAT) [75].

If the set sizes $|F'|$ and $|F''|$ are constant, e.g., user-defined, Eq. (4) implies that the threshold τ exhibits a linear relationship with the maximum number of overlapping features $|F' \cap F''|$. This correspondence simplifies the interpretation of τ and encourages us to utilize the Dice dissimilarity in the subsequent steps. On the contrary, the Jaccard distance exhibits a nonlinear relationship between τ and the overlap size. This insight arises from rearranging Eq. (2) in conjunction with Definition 1, obtaining Eq. (7):

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

Further, if $|F'| = |F''|$, Eq. (4) implies that parameter τ represents the share of features in one set that must not be a member of the other one, and vice versa, as expressed by Eq. (8):

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

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

3.2.2 Multiple alternatives

One can search for multiple alternatives either sequentially or simultaneously. The number of alternatives $a \in \mathbb{N}_0$ is a parameter for the users. The overall number of feature sets is $a + 1$ since we deem one feature set the 'original' one.

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

	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 compares the sizes of the optimization problems for these two search methods.

Sequential alternatives

With sequential search, users obtain alternatives iteratively, with one alternative per iteration. Thus, users can monitor the quality of alternatives and end the search after each iteration. This option frees users from setting the parameter a a priori. Each feature set has to be an alternative to all previously found ones, which are contained within 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 (Definition 1) to each $F' \in \mathbb{F}$.

Adapting Eq. (1) to Definition 2, we obtain the following optimization problem (Eq. 9) 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, i.e., optimizes only one feature set F_s per iteration. In particular, with $\mathbb{F} = \emptyset$ in the first iteration, we optimize for the 'original' feature set, which is the same as in conventional feature selection without constraints for alternatives. Thus, the number of variables in each optimization is independent of the number of alternatives a, and no linearization variables (Eq. 6) are required. Each alternative only adds one new constraint to the problem. Thus, we expect the runtime of sequential search to scale well with the number of alternatives. Additional runtime improvements can be achieved if the optimizer retains a state between iterations and can warm-start.

Simultaneous alternatives

Through simultaneous search, users can acquire multiple alternatives simultaneously, whose number a must be set beforehand. Once more, we employ pairwise dissimilarity constraints for these alternatives:

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 (Definition 1) to each other set $F'' \in \mathbb{F}$, $F' \neq F''$.

Adapting Eq. (1) to Definition 3, we obtain the following optimization problem (Eq. 10):

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

In contrast to the sequential case (Eq. 9), the problem requires a modified objective function and $a+1$ instead of one decision vector s. The operator $\text{agg}(\cdot)$ defines how to aggregate the qualities of the feature sets. In our experiments, we consider the sum as well as the minimum to instantiate $\text{agg}(\cdot)$, which we refer to as *sum-aggregation* and *min-aggregation*. The latter explicitly fosters balanced feature-set qualities. Appendix 1 discusses these two aggregation operators and additional ideas for balancing qualities in detail.

In terms of runtime, we anticipate simultaneous search to scale worse with the number of alternatives a than sequential search. This is because simultaneous search addresses a single large optimization problem, as opposed to several smaller ones. The number of decision variables grows linearly with a, while the number of constraints increases quadratically. Further, auxiliary variables are necessary to obtain linear constraints (Eq. 6).

Quality-wise, simultaneous search may benefit from optimizing once rather than greedily. Furthermore, the quality may be distributed more evenly across the alternatives compared to sequential search. However, a drawback is that there are no intermediate steps where users can interrupt the search after assessing the current feature-set quality.

3.3 Objective functions—finding alternatives

In this section, we delve into the process of identifying alternative feature sets. Specifically, we describe how to solve the optimization problem introduced in Sect. 3.1 for the different categories of feature-set quality measures featured in Sect. 2.2. We distinguish between white box (Sect. 3.3.1), black box (Sect. 3.3.2), and embedding (Sect. 3.3.3) approaches.

3.3.1 White-box optimization

If the feature-set quality function $Q(s, X, y)$ is sufficiently simple, one can find alternatives with a solver for white-box optimization problems. We already showed that our notion of alternative feature sets can be expressed as 0–1 integer linear constraints (Proposition 1). We now discuss four popular feature-selection methods with objectives that admit formulating a 0–1 integer linear problem. We evaluate all four methods in our experiments later.

Univariate filter feature selection

For univariate filter methods, the objective function is linear by default since the quality of a feature set is the sum of the qualities of the individual features, as defined by Eq. (11):

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

Here, $q(\cdot)$ is a bivariate dependency measure to quantify the relationship between each feature and the prediction target, e.g., mutual information [37] or the absolute value of Pearson correlation.

For this objective, Appendix A.2 specifies the complete optimization problem, including the constraints for alternatives we already introduced in Sect. 3.2.

Post hoc feature importance

One can also insert post hoc feature-importance scores as univariate feature qualities $q(X_{\cdot j}, y)$ into Eq. (11). However, such post hoc importance scores typically evaluate the quality of each feature in the presence of other features. For example, a feature may only be important in subsets where another feature is present, due to feature interaction, but unimportant otherwise, and a post hoc importance method like SHAP [44] may reflect both these aspects. In contrast, Eq. (11) implicitly assumes feature independence and cannot adapt importance scores depending on whether other features are selected. Thus, treating pre-computed post hoc importance scores as univariate feature qualities in the optimization objective can serve as a heuristic but may not faithfully represent the actual feature qualities in a particular selected set.

FCBF

The Fast Correlation-Based Filter (FCBF) [81] is a multivariate filter method based on the notion of predominance: The correlation of each selected feature with the prediction target must exceed a user-defined threshold and must also be higher than the correlation of each other selected feature with the given one. While FCBF originally employs a heuristic search to identify predominant features, we propose a formulation as a constrained optimization problem (Eq. 12) to allow a white-box optimization for alternatives:

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

We drop the original FCBF's threshold on feature-target correlation and maximize the latter instead, as in the univariate-filter case. Further, we keep the constraints of FCBF on feature–feature correlation. Specifically, we prohibit the concurrent selection of two features if their correlation is at least as high as the correlation of either feature with the target variable. Since the condition $q(X_{\cdot j_1}, y) \leq q(X_{\cdot j_2}, X_{\cdot j_1})$ in Eq. (12) does not depend on the decision variables s, one can pre-compute whether it holds before formulating the optimization problem and add the corresponding linear constraint $s_{j_1} + s_{j_2} \leq 1$ only for feature pairs where it is needed.

mRMR

The multivariate filter method Minimal Redundancy Maximum Relevance (mRMR) [63] combines two criteria, relevance and redundancy. Relevance is the dependency between selected features and the prediction target, as for univariate filter methods. Redundancy, in turn, quantifies the dependency between selected features. Both terms are averaged over the selected features. The objective is to maximize the difference between relevance and redundancy, as defined by Eq. (13):

$$\begin{aligned} \max_{s} \quad Q_{\text{mRMR}}(s, X, y) = & \frac{\sum_{j=1}^{n} q(X_{\cdot j}, y) \cdot s_j}{\sum_{j=1}^{n} s_j} \\ & - \frac{\sum_{j_1=1}^{n} \sum_{j_2=1}^{n} q(X_{\cdot j_1}, X_{\cdot j_2}) \cdot s_{j_1} \cdot s_{j_2}}{(\sum_{j=1}^{n} s_j)^2} \end{aligned} \tag{13}$$

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 [55, 66]. If one additionally replaces each product terms $s_{j_1} \cdot s_{j_2}$ according to Eq. (6), the problem becomes linear. However, there is a more efficient linearization [56, 57], which we use in our experiments, displayed in Eq. (14):

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

Here, A_{j_1} is the sum of all redundancy terms related to the feature with index j_1, i.e., the summed dependency value between this feature and all other selected features. 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 otherwise ($s_j = 0$). To this end, M is a large positive value that deactivates the constraint $z_j \geq A_j$ if $s_j = 0$.

Since Eq. (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 [56, 57]. Additionally, in accordance with [55], we assign a value of zero to the self-redundancy terms $q(X_{\cdot j}, X_{\cdot j})$, effectively excluding them from the objective function. Thus, the redundancy term uses $k \cdot (k-1)$ instead of k^2 for averaging.

3.3.2 Black-box optimization

Overview of approaches

If feature-set quality does not have an expression suitable for white-box optimization, 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. There are several ways to consider the constraints for alternatives in such scenarios: First, one can enumerate and evaluate all feature sets satisfying the constraints, which is inefficient. Second, one can sample from the space of valid feature sets. However, uniform sampling from a constrained space is a computationally hard problem, possibly harder than determining if a valid solution exists or not [17]. Third, one can formulate a multi-objective problem to avoid hard constraints on alternatives. However, as explained in Sect. 3.1, we decided to pursue a single-objective formulation with constraints. Fourth, one can push constraints into a search heuristic; this is what we describe next. One idea is to prevent the heuristic from producing invalid feature sets or to make this less likely, e.g., by adding a penalty to the objective function. Another idea is to 'repair' invalid feature sets in the search, e.g., by replacing them with the most similar feature sets satisfying the constraints. Such solver-assisted approaches are popular for finding valid software configurations [22, 28, 79]. In our experiments, we also employ a repair-based approach, which we describe in the following.

Greedy Wrapper

We propose a novel greedy hill-climbing procedure, displayed in Algorithm 1. Unlike standard hill climbing for feature selection [35], our procedure observes constraints. 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 one solution S' containing this swap and satisfying the other constraints. If such a solution S' exists and its quality $Q(S', X, y)$ is higher than the one of the current solution, the algorithm proceeds with the new solution, attempting again to swap the first and second features (Lines 10–12). Otherwise, it tries to swap the next pair of features (Lines 13–17). Specifically, we assess only one solution per swap before proceeding instead of exhaustively enumerating and evaluating all valid solutions involving 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

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
```

the iteration count as the number of invocations of the solver, i.e., attempts to generate valid alternatives. This number also bounds the number of prediction models trained. However, we only train a model for valid solutions (Line 9), 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. For instance, one could prevent decision trees from splitting on a feature if the resulting feature set is too similar to a given feature set. As another example, there are various formal encodings of prediction models, e.g., as SAT formulas [54, 70, 82]. In such representations, one might directly add the constraints for alternatives and employ a solver for 'training'.

3.4 Computational complexity

In this section, we discuss the time complexity of alternative feature selection. We are interested in scalability regarding the number of features $n \in \mathbb{N}$, also taking the feature-set size $k \in \mathbb{N}$ and the number of alternatives $a \in \mathbb{N}_0$ into account. Section 3.4.1 examines exhaustive search, which works for arbitrary feature-selection methods, while Sect. 3.4.2 analyzes the optimization problem with univariate feature qualities (Eq. 11).

3.4.1 Exhaustive search for arbitrary feature-selection methods

An exhaustive search over all possible feature sets is a simple but inefficient approach to finding alternatives. This procedure provides an upper bound for the time complexity of the optimization problem.

Sequential search

Even without considering alternatives, the search space of feature selection grows exponentially with n. In particular, there are $2^n - 1$ non-empty feature sets of arbitrary size and $\binom{n}{k} = \frac{n!}{k!\cdot(n-k)!} \leq n^k$ sets of a fixed size k. These numbers also hold for sequential search (Eq. 9), which optimizes alternatives one at a time. For each feature set, one must evaluate the objective and check the constraints. The cost of the former depends on the feature-selection method but should usually be polynomial in n. The latter 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. In particular, constraint checking involves iterating over all existing feature sets and features to compute the dissimilarity between sets (Eq. A4). With $O(n^k)$ feature sets as solution candidates, we obtain the following proposition:

Proposition 2 (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.*

This complexity is polynomial in n if k is a small constant independent of n, i.e., $k \in O(1)$, and if a is at most polynomial in n, i.e., $a \in O(n^c)$, $c \in O(1)$. The assumption $k \in O(1)$ makes sense for feature selection, where one typically wants to obtain small feature sets from a high-dimensional dataset. However, the exponent k might still render an exhaustive search practically infeasible. In terms of parameterized complexity, the problem resides in class $\mathcal{XP}$ since the complexity term has the form $O(f(k) \cdot n^{g(k)})$ [14] if a is constant, having the parameter k and functions $f(k) = 1$, $g(k) = k + 1$.

Simultaneous search

Simultaneous search enlarges the search space by optimizing $a + 1$ feature sets at once (Eq. 10), having $O((n^k)^{a+1}) = O(n^{k\cdot(a+1)})$ solution candidates. Thus, the following proposition holds:

Proposition 3 (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 respect to n is less favorable compared to sequential search, as the number of alternatives a now appears in the exponent. Further, Proposition 3 assumes that the constraints do not use linearization variables (Eqs. 6 and A5), 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 4 (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

While the assumption $a \cdot k \in O(1)$ ensures polynomial runtime regarding n 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* (Eq. 11). This feature-selection method arguably has the simplest objective function, where the quality of a feature set is equal to the sum of the individual qualities of its constituent features. This simplicity eases the transformation from and to well-known $\mathcal{NP}$-hard problems.

In the following complexity analyses, we assume the feature qualities $q(X_{\cdot j}, y)$ are given. In particular, one can pre-compute these qualities before searching alternatives and treat them as constants in the optimization problem. The complexity of this computation depends on the particular feature-quality measure and the number of data objects m. However, the number of features n should only affect the complexity linearly due to the univariate setting.

We have previously observed that our optimization problem with univariate feature qualities can be formulated as an integer-linear program (Sect. 3.3.1). While INTEGER PROGRAMMING is $\mathcal{NP}$-complete in general [20, 32], our problem is a special case of it and could therefore have lower complexity.

Min-aggregation

We start with two assumptions, both of 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. We call the combination of these assumptions, which implies $n = (a + 1) \cdot k$, a *complete partitioning*. This scenario differs from $a \cdot k \ll n$, which we assumed in Sect. 3.4.1, where runtime was polynomial in n.

The currently assumed scenario is a variant of an existing problem known as MULTI-WAY NUMBER PARTITIONING [40] or MULTIPROCESSOR SCHEDULING [20] in literature: Partition a set of n integers into a fixed number of subsets such that all subset sums are as equal as possible [36]. A typical application is assigning tasks with different lengths to a fixed number of processors, aiming to balance the load. In our scenario, the tasks correspond to features, the task lengths to univariate feature qualities, and the processors to feature sets. Maximizing the minimum sum is one of several possible objectives in the literature [36, 40]. This objective corresponds to simultaneous search (Eq. 10) with min-aggregation (Eq. A2) in our scenario. Since MULTIPROCESSOR SCHEDULING is $\mathcal{NP}$-complete, even for just two partitions [20], we obtain the following result:

Proposition 5 (Complexity of simultaneous search with min-aggregation, complete partitioning, and unconstrained feature-set size) *Assuming univariate feature qualities (Eq. 11), a dissimilarity threshold $\tau = 1$, and 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.*

Dropping several assumptions from Proposition 5, we directly obtain the following, more general proposition:

Proposition 6 (Complexity of simultaneous search with min-aggregation) *Simultaneous search for alternative feature sets with min-aggregation is $\mathcal{NP}$-hard.*

While Proposition 5 allows arbitrary set sizes, the hardness result remains valid for fixed or bounded set sizes k, known as BALANCED NUMBER PARTITIONING [46, 84] or K-PARTITIONING [27, 41] problem in the literature:

Proposition 7 (Complexity of simultaneous search with min-aggregation, complete partitioning, and constrained feature-set size) *Assuming univariate feature qualities (Eq. 11), 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.*

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:

Proposition 8 (Complexity of simultaneous search with min-aggregation, incomplete partitioning, and constrained feature-set size) *Assuming univariate feature qualities (Eq. 11), 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.*

See Appendix A.3 for the proof of this proposition and of the ones that follow.

Next, we also allow $\tau < 1$, i.e., feature-set overlap, and obtain another $\mathcal{NP}$-hardness result:

Proposition 9 (Complexity of simultaneous search with min-aggregation, $\tau < 1$, and constrained feature-set size) *Assuming univariate feature qualities (Eq. 11), 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 hardness results, sum-aggregation (Eq. A1) admits polynomial-time algorithms for $\tau = 1$:

Proposition 10 (Complexity of search with sum-aggregation and $\tau = 1$) *Assuming univariate feature qualities (Eq. 11) 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 result applies to both sequential and simultaneous search, whether using complete or incomplete partitioning, for any arbitrary number a of alternatives, and even when feature sets vary in size k. The primary reason for achieving polynomial runtime is that sum-aggregation does not necessitate 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.

4 Experimental design

In this section, we describe our experimental design. We provide a concise overview of its components and objectives in Sect. 4.1 before delving into a detailed explanation of its components. In particular, we describe evaluation metrics (Sect. 4.2), methods (Sect. 4.3), datasets (Sect. 4.4), and implementation (Sect. 4.5).

4.1 Overview

We conduct experiments with 30 datasets representing binary-classification problems. As evaluation metrics, we take into account feature-set quality and runtime. We compare five feature-selection methods, each representing distinct concepts of feature-set quality. Furthermore, we train prediction models using the resultant feature sets and analyze their prediction performance. In our pursuit of alternatives, we explore both simultaneous and sequential search methods. We systematically vary the number of alternatives a and the dissimilarity threshold τ.

4.2 Evaluation metrics

Our evaluation focuses on the trade-off between feature-set quality and obtaining alternatives. In addition, we evaluate the runtime of the search for alternatives.

Feature-set quality

We use two metrics for feature-set quality. First, we evaluate the objective functions $Q(s, X, y)$ of the feature-selection methods and report their *objective value*. Second, we train prediction models with the feature sets found. We report *prediction performance* in terms of Matthews correlation coefficient (MCC) [45], which is insensitive to imbalanced prediction targets, reaches 1 for perfect predictions, and is 0 for random guessing.

To analyze how well feature selection and predictions generalize, we conduct a stratified fivefold cross-validation. Model training and the search for alternatives only have access to the training data. However, we also use the test data to evaluate the quality of each feature set found with the training data. For the test-set objective value, we initialize the objective function with feature qualities computed on the test set but insert the feature selection from the training set. For the test-set prediction performance, we predict on the test set but use a prediction model trained with these features on the training set.

Runtime

Regarding runtime, we first analyze the *optimization time*. For white-box feature-selection methods, this corresponds to the summed runtime of all solver calls. We exclude the time for feature-quality computations that one can reuse for multiple solver calls. For *Greedy Wrapper*, we measure the runtime of the entire black-box optimization procedure with multiple solver calls and model trainings. As a second metric, we examine the *optimization status*, which can take four values. If the solver finished before reaching its 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 is unclear or no valid solution (yet), so the problem is *not solved*.

4.3 Methods

4.3.1 Prediction

As prediction models, we use decision trees [7] since these models allow learning complex, nonlinear dependencies. Preliminary experiments with random forests [8] and a k-nearest neighbors classifier yielded similar insights. We leave the hyperparameters of the trees at their defaults, except for using information gain instead of Gini impurity as the split criterion, to be consistent with our filter feature-selection methods. The trees may only use features from the alternatives, which allows to assess the quality of alternatives.

4.3.2 Feature selection (objective functions)

We choose five well-known feature-selection methods to provide objective functions for optimization. They cover the different feature-selection categories from Sect. 2.2 except *embedded*, as explained in Sect. 3.3.3. However, we use feature importance from an embedded method, i.e., decision trees, as post hoc importance scores.

One method (*Greedy Wrapper*) requires black-box optimization, while the other four methods have white-box objectives. With each feature-selection method, we select $k \in \{5, 10\}$ features, yielding small feature sets. We enforce the desired k with an additional constraint in the optimization, using the feature-set-size expression from Eq. (5).

Filter feature selection

As filter methods, we use the univariate *MI* (Eq. 11), the multivariate *FCBF* (Eq. 12), and the multivariate *mRMR* (Eq. 13). In all three methods, mutual information [37] serves as the dependency measure $q(\cdot)$. Since mutual information has no fixed upper bound, we normalize its values 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 our hill-climbing procedure *Greedy Wrapper* (Algorithm 1) with $max\_iters = 1000$. To evaluate feature-set quality within the wrapper, we

use the MCC score of a decision tree on a 20% validation split of the data.

Post hoc feature importance

As a post hoc importance measure called *Model Gain*, we use importance scores from *scikit-learn's* decision trees. There, importance quantifies a feature's contribution toward 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 these importances into Eq. (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.

4.3.3 Alternatives (constraints)

Optimization

We address the optimization problem of alternative feature selection with a solver. Thus, when we speak of *sequential search* and *simultaneous search* in the evaluation, we refer to the following solver-based optimization procedures rather than the optimization problems in general: For the four white-box feature-selection methods, we use the solver to exactly solve the underlying optimization problems. Thus, given sufficient solving time, these alternatives are globally optimal. With *Greedy Wrapper* as the feature-selection method, the search procedure (Algorithm 1) is heuristic and might not cover the entire search space. There, the solver only assists in finding valid solutions but does not optimize.

Competitors

Other approaches are possible to tackle the sequential and simultaneous optimization problem for alternative feature selection. E.g., heuristic approaches may speed up the optimization for the white-box feature-selection methods but are out of the scope of the current article. Another source for competitors could be related work. However, as we discuss in Sect. 6, 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 search approaches, we can still put the feature-set quality into perspective by comparing alternatives to each other. In particular, the quality of the 'original' feature set, i.e., obtained by running the feature-selection methods without constraints for alternatives, serves as a natural reference point.

Search parametrization

We analyze *sequential* (Eq. 9) and *simultaneous* (Eq. 10) search for alternatives. For the latter, we employ sum-aggregation (Eq. A1) and min-aggregation (Eq. A2) in the objective. In figures and tables, we use the abbreviations *seq.*, *sim. (sum)*, and *sim. (min)* to denote these search methods.

We vary the parameters of the search systematically: We evaluate $a \in \{1, \ldots, 10\}$ alternatives for sequential search and $a \in \{1, \ldots, 5\}$ 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 (Eqs. 3 and 8). Thus, for $k = 5$, we consider $\tau \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, corresponding to an overlap of four to zero features. For $k = 10$, we consider $\tau \in \{0.1, 0.2, \ldots, 1.0\}$. We exclude $\tau = 0$, which would allow returning duplicate feature sets.

Timeout

We employ a solver timeout to make a large-scale evaluation feasible and to account for the high variance of solver runtime. In particular, we grant each solver call 60 s multiplied by the number of feature sets sought. Thus, sequential search conducts multiple solver calls with 60 s timeout each, while simultaneous search conducts one solver call with proportionally more time, e.g., 300 s for five feature sets (i.e., four alternatives). 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.

4.4 Datasets

Selection criteria

We use a variety of datasets from the Penn Machine Learning Benchmarks (PMLB) [60, 67]. To harmonize evaluation, we only consider binary classification, though alternative feature selection works for regression and multi-class problems as well. We exclude datasets containing fewer than 100 data objects since they might incur a high uncertainty when assessing feature-set quality. Otherwise, the number of data objects should not systematically impact feature-set quality and is unimportant for our evaluation. We also exclude datasets with fewer than 15 features to leave 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.

30 datasets with 106 to 9822 data objects and 15 to 168 features remain. The datasets do not contain any missing values. Categorical features have an ordinal encoding by default. Table 2 provides an overview of the datasets used.

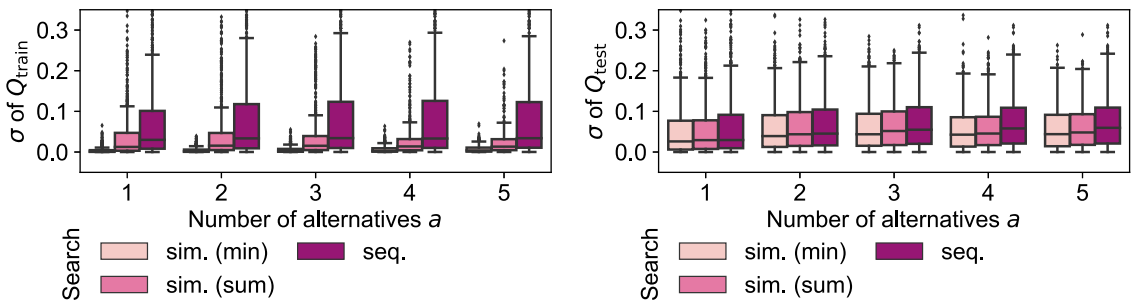
Feature correlation

Table 2 also displays the datasets' average feature correlation. In particular, we compute Spearman's rank correlation between each pair of features and take the absolute values to evaluate the strength of dependencies rather than their sign. For the datasets in our study, the average feature correlations

Table 2 Datasets from PMLB used in our experiments. *m* denotes the number of data objects and *n* the number of features

Dataset	*m*	*n*	Mean corr.
backache	180	32	0.10
chess	3196	36	0.08
churn	5000	20	0.04
clean1	476	168	0.25
clean2	6598	168	0.25
coil2000	9822	85	0.07
credit_a	690	15	0.12
credit_g	1000	20	0.07
dis	3772	29	0.08
GAMETES_Epistasis_2_Way_20atts_0.1H_EDM_1_1	1600	20	0.02
GAMETES_Epistasis_2_Way_20atts_0.4H_EDM_1_1	1600	20	0.02
GAMETES_Epistasis_3_Way_20atts_0.2H_EDM_1_1	1600	20	0.02
GAMETES_Heterogeneity_20atts_1600_Het_0.4_0.2_50_EDM_2_001	1600	20	0.02
GAMETES_Heterogeneity_20atts_1600_Het_0.4_0.2_75_EDM_2_001	1600	20	0.02
hepatitis	155	19	0.15
Hill_Valley_with_noise	1212	100	1.00
horse_colic	368	22	0.16
house_votes_84	435	16	0.30
hypothyroid	3163	25	0.15
ionosphere	351	34	0.25
molecular_biology_promoters	106	57	0.08
mushroom	8124	22	0.18
ring	7400	20	0.02
sonar	208	60	0.21
spambase	4601	57	0.14
spect	267	22	0.20
spectf	349	44	0.19
tokyo1	959	44	0.44
twonorm	7400	20	0.17
wdbc	569	30	0.42

Mean corr. is the average of absolute values of all pairwise Spearman's rank correlations between features



(a) Training-set objective value.

(b) Test-set objective value.

Fig. 1 Standard deviation of feature-set quality within search runs 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

are often weak, mostly below 0.3. Generally, correlated features indicate that alternative feature sets may exist. However, there are two caveats. First, rank correlation only captures certain types of dependencies, while our feature-selection criteria and prediction models are more general. Second, for optimal alternatives, the dependency between highly predictive features is crucial, while the dependency between unimportant features matters less. However, the table only shows the mean over all feature pairs.

4.5 Implementation and execution

We implemented our experimental pipeline in Python 3.8, using *scikit-learn* [62] for machine learning and the solver *SCIP* [5] via the package *OR-Tools* [64] 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 h, i.e., about 10.6 days.

5 Evaluation

In this section, we evaluate our experiments. In particular, we discuss the parametrization for searching alternatives: the search method (Sect. 5.1) and the two user parameters (Sect. 5.2), i.e., the number of alternatives a and dissimilarity threshold τ.

5.1 Search methods for alternatives

Variance in feature-set quality

As expected, the search method influences how much the training-set objective value Q varies among alternatives within each search run. Figure 1a visualizes this observation for *MI*, though it applies to all other white-box feature-selection methods as well. Each box in the figure shows how the variance within individual search runs for alternatives is distributed over other experimental settings, e.g., datasets and cross-validation folds. In particular, the quality of multiple alternatives from the same search run varies less for simultaneous than sequential search and less for min-aggregation than sum-aggregation. However, this difference in variance largely disappears on the test set, for the objective value (Fig. 1b) as well as prediction performance. This effect might result from overfitting: Even with similar training-set quality, some alternatives may generalize better 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 quality-homogeneous alternatives can be one goal of simultaneous search, the main selling point would be obtaining alternatives of higher average quality than sequential search. However, this potential advantage rarely materialized in our experiments. In particular, Fig. 2a compares the distribution of the mean training-set objective in search runs for *MI* as the feature-selection method. We observe that all three search methods yield similar distributions of training-set objective values. This observation holds for feature-set quality on the test set (Fig. 2b) as well. Further, the other four feature-selection methods besides *MI* also do not show a general quality advantage of simultaneous search. Finally, depending on the concrete experimental setting, either sequential or simultaneous search can yield significantly higher quality than the other search method, even if the overall mean difference is close to zero.

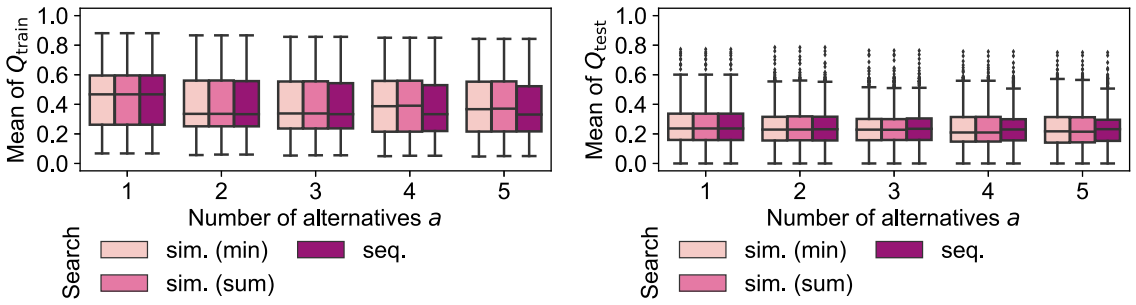
Optimization status

One reason why simultaneous search fails to consistently beat the quality of sequential search is that search results may 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. In particular, the corresponding solver status *feasible* occurs more often for simultaneous than sequential search, as Table 3 shows. 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. The feature-selection method *mRMR* is especially prone to suboptimal solutions, likely because it has a more complex objective than *MI* and *Model Gain*. In contrast, *FCBF* often results in infeasible optimization problems since its constraints preventing the selection of redundant features might prevent finding any valid feature set of size k.

Further, the fraction of timeouts strongly depends on the number of alternatives a. E.g., for simultaneous search with $k = 5$ and sum-aggregation, roughly 8% of the white-box searches timed out for one alternative but 20% for three alternatives and 30% for five alternatives. While we grant simultaneous searches proportionally more time to obtain more feature sets, the runtime increases clearly super-proportionally, as we analyze next.

Optimization time

The optimization time also favors sequential search. As Table 4 shows, the mean optimization time of sequential search is lower for all five feature-selection methods, partly by orders of magnitude. Further, *FCBF*, *MI*, and *Model Gain* experience a dramatic increase in optimization time with parameter a. E.g., for simultaneous search with $k = 5$ and sum-aggregation, *MI* has a mean optimization time of 0.03 s



(a) Training-set objective value.

(b) Test-set objective value.

Fig. 2 Mean of feature-set quality within search runs over the number of alternatives a, by search method for alternatives and evaluation metric. Results with *MI* as feature-selection method and $k = 5$

Table 3 Frequency of optimization statuses (Sect. 4.2) by feature-selection method and search method for alternatives

Feature selection	Search	Optimization status		
		Infeasible (%)	Feasible (%)	Optimal (%)
FCBF	seq.	74.51	0.00	25.49
FCBF	sim. (min)	73.07	1.73	25.20
FCBF	sim. (sum)	73.07	2.19	24.75
MI	seq.	4.93	0.00	95.07
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.	4.88	9.63	85.49
mRMR	sim. (min)	4.67	49.04	46.29
mRMR	sim. (sum)	4.67	67.39	27.95

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%

Table 4 Mean optimization time by feature-selection method and search method for alternatives

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

Results with $k = 5$ and $a \in \{1, 2, 3, 4, 5\}$

for $a = 1$, 0.09 s for $a = 2$, 0.31 s for $a = 3$, 3.84 s for $a = 4$, and 122.69 s for $a = 5$. In contrast, the runtime increase is considerably less for sequential search, which shows an approximately linear trend with the number of alternatives.

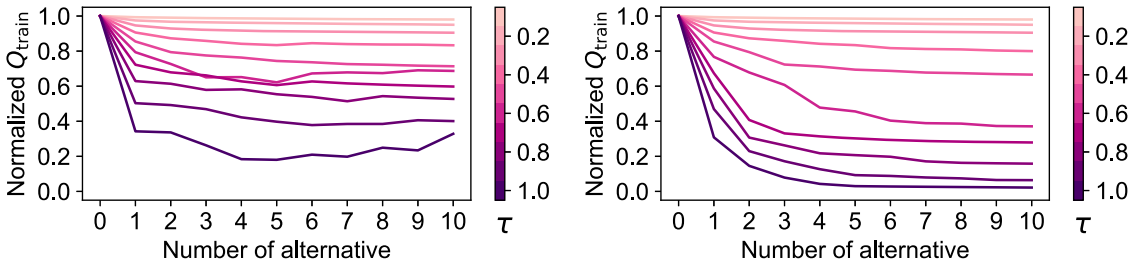
Another interesting question is how the runtime relates to n, the number of features in the dataset. Our experimental data show a positive correlation, as we expected. However, the observed trend is very noisy, particularly for simultaneous search. E.g., runtimes vary considerably even for a fixed n, and some higher-dimensional datasets even show lower average runtimes than lower-dimensional ones. This result indicates that 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.

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

5.2 User parameters a and τ

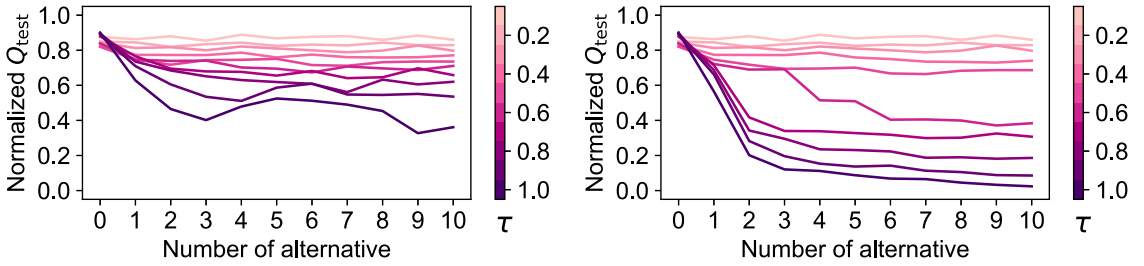
Feature-set quality

One would expect a decrease in feature-set quality with an increasing number of alternatives a and dissimilarity threshold τ, giving users control over alternatives. In particular,



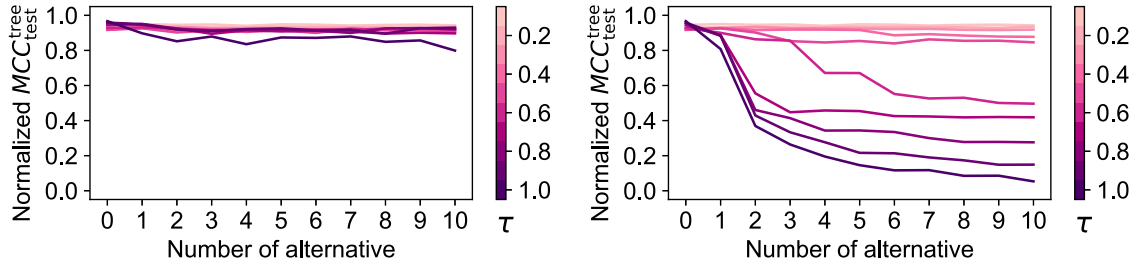
(a) Training-set objective value. Infeasible feature sets excluded.

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



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

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



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

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

Fig. 3 Mean of feature-set quality, max-normalized per search run for alternatives, over the number of alternatives and dissimilarity threshold τ, by evaluation metric. Results from sequential search with *MI* as feature-selection method and $k = 10$

higher values of the two user parameters introduce more (for a) and stronger (for τ) constraints to the optimization problem. Figure 3 shows this decreasing quality trend for *MI*-based feature selection. Since feature-set quality varies significantly among datasets, we display it normalized to $[0, 1]$ within each search run for alternatives: We shift the range of all evaluation metrics to $[0, 1]$ and max-normalize feature-set quality for each search of alternatives, i.e., the highest feature-set quality in the search run is set to 1, and the other qualities are scaled accordingly.

As visible in Fig. 3, there might be multiple alternatives of similar quality, particularly for low values of τ. Further, the objective value decreases most from the original, unconstrained feature set, i.e., the zeroth alternative in the figures, to the first alternative, but the decrease is less beyond. Note that Fig. 3 averages the normalized feature-set quality over datasets with different dimensionality. In our experiments, datasets with more features n tend to experience a smaller decrease in quality over a and τ. As higher-dimensional datasets offer more options for alternatives, this observation makes sense. However, this effect is not guaranteed since datasets with many features could also contain many useless features instead of interesting alternatives.

The overall decrease in quality is slightly less pronounced for the test-set objective value (Fig. 3c) than for the training-set one (Fig. 3a) since overfitting might occur. In particular, the original feature set can even have lower test-set quality than the subsequent alternatives. The change in feature-set quality becomes even less clear when using the alternative feature sets to train prediction models, i.e., decision trees. As Fig. 3e shows, the prediction performance varies little over the number of alternatives and the dissimilarity threshold τ. In particular, the optimization objective Q may only partially indicate actual prediction performance since the

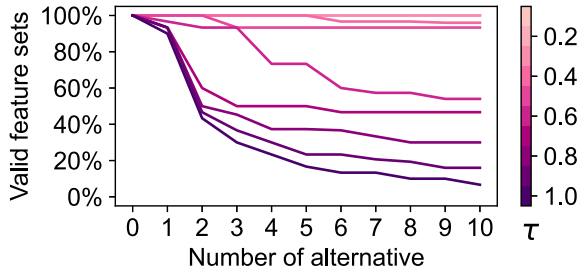


Fig. 4 Fraction of optimization runs yielding a valid feature set over the number of alternatives and dissimilarity threshold τ. Results from sequential search with *MI* as feature-selection method and $k = 10$

former may use a simplified feature-set quality criterion. Indeed, the overall correlation between optimization objective Q and prediction MCC is only weak to moderate in our experiments. Among the feature-selection methods, this correlation is highest for *Greedy Wrapper* but also not perfect there. In particular, the search procedure of *Greedy Wrapper* evaluates feature sets with a validation split of the training set. MCC on this holdout set may not perfectly correspond to MCC on the test set, which is not used in the search.

Optimization status

Increasing a and τ does not only affect quality but can also render the optimization problem infeasible, i.e., no valid alternative might exist. Figure 4 visualizes how the percentage of valid feature sets develops over a and τ in our experiments. In our prior analysis of feature-set quality, we excluded infeasible feature sets. For comparison, Figs. 3b, d, f show the same data as Fig. 3a, c, e but with the quality of infeasible feature sets set to zero, i.e., the theoretical minimum quality after we shifted the value ranges of evaluation metrics. In these three figures, the decrease in feature-set quality is noticeably stronger for all evaluation metrics. In particular, test-set prediction performance, which is relatively stable when excluding infeasible feature sets (Fig. 3e), also decreases now (Fig. 3f).

In contrast, if only considering valid feature sets, the mean quality may even increase over the number of alternatives, as visible in Fig. 3a and Fig. 3c for $\tau = 1.0$. This counter-intuitive 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

While we discussed *MI* before, the decrease in the objective value over parameter a occurs for all feature-selection methods in our experiments, as Fig. 5a shows. However, the strength of the decrease varies considerably among the feature-selection methods. For example, *Greedy Wrapper* and *mRMR* show the least effect of a. The effect of τ also depends on the feature-selection method, as Fig. 5b shows. Again, *MI* and *Model Gain*, which both use the simple univariate objective (Eq. 11), show a stronger trend than *Greedy Wrapper* and *mRMR*. The fifth feature-selection method, *FCBF*, also exhibits a clear quality decrease over a and τ, mainly due to a high rate of infeasible feature sets rather than a quality decrease of the found feature sets. Ultimately, choosing a particular feature-selection method depends on the use case. Thereby, users should be aware that effects regarding alternative feature sets may depend on this choice.

6 Related work

In this section, we review related work from the field of feature selection and other areas relevant to this article. To our knowledge, searching for optimal alternative feature sets in the sense of this article is novel. However, there is literature on optimal alternatives outside the field of feature selection. Additionally, there has been research focused on discovering multiple diverse feature sets.

Feature selection

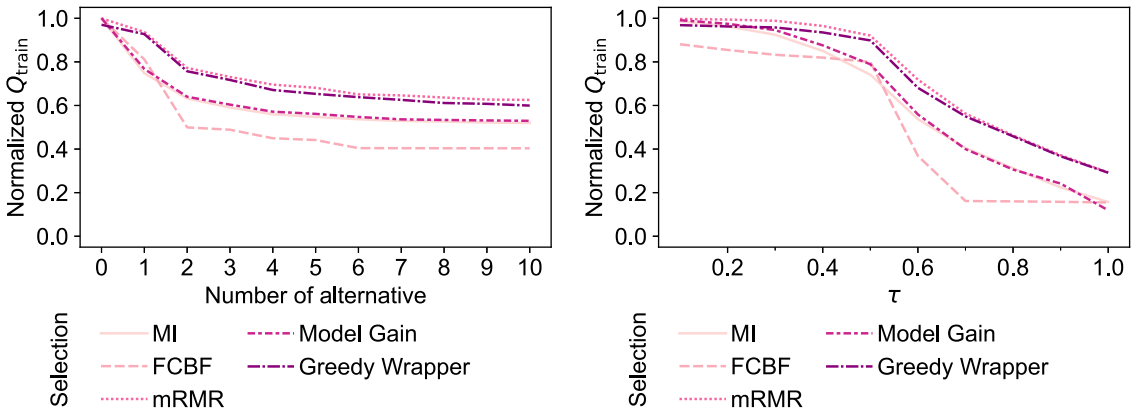
Most feature-selection methods yield one solution [6], though there are some exceptions [16, 52, 72], e.g., fostering diversity within wrapper methods. Ensemble feature selection [69, 71] combines feature-selection results obtained by different feature-selection methods or on different samples of the data. There, diversity can be an auxiliary goal to improve prediction performance [23, 43, 80]. In addition, the notion of feature-set diversity differs from ours. In particular, these approaches give users less control over alternatives, e.g., do not enforce alternatives with hard constraints.

Approaches for statistically equivalent feature sets [6, 39] use statistical tests to determine feature sets that are equivalent for predictions. In contrast, we do not require equivalent quality but allow users to control the number and dissimilarity of alternatives directly.

There also is work on constraints for feature selection, e.g., for feature cost [61], feature groups [83], or domain knowledge [2, 21]. Such constraints do not explicitly target alternatives but could be integrated into our integer-programming formulation as well.

Subgroup discovery

Diversity also is an issue in subgroup set discovery, which searches for interesting regions in the data space instead of selecting features as a whole. [76] presents six strategies to facilitate subgroup diversity. Two are directly transferable to feature selection, but the criteria for being alternative are different from ours. In particular, one strategy prunes similar subgroups with exactly the same quality as previous solutions. The other one limits the number of subgroups a feature may occur in but does not constrain the overlap of subgroups per se.



(a) Over the number of alternatives a.

(b) Over the dissimilarity threshold τ.

Fig. 5 Mean of training-set objective value, max-normalized per search run for alternatives, by feature-selection method and evaluation metric. Infeasible feature sets assigned a quality of 0. Results from sequential search with $k = 10$

Clustering

Finding alternative solutions has been addressed extensively in the field of clustering. [4] gives a taxonomy and describes algorithms for alternative clustering. Our problem definition is different in several respects. The notion of dissimilarity differs since we strive for differently composed feature sets, while alternative clustering targets different assignments of data objects to clusters. Next, using constraints is only one approach to obtain alternative clusterings. Finally, we optimize feature-set quality in a supervised prediction scenario while clustering is unsupervised.

Subspace search

Finding several useful feature sets also is part of subspace clustering [29, 51] and subspace search [19, 58, 74]. These approaches strive to improve data-mining algorithms by using subspaces, i.e., feature sets, and not all features. Some approaches explicitly try to remove redundancy between subspaces [51, 58] or increase subspace diversity [19, 74]. See [29] for an overview. Alternative feature selection is different in at least one of the following aspects: First, the objective function, i.e., the notion of quality, differs. Second, definitions of subspace redundancy may hinge on feature values instead of binary selection decisions. Third, users typically do not have any control over the dissimilarity. At best, there is a regularization parameter rather than a hard threshold.

Explainable artificial intelligence (XAI)

In XAI, alternative explanations might provide additional insights, allow to test different hypotheses, and foster trust in the predictions [34, 78]. In contrast, obtaining significantly different explanations for a prediction might raise doubts regarding their meaningfulness [30]. Diversity is a criterion studied for various explainers, e.g., for counterfactuals [13, 31, 47, 50, 68, 77], criticisms [33], or semifactual explanations [1]. There are several approaches to foster diversity, e.g., ensembling explanations [73], considering multiple local minima [77], using a search algorithm that maintains diversity [13], extending the optimization objective [1, 33, 50], or introducing constraints [31, 47, 68]. The last option is similar to the way we enforce alternatives. Of all these approaches, only [1, 47, 50] introduce parameters to control the diversity of solutions; only [47] offers a user-friendly dissimilarity threshold in [0, 1], while the other two approaches regularize the objective. Further, all the previously mentioned XAI techniques provide local explanations, i.e., aim at prediction outcomes for individual data objects rather than optimizing the global prediction quality of feature sets.

Rashomon sets

A Rashomon set is a set of prediction models that reach a certain, e.g., close-to-optimal, prediction performance [18]. Despite similar performance, these models may still assign different feature importance scores, leading to different explanations [38]. Thus, Rashomon sets may yield partial information about alternative feature sets. However, approaches for Rashomon sets do not explicitly search for alternative feature sets as a whole, i.e., feature sets satisfying a dissimilarity threshold relative to other sets. Instead, these approaches focus on the range of each feature's importance over prediction models. Further, our notion of alternatives is not bound to model-based feature importance but encompasses a broader range of feature-selection methods. Finally, we use importance scores from one model instead of multiple models to find importance-based alternatives.

7 Conclusions and future work

Conclusions

Our article has studied alternative feature selection, the problem of identifying diverse feature sets that simultaneously maintain high quality. We formalized alternative feature

selection as an optimization problem, using constraints that are independent of the chosen feature-selection method. These constraints can be combined with other constraints on feature sets and offer users control over diversity via two parameters: the number of alternatives and a dissimilarity threshold. We analyzed the computational complexity of this optimization problem, showing its $\mathcal{NP}$-hardness. Additionally, we discussed the integration of various categories of feature-selection methods. Finally, we assessed the effectiveness of alternative feature selection using 30 datasets representing binary-classification problems.

Future work

In the current article, we conducted a broad quantitative evaluation of alternative feature selection on datasets from various domains. Practitioners can employ alternative feature selection in domain-specific case studies and evaluate the alternative feature sets qualitatively, thereby assessing their usefulness for interpretability. Within alternative feature selection, opportunities exist to modify the formulation of the optimization problem, such as by incorporating soft constraints or adopting a multi-objective optimization approach instead of employing hard constraints. Furthermore, one can integrate additional feature-selection methods into the search for alternatives, including embedded methods. Finally, there is potential for enhancing the efficiency of the search for alternatives, particularly in the case of simultaneous search, by developing heuristics instead of seeking exact solutions to the optimization problem.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the Ministry of Science, Research and the Arts Baden-Württemberg, project *Algorithm Engineering for the Scalability Challenge (AESC)*.

Availability of data and materials All experimental data are available online at https://doi.org/10.35097/1975.

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

Declarations

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

Appendix A: Appendix

In this section, we provide technical details and proofs. Section 1 discusses aggregation operators for the objective of simultaneous search (Eq. 10). Section A.2 contains complete definitions of the alternative-feature-selection problem (Sect. 3.2) for the univariate objective (Eq. 11). Sect. A.3 supplies proofs for the complexity analysis (Sect. 3.4).

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 (Eq. 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*, as defined by Eq. (A1):

$$\max_{s^{(0)},\ldots,s^{(a)}} \sum_{i=0}^{a} Q(s^{(i)}, X, y) \tag{A1}$$

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 (Eq. 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 possible 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 possible 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,

as defined by Eq. (A2):

$$\max_{s^{(0)},\ldots,s^{(a)}} \min_{i\in\{0,\ldots,a\}} Q(s^{(i)}, X, y) \tag{A2}$$

In the terminology of social choice theory, this objective uses an egalitarian rule instead of a utilitarian one [53]. In particular, *min-aggregation* maximizes the quality of the worst selected alternative. Thereby, it incentivizes all alternatives to have high quality and implicitly balances their quality.

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 (Example 1). In the following, we demonstrate the other direction:

Example 2 (Min-aggregation) Consider $n = 6$ features with univariate feature qualities (Eq. 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 feature-set quality, only the first solution optimizes the objective with sum-aggregation. In particular, min-aggregation permits reducing the quality of feature sets as long as it remains above the minimum of all sets.

From the technical perspective, Eq. (A2) has the disadvantage of being nonlinear regarding the decision variables $s^{(0)}, \ldots, s^{(a)}$. However, we can linearize it with one constraint per feature set and an auxiliary variable $Q_{\min}$, as shown in Eq. (A3):

$$\begin{aligned} \max_{s^{(0)},\ldots,s^{(a)}} \quad & Q_{\min} \\ \text{s.t.: } \forall i \in \{0,\ldots,a\}: \quad & Q_{\min} \leq Q(s^{(i)}, X, y) \\ & Q_{\min} \in \mathbb{R} \end{aligned} \tag{A3}$$

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 [49].

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 incentivizes high quality for all sets. One can alleviate this issue by adapting the objective or constraints. First, related work on Multi-Way Number Partitioning (Sect. 3.4.2) also uses other objectives for balancing [36, 40]. 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 optimizing 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 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 Sect. 3. We use the objective of univariate filter feature selection (Eq. 11). The corresponding feature qualities $q(\cdot)$ are constants in the optimization problem. Further, we use the Dice dissimilarity (Eqs. 3 and 8) to measure feature-set dissimilarity for alternatives. The dissimilarity threshold $\tau \in [0, 1]$ is a user-defined constant. Finally, we assume fixed, user-defined feature-set sizes $k \in \mathbb{N}$.

Sequential alternatives

Inthe 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. We obtain the following optimization problem (Eq. A4):

$$\begin{aligned} \max_{s} \quad & Q_{\text{uni}}(s, X, y) = \sum_{j=1}^{n} q(X_{\cdot j}, y) \cdot s_j \\ \text{s.t.:} \quad & \forall F_{\bar{s}} \in \mathbb{F}: \quad \sum_{j=1}^{n} s_j \cdot \bar{s}_j \leq (1-\tau) \cdot k \\ & \sum_{j=1}^{n} s_j = k \\ & s \in \{0,1\}^n \end{aligned} \tag{A4}$$

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 (Eq. 6). Finally, we use sum-aggregation (Eq. A1) over the alternatives in the objective here. We obtain the following optimization problem (Eq. A5):

$$
\begin{aligned}
\max_{s^{(0)},\ldots,s^{(a)}} \quad & \sum_i Q_{\text{uni}}(s^{(i)}, X, y) \\
& = \sum_i \sum_j q(X_{\cdot j}, y) \cdot s_j^{(i)} \\
\text{s.t.:} \quad \forall i_1\, \forall i_2 : \quad & \sum_j t_j^{(i_1,i_2)} \leq (1-\tau) \cdot k \\
\forall i_1\, \forall i_2\, \forall j : \quad & t_j^{(i_1,i_2)} \leq s_j^{(i_1)} \\
\forall i_1\, \forall i_2\, \forall j : \quad & t_j^{(i_1,i_2)} \leq s_j^{(i_2)} \\
\forall i_1\, \forall i_2\, \forall j : \quad & 1 + t_j^{(i_1,i_2)} \geq s_j^{(i_1)} + s_j^{(i_2)} \\
\forall i : \quad & \sum_j s_j^{(i)} = k \\
\forall i : \quad & s^{(i)} \in \{0,1\}^n \\
\forall i_1\, \forall i_2 : \quad & t^{(i_1,i_2)} \in \{0,1\}^n \\
\text{with:} \quad & i \in \{0,\ldots,a\} \\
& i_1 \in \{1,\ldots,a\} \\
& i_2 \in \{0,\ldots,i_1-1\} \\
& j \in \{1,\ldots,n\}
\end{aligned}
\tag{A5}
$$

A.3 Complexity proofs

In this section, we provide proofs for propositions from Sect. 3.4.

Proof of Proposition 8 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 in the selected feature qualities, selecting feature f' in a 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 (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 (Proposition 7), 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 9 Let an arbitrary problem instance I of the complete-partitioning problem be given and the feature-set size k be fixed. We create a new problem 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 problem 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 of the problem instance overall. 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., $\tau = \{0, 1/k, \ldots, (k-1)/k, 1\}$. 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 10 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 assign 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: This objective cannot decrease when selecting features of higher quality. Thus, we 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. □

References

1. Artelt, A., Hammer, B.: "Even if ..."—diverse semifactual explanations of reject (2022). arXiv:2207.01898 [cs.LG]
2. Bach, J., Zoller, K., Trittenbach, H., et al.: An empirical evaluation of constrained feature selection. SN Comput. Sci. **3**(6) (2022). https://doi.org/10.1007/s42979-022-01338-z
3. Bach, J.: Finding optimal diverse feature sets with alternative feature selection (2023). arXiv:2307.11607v1 [cs.LG]
4. Bailey, J.: Alternative clustering analysis: a review. In: Data Clustering: Algorithms and Applications, 1st edn. CRC Press, chap 21, pp. 535–550 (2014) https://doi.org/10.1201/9781315373515
5. Bestuzheva, K., Besançon, M., Chen, W.K., et al.: The SCIP Optimization Suite 8.0. Tech. rep., Zuse Institute Berlin, Germany (2021) http://nbn-resolving.de/urn:nbn:de:0297-zib-85309
6. Borboudakis, G., Tsamardinos, I.: Extending greedy feature selection algorithms to multiple solutions. Data Min. Knowl. Disc. **35**(4), 1393–1434 (2021). https://doi.org/10.1007/s10618-020-00731-7
7. Breiman, L., Friedman, J.H., Olshen, R.A., et al.: Classification and Regression Trees, 1st edn. Wadsworth (1984). https://doi.org/10.1201/9781315139470
8. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324
9. Carvalho, D.V., Pereira, E.M., Cardoso, J.S.: Machine learning interpretability: a survey on methods and metrics. Electronics **8**(8) (2019). https://doi.org/10.3390/electronics8080832
10. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Comput. Electr. Eng. **40**(1), 16–28 (2014). https://doi.org/10.1016/j.compeleceng.2013.11.024
11. Choi, S.S., Cha, S.H., Tappert, C.C.: A survey of binary similarity and distance measures. J. Syst. Cybern. Inf. **8**(1), 43–48 (2010)
12. Covert, I., Lundberg, S.M., Lee, S.I.: Understanding global feature contributions with additive importance measures. In: Proceedings of NeurIPS, pp. 17212–17223 (2020), https://proceedings.neurips.cc/paper/2020/file/c7bf0b7c1a86d5eb3be2c722cf2cf746-Paper.pdf
13. Dandl, S., Molnar, C., Binder, M., et al.: Multi-objective counterfactual explanations. In: Proceedings of PPSN, pp. 448–469 (2020) https://doi.org/10.1007/978-3-030-58112-1_31
14. Downey, R.G., Fellows, M.R., Stege, U.: Parameterized complexity: a framework for systematically confronting computational intractability. In: Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future, pp. 49–99 (1997) https://doi.org/10.1090/dimacs/049/04
15. Egghe, L.: New relations between similarity measures for vectors based on vector norms. J. Am. Soc. Inf. Sci. Technol. **60**(2), 232–239 (2009). https://doi.org/10.1002/asi.20949
16. Emmanouilidis, C., Hunter, A., MacIntyre, J., et al.: Selecting features in neurofuzzy modelling by multiobjective genetic algorithms. In: Proceedings of ICANN, pp. 749–754 (1999) https://doi.org/10.1049/cp:19991201
17. Ermon, S., Gomes, C., Selman, B.: Uniform solution sampling using a constraint solver as an oracle. In: Proceedings of UAI, pp. 255–264 (2012) https://www.auai.org/uai2012/papers/160.pdf
18. Fisher, A., Rudin, C., Dominici, F.: All models are wrong, but many are useful: learning a variable's importance by studying an entire class of prediction models simultaneously. J. Mach. Learn. Res. **20**(177), 1–81 (2019)
19. Fouché, E., Kalinke, F., Böhm, K.: Efficient subspace search in data streams. Inf. Syst. **97** (2021). https://doi.org/10.1016/j.is.2020.101705
20. Garey, M.R., Johnson, D.S.: Computers and Intractibility: A Guide to the Theory of NP-Completeness, 24th edn. W. H. Freeman and Company (2003) https://www.worldcat.org/title/440655898
21. Groves, W.C.: Toward automating and systematizing the use of domain knowledge in feature selection. Ph.D. thesis, University of Minnesota (2015) https://hdl.handle.net/11299/175444
22. Guo, J., Shi, K.: To preserve or not to preserve invalid solutions in search-based software engineering: a case study in software product lines. In: Proceedings of ICSE, pp. 1027–1038 (2018) https://doi.org/10.1145/3180155.3180163
23. Guru, D.S., Suhil, M., Raju, L.N., et al.: An alternative framework for univariate filter based feature selection for text categorization. Pattern Recognit. Lett. **103**, 23–31 (2018). https://doi.org/10.1016/j.patrec.2017.12.025
24. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**(Mar), 1157–1182 (2003)
25. Hall, M.A.: Correlation-based feature selection for machine learning. Ph.D. thesis, University of Waikato, Hamilton, New Zealand (1999). https://www.cs.waikato.ac.nz/~ml/publications/1999/99MH-Thesis.pdf
26. Hall, M.A.: Correlation-based feature selection of discrete and numeric class machine learning. Tech. rep., University of Waikato, Hamilton, New Zealand (2000). https://hdl.handle.net/10289/1024
27. He, Y., Tan, Z., Zhu, J., et al.: k-partitioning problems for maximizing the minimum load. Comput. Math. Appl. **46**(10–11), 1671–1681 (2003). https://doi.org/10.1016/S0898-1221(03)90201-X

28. Henard, C., Papadakis, M., Harman, M., et al.: Combining multi-objective search and constraint solving for configuring large software product lines. In: Proceedings of ICSE, pp. 517–528 (2015). https://doi.org/10.1109/ICSE.2015.69
29. Hu, J., Pei, J.: Subspace multi-clustering: a review. Knowl. Inf. Syst. **56**(2), 257–284 (2018). https://doi.org/10.1007/s10115-017-1110-9
30. Jain, S., Wallace, B.C.: Attention is not explanation. In: Proceedings of NAACL-HLT, pp. 3543–3556 (2019). https://doi.org/10.18653/v1/N19-1357
31. Karimi, A.H., Barthe, G., Balle, B., et al.: Model-agnostic counterfactual explanations for consequential decisions. In: Proceedings of AISTATS, pp. 895–905 (2020). https://proceedings.mlr.press/v108/karimi20a.html
32. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations. Plenum Press, pp. 85–103 (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
33. Kim, B., Khanna, R., Koyejo, O.: Examples are not enough, learn to criticize! criticism for interpretability. In: Proceedings of NIPS (2016). https://proceedings.neurips.cc/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf
34. Kim, M.Y., Atakishiyev, S., Babiker, H.K.B., et al.: A multi-component framework for the analysis and design of explainable artificial intelligence. Mach. Learn. Knowl. Extract. **3**(4), 900–921 (2021). https://doi.org/10.3390/make3040045
35. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**(1–2), 273–324 (1997). https://doi.org/10.1016/S0004-3702(97)00043-X
36. Korf, R.E.: Objective functions for multi-way number partitioning. In: Proceedings of SoCS, pp. 71–72 (2010). https://doi.org/10.1609/socs.v1i1.18172
37. Kraskov, A., Stögbauer, H., Grassberger, P.: Estimating mutual information. Phys. Rev. E **69**(6) (2004). https://doi.org/10.1103/PhysRevE.69.066138
38. Laberge, G., Pequignot, Y., Khomh, F., et al.: Partial order in chaos: consensus on feature attributions in the rashomon set (2023). arXiv:2110.13369v2 [cs.LG]
39. Lagani, V., Athineou, G., Farcomeni, A., et al.: Feature selection with the R package MXM: Discovering statistically equivalent feature subsets. J. Stat. Softw. **80**(7), 1–25 (2017). https://doi.org/10.18637/jss.v080.i07
40. Lawrinenko, A.: Identical parallel machine scheduling problems: structural patterns, bounding techniques and solution procedures. Ph.D. thesis, Friedrich-Schiller-Universität Jena (2017), https://nbn-resolving.org/urn:nbn:de:gbv:27-dbt-20170427-0956483
41. Lawrinenko, A., Schwerdfeger, S., Walter, R.: Reduction criteria, upper bounds, and a dynamic programming based heuristic for the max-min k_i-partitioning problem. J. Heuristics **24**, 173–203 (2018). https://doi.org/10.1007/s10732-017-9362-9
42. Li, J., Cheng, K., Wang, S., et al.: Feature selection: a data perspective. ACM Comput. Surv. **50**(6) (2017). https://doi.org/10.1145/3136625
43. Liu, K., Tian, J.: Subspace learning with an archive-based genetic algorithm. In: Proceedings of IEEM, pp. 181–188 (2018). https://doi.org/10.1007/978-981-13-3402-3_20
44. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of NIPS (2017) https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf
45. Matthews, B.W.: Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim. Biophys. Acta - Protein Struct. **405**(2), 442–451 (1975). https://doi.org/10.1016/0005-2795(75)90109-9
46. Michiels, W., Aarts, E., Korst, J., et al.: Computer-assisted proof of performance ratios for the differencing method. Discrete Optim. **9**(1), 1–16 (2012). https://doi.org/10.1016/j.disopt.2011.10.001
47. Mohammadi, K., Karimi, A.H., Barthe, G., et al.: Scaling guarantees for nearest counterfactual explanations. In: Proceedings of AIES, pp. 177–187 (2021). https://doi.org/10.1145/3461702.3462514
48. Molnar, C., Casalicchio, G., Bischl, B.: Interpretable machine learning: a brief history, state-of-the-art and challenges. In: Proceedings of XKDD, pp. 417–431 (2020). https://doi.org/10.1007/978-3-030-65965-3_28
49. MOSEK, A.P.S.: MOSEK modeling cookbook : Mixed integer optimzation (2022). https://docs.mosek.com/modeling-cookbook/mio.html. Accessed 18 Oct 2022
50. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of FAT*, pp. 607–617 (2020). https://doi.org/10.1145/3351095.3372850
51. Müller, E., Assent, I., Günnemann, S., et al.: Relevant subspace clustering: mining the most interesting non-redundant concepts in high dimensional data. In: Proceedings of ICDM, pp. 377–386 (2009). https://doi.org/10.1109/ICDM.2009.10
52. Müller, I.M.: Feature selection for energy system modeling: identification of relevant time series information. Energy AI **4** (2021). https://doi.org/10.1016/j.egyai.2021.100057
53. Myerson, R.B.: Utilitarianism, egalitarianism, and the timing effect in social choice problems. Econometrica **49**(4), 883–897 (1981). https://doi.org/10.2307/1912508
54. Narodytska, N., Ignatiev, A., Pereira, F., et al.: Learning optimal decision trees with SAT. In: Proceedings of IJCAI, pp. 1362–1368 (2018). https://doi.org/10.24963/ijcai.2018/189
55. Nguyen, X.V., Chan, J., Romano, S., et al.: Effective global approaches for mutual information based feature selection. In: Proceedings of KDD, pp. 512–521 (2014). https://doi.org/10.1145/2623330.2623611
56. Nguyen, H., Franke, K., Petrović, S.: Optimizing a class of feature selection measures. In: Proceedings of DISCML (2009). https://www.researchgate.net/publication/231175763
57. Nguyen, H.T., Franke, K., Petrović, S.: Towards a generic feature-selection measure for intrusion detection. In: Proceedings of ICPR, pp. 1529–1532 (2010). https://doi.org/10.1109/ICPR.2010.378
58. Nguyen, H.V., Müller, E., Böhm, K.: 4S: Scalable subspace search scheme overcoming traditional a priori processing. In: Proceedings of Big Data, pp. 359–367 (2013). https://doi.org/10.1109/BigData.2013.6691596
59. Njoku, U.F., Abelló, A., Bilalli, B., et al.: Wrapper methods for multi-objective feature selection. In: Proceedings of EDBT, pp. 697–709 (2023). https://doi.org/10.48786/edbt.2023.58
60. Olson, R.S., La Cava, W., Orzechowski, P., et al.: PMLB: a large benchmark suite for machine learning evaluation and comparison. Biodata Min. **10** (2017). https://doi.org/10.1186/s13040-017-0154-4
61. Paclík, P., Duin, R.P.W., van Kempen, G.M.P., et al.: On feature selection with measurement cost and grouped features. In: Proceedings of SSPR /SPR, pp. 461–469 (2002). https://doi.org/10.1007/3-540-70659-3_48
62. Pedregosa, F., Varoquaux, G., Gramfort, A., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**(85), 2825–2830 (2011)
63. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. **27**(8), 1226–1238 (2005). https://doi.org/10.1109/TPAMI.2005.159
64. Perron, L., Furnon, V.: OR-Tools (2022). https://developers.google.com/optimization/. Accessed 18 Oct 2022
65. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you? Explaining the predictions of any classifier. In: Proceedings of KDD, pp. 1135–1144 (2016). https://doi.org/10.1145/2939672.2939778

66. Rodriguez-Lujan, I., Huerta, R., Elkan, C., et al.: Quadratic programming feature selection. J. Mach. Learn. Res. **11**(49), 1491–1516 (2010)
67. Romano, J.D., Le, T.T., La Cava, W., et al.: PMLB v1.0: an open source dataset collection for benchmarking machine learning methods (2021). arXiv:2012.00058v3 [cs.LG]
68. Russell, C.: Efficient search for diverse coherent explanations. In: Proceedings of FAT*, pp. 20–28 (2019). https://doi.org/10.1145/3287560.3287569
69. Saeys, Y., Abeel, T., Peer, Y.V.D.: Robust feature selection using ensemble feature selection techniques. In: Proceedings of ECML PKDD, pp. 313–325 (2008). https://doi.org/10.1007/978-3-540-87481-2_21
70. Schidler, A., Szeider, S.: SAT-based decision tree learning for large data sets. In: Proceedings of AAAI, pp. 3904–3912 (2021). https://doi.org/10.1609/aaai.v35i5.16509
71. Seijo-Pardo, B., Porto-Díaz, I., Bolón-Canedo, V., et al.: Ensemble feature selection: homogeneous and heterogeneous approaches. Knowl-Based Syst. **118**, 124–139 (2017). https://doi.org/10.1016/j.knosys.2016.11.017
72. Siddiqi, U.F., Sait, S.M., Kaynak, O.: Genetic algorithm for the mutual information-based feature selection in univariate time series data. IEEE Access **8**, 9597–9609 (2020). https://doi.org/10.1109/ACCESS.2020.2964803
73. Silva, W., Fernandes, K., Cardoso, J.S.: How to produce complementary explanations using an ensemble model. In: Proceedings of IJCNN (2019). https://doi.org/10.1109/IJCNN.2019.8852409
74. Trittenbach, H., Böhm, K.: Dimension-based subspace search for outlier detection. Int. J. Data Sci. Anal. **7**(2), 87–101 (2019). https://doi.org/10.1007/s41060-018-0137-7
75. Ulrich-Oltean, F., Nightingale, P., Walker, J.A.: Selecting SAT encodings for pseudo-boolean and linear integer constraints. In: Proceedings of CP, pp. 38:1–38:17 (2022). https://doi.org/10.4230/LIPIcs.CP.2022.38
76. van Leeuwen, M., Knobbe, A.: Diverse subgroup set discovery. Data Min. Knowl. Discov. **25**(2), 208–242 (2012). https://doi.org/10.1007/s10618-012-0273-y
77. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: automated decisions and the GDPR. Harv. J. Law Technol. **31**(2), 841–887 (2017)
78. Wang, D., Yang, Q., Abdul, A., et al.: Designing theory-driven user-centric explainable AI. In: Proceedings of CHI (2019). https://doi.org/10.1145/3290605.3300831
79. White, J., Benavides, D., Schmidt, D.C., et al.: Automated diagnosis of feature model configurations. J. Syst. Softw. **83**(7), 1094–1107 (2010). https://doi.org/10.1016/j.jss.2010.02.017
80. Woznica, A., Nguyen, P., Kalousis, A.: Model mining for robust feature selection. In: Proceedings of KDD, pp. 913–921 (2012). https://doi.org/10.1145/2339530.2339674
81. Yu, L., Liu, H.: Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of ICML, pp. 856–863 (2003). https://aaai.org/Papers/ICML/2003/ICML03-111.pdf
82. Yu, J., Ignatiev, A., Stuckey, P.J., et al.: Learning optimal decision sets and lists with SAT. J. Artif. Intell. Res. **72**, 1251–1279 (2021). https://doi.org/10.1613/jair.1.12719
83. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. J. R. Stat. Soc.: Ser. B (Stat. Methodol.) **68**(1), 49–67 (2006). https://doi.org/10.1111/j.1467-9868.2005.00532.x
84. Zhang, J., Mouratidis, K., Pang, H.: Heuristic algorithms for balanced multi-way number partitioning. In: Proceedings of IJCAI, pp. 693–698, (2011). https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-122