Finding Optimal Solutions for Alternative Feature Selection

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

***Abstract*—Feature-selection methods are popular to obtain small, interpretable, yet highly accurate prediction models. Existing feature-selection methods typically yield only one feature set, which might not be sufficient in some cases. For example, users might be interested in finding different feature sets with similar prediction quality, offering alternative explanations of the data. In this article, we formalize alternative feature selection as optimization problem. Next, we propose several general approaches to solve this problem. In particular, we show how to integrate various categories of existing feature-selection methods. Finally, we evaluate these approaches in a study with 30 classification datasets. In our experiments, we are able to determine alternative feature sets with high prediction quality.**

***Index Terms*—feature selection, alternatives, constraints, explainability, interpretability**

I. Introduction

a) Motivation: Feature-selection methods are ubiquitous for various reasons. By reducing dataset dimensionality, they lower computational and memory requirements of prediction models. Next, models might generalize better if irrelevant and spurious predictors are removed. Finally, prediction models might be smaller and more comprehensive due to feature selection [1], improving interpretability.

Traditional feature-selection methods mostly return only one feature set [2]. 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. For the user, these alternative feature sets might be interesting for multiple reasons. First, some of the originally selected features might be costly to obtain, so the user would prefer cheaper alternatives. Second, some of the originally selected features might contain sensitive information that should not be used in predictions. Third, the user might be interested in explanations for the prediction. In such a situation, knowing alternatives allows formulating multiple hypotheses and broadens the understanding.

b) Problem statement: This article addresses the problem of alternative feature selection, which we define as follows:

Definition 1 (Alternative feature selection (informal))**.** Given an original feature set, find a sufficiently different feature set that optimizes feature-set quality at the same time.

We provide formal definitions later. If the original feature set had high quality, the alternative feature set should have similar quality. Depending on how different the alternative should be, one might have to compromise on quality. E.g., if original feature set already contains most of the highly predictive features, the alternative might have significantly lower quality. We analyze this effect in our article. Also, we consider finding multiple alternatives rather than just one.

Two points are essential for alternative feature selection, which we both address in this article. First, one needs to formalize and quantify what an alternative feature sets is. Second, one needs an approach to efficiently find alternative feature sets. Ideally, the approach should be general, i.e., cover a broad range of existing feature-selection methods.

c) Related work: In machine learning, finding alternative solutions has already been addressed extensively in clustering [3]. However, there is a lack of such approaches and corresponding experimental evaluation for feature selection. Only few feature-selection methods target at obtaining multiple, diverse feature sets [2], [4]. Techniques for ensemble feature selection [5], [6] and statistically equivalent feature subsets [7] produce multiple feature sets, but do not focus on optimal alternatives. In the field of explainable AI, counterfactual explanations have massively gained popularity in the last few years [8], [9]. Such explanations involve data objects with similar feature values, but a different prediction outcome. In contrast, we target at sets of different features, but similar feature-set quality, e.g., prediction performance.

d) Contributions: Our contribution is threefold. First, we formalize alternative feature selection as an optimization problem. In particular, we foster alternatives via constraints on feature sets. Second, we propose heuristic and exact approaches to solve this optimization problem. To that end, we discuss how to integrate different categories of feature-selection methods in the objective function. Our notion of alternative feature sets is general and thus not tailored towards specific feature-selection methods. Third, we evaluate our approaches for alternative feature selection with comprehensive experiments. We use 30 classification dataset from the Penn Machine Learning Benchmarks (PMLB) [10], [11] and four different feature-selection methods. We focus on the question if one can find alternative feature sets with similar feature-set quality as the original ones.

e) Results: Our experiments show that finding alternative feature sets with similar feature-set quality is possible. This

outcome encourages using alternative feature sets as a tool for alternative explanations of predictions. As expected, feature-set quality tends to decrease with the number of alternatives. Naturally, the exact decrease depends on the dataset and how feature-set quality is distributed in it. Further, we note that the quality of alternatives significantly depends on the dissimilarity threshold for how alternative feature sets should be. Thereby, this threshold allows the user to exercise control over alternatives and make use-case specific choices.

We publish our code[1] and our experimental data[2] online.

f) Outline: Section II explains fundamentals of feature selection. Section III defines the optimization problem of alternative feature selection and discusses approaches to solve it. Section IV reviews related work. Section V describes our experimental design. Section VI presents the experimental results. Section VII concludes.

II. Fundamentals

In this section, we introduce basic notation and review different methods to measure quality of feature sets.

A. Notation

Let $X \in \mathbb{R}^{m \times n}$ be a dataset represented as a matrix. Each row is a data object, and each column is a feature. Let $F = \{f_1, \ldots, f_n\}$ denote the set of feature names. We assume categorical features have already been made numeric, e.g., via one-hot encoding. Let $X_{\cdot j} \in \mathbb{R}^m$ denote the vector representation of the j-th feature. Further, let $y \in \mathbb{R}^m$ represent the prediction target. The actual domain of the target might also be smaller, e.g., $\{0, 1\}$ for binary classification.

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

B. Measuring Feature (Set) Quality

There are different ways to evaluate feature-set quality $Q(s, X, y)$. Note that we only give a short overview here, and focus on methods that we use in our evaluation. See [12], [1] for comprehensive surveys of feature selection. A typical categorization of feature selection is into filter, wrapper, and embedded methods [13].

a) Filter methods: Filter methods evaluate feature sets without training a prediction model. Univariate filters assess each feature on its own, while multivariate filters evaluate feature sets. Examples for univariate filters are the Pearson correlation or the mutual information between a feature and the prediction target. Such methods ignore interaction between features, e.g., redundancies. Multivariate methods, like CFS [14], FCBF [15], and mRMR [16], often combine a measure of feature relevance with a measure of feature redundancy to express feature-set quality.

b) Wrapper methods: Wrapper methods [17] assess feature-set quality by training a prediction model with the feature set and evaluating prediction performance. They can use an arbitrary black-box optimization strategy to iterate over feature sets, e.g., genetic algorithms.

c) Embedded methods: Embedded methods have feature selection built into training of prediction models, e.g., decision trees [18] or random forests [19]. Thus, the criterion to evaluate feature-set quality often is model-specific. Tree-based models typically use information gain or the Gini index.

d) Post-hoc feature-importance methods: Apart from traditional feature selection, there are various methods that assess feature importance after training a model. Assessing feature importance plays a crucial role in the emerging fields of ML interpretability [20]. Approaches range from local explanation methods, like LIME [21] or SHAP [22], to global importance methods, like permutation importance [19] or SAGE [23].

III. Alternative Feature Selection

In this section, we present the problem and approaches for alternative feature selection. First, we give an overview of the optimization problem. Second, we formalize the notion of alternatives via constraints. Third, we discuss different objective functions, corresponding to different feature-set quality measures from Section II-B In particular, we describe how to solve the resulting the optimization problem.

A. Optimization Problem

In alternative feature selection, there are two goals. First, the quality of an alternative feature set should be high. Second, an alternative feature set should be different to one or more existing feature set(s). There are different ways to combine these two goals in an optimization problem:

First, one can consider both goals as objectives, obtaining an unconstrained multi-objective problem. Second, one can treat feature-set quality as objective and enforce alternatives with constraints. Third, one can consider being alternative as objective and constrain feature-set quality, e.g., with a lower bound. Fourth, one can define constraints for both, feature-set quality and being alternative, searching for feasible solutions instead of optimizing. Depending on the use case, any of these four formulations might be appropriate.

Following the informal Definition 1 from the introduction, we stick to the second formulation, i.e., optimizing feature-set quality subject to being alternative. This option has the advantage of keeping the original objective function of feature selection. Consequently, one does not need to specify a range or a threshold on feature-set quality. Instead, the user can control how alternative the feature set should be. This yields the following optimization problem:

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

We discuss different constraints for *being alternative* and different objective functions $Q(s, X, y)$ in the following.

[1] https://github.com/Jakob-Bach/AFS

[2] temporary link: https://bwsyncandshare.kit.edu/s/xxx; will be moved to a public repository after review

B. Constraints – Defining Alternative Feature Sets

In this section, we formalize alternative feature sets. This part our work is independent from the feature-selection method. First, we discuss the base case where an individual feature set is an alternative to another one. Second, we extend this notion to multiple alternatives, considering sequential as well as simultaneous search procedures.

1) Single Alternatives: We consider a feature set to be an alternative to another feature set if it differs sufficiently. Mathematically, we express this with a set-dissimilarity measure [24]. Most of these measures describe how strongly two sets overlap and set this in relation to the size of the sets. For example, one can define dissimilarity based on the Dice coefficient, a simple and well-known set-similarity measure:

$$d_{Dice}(F_1, F_2) = 1 - \frac{2 \cdot |F_1 \cap F_2|}{|F_1| + |F_2|} \tag{2}$$

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

Definition 2 (Pairwise alternative)**.** A feature set F_2 is an alternative to a feature set F_1 (and vice versa) for a dissimilarity threshold $\tau \in [0, 1]$ if $d(F_1, F_2) \geq \tau$.

Depending on the preferences of the user, different values of τ might be appropriate. τ controls how alternative the new feature set should be. A feature set that differs strongly might cause a large drop in feature-set quality. Thus, we leave τ as a parameter. If the set-dissimilarity measure $d()$ is normalized to $[0, 1]$, like the Dice dissimilarity, the interpretation of tau is user-friendly: A value of 0 allows the alternative feature set to be identical to the original feature set. A value of 1 requires the two feature sets to have no overlap.

If the feature sets sizes $|F_1|$ and $|F_2|$ are known, a threshold τ corresponds to a particular maximum number of overlapping features $|F_1 \cap F_2|$. This follows from re-arranging Equation 2 in combination with Definition 2:

$$d_{Dice}(F_1, F_2) \geq \tau \Leftrightarrow |F_1 \cap F_2| \leq \frac{1 - \tau}{2} \cdot (|F_1| + |F_2|) \tag{3}$$

The latter formulation also avoids having the feature-set sizes, and thereby the decision variables, in a quotient. We can see that the effect of setting τ on the maximum overlap size $|F_1 \cap F_2|$ is linear. For the Jaccard distance, another popular set-dissimilarity measure, we would obtain a non-linear expression regarding τ.

If we additionally require $|F_1| = |F_2|$, the parameter τ directly controls which fraction of features in one set needs to differ from the other set, and vice versa:

$$\begin{aligned} &\text{If } |F_1| = |F_2| : \quad d_{Dice}(F_1, F_2) \geq \tau \\ \Leftrightarrow\ &|F_1 \cap F_2| \leq (1 - \tau) \cdot |F_1| = (1 - \tau) \cdot |F_2| \end{aligned} \tag{4}$$

Up to now, we have worked with feature set sizes instead of the binary feature-selection vector s. However, expressing the feature-set sizes in terms of s is straightforward:

$$|F_s| = \sum_{j=1}^{n} s_j \quad \text{and} \quad |F_{s_1} \cap F_{s_2}| = \sum_{j=1}^{n} s_{1,j} \cdot s_{2,j} \tag{5}$$

Combining Equation 3 with Equation 5 yields an inequality only involving sums and products of the decision variables and constants. In fact, one can replace the product $s_{1,j} \cdot s_{2,j}$ with linear constraints by introducing an auxiliary variable t_j [25]:

$$\begin{aligned} t_j \leq & s_{1,j} & t_j \leq & s_{2,j} \\ 1 + t_j \geq & s_{1,j} + s_{2,j} & t_j \in & \{0, 1\} \end{aligned} \tag{6}$$

If there is an existing feature set, i.e., one either knows s_1 or s_2, Equation 5 already is linear without Equation 6.

Overall, one can express alternative features sets according to Definition 2 with 0-1 integer linear constraints. This simple constraint type allows using a broad range of solvers, given the right objective function, which we discuss later.

2) Multiple Alternatives: If the user desires multiple alternative feature sets rather than just one, one can compute these alternatives either sequentially or simultaneously.

a) Sequential alternatives: In the sequential case, we obtain alternatives iteratively, with one alternative feature set per iteration. We constrain the new feature set to be alternative to all previously found feature sets:

Definition 3 (Sequential alternative)**.** A feature set F_2 an alternative to a set of feature sets $\mathbb{F}$ (and vice versa) for a dissimilarity threshold $\tau \in [0, 1]$ if $\forall F_1 \in \mathbb{F} : d(F_1, F_2) \geq \tau$.

This definition duplicates the constraint from the base case and instantiates it for different existing feature sets. One could also use a less strict constraint, e.g., requiring only the average dissimilarity to all existing feature sets to pass the threshold. The objective function remains the same as in the base case, i.e., we only optimize the quality of the new feature set F_2. Thus, the number of variables in the optimization problem remains constant, independent from the number of alternatives. Further, we do not need to introduce interaction variables as in Equation 6. Each alternative only adds one new constraint to the optimization problem. Thus, we expect the runtime of sequential search to scale well with the number of alternatives. If the solver keeps a state between iterations and can warm-start, further runtime gains might be possible. Regarding feature-set quality, one can expect a drop with each alternative, as the solution space becomes narrower over iterations. The user can decide after each iteration if feature-set quality is too low or if another alternative should be found.

b) Simultaneous alternatives: In the simultaneous case, we directly obtain several alternatives. The user needs to decide on the number of feature sets $\mathbb{F}$ beforehand. This entails pairwise dissimilarity constraints:

Definition 4 (Simultaneous alternatives)**.** A set of feature sets $\mathbb{F}$ contains pairwise alternatives for a dissimilarity threshold $\tau \in [0, 1]$ if $\forall F_1 \in \mathbb{F}, F_2 \in \mathbb{F}, F_1 \neq F_2 : d(F_1, F_2) \geq \tau$.

Again, one could resort to less strict constraints, e.g., based on the average dissimilarity between alternatives. In contrast to the sequential case, we need to introduce further decision variables and modify the objective function here, as we optimize multiple feature sets at once. In this paper, we consider the summed quality of all feature sets as objective.

Runtime-wise, we expect simultaneous search to scale worse with the number of alternatives than sequential search. The number of decision variables increases linearly with the number of alternatives, while the number of constraints grows quadratically. Also, for each feature and each pair of alternatives, we need to introduce an interaction variable as in Equation 6 if we want to obtain a linear problem.

In contrast to the greedy procedure of the sequential procedure, the simultaneous procedure optimizes alternatives globally. Thus, for the same number of alternatives, the simultaneous procedure should yield higher average feature-set quality. Also, we expect the qualities of the alternatives to be more evenly distributed, opposed to the dropping quality over the course of the sequential procedure.

C. Objective Functions – Finding Alternative Feature Sets

In this section, we present concrete approaches to find alternative feature sets. We discuss how to solve the optimization problem from Section III-A for the different categories of feature-set quality measures from Section II-B. In particular, we categorize solution approaches as white-box optimization, black-box optimization, and embedding alternatives.

1) White-Box Optimization: If feature-set quality $Q(s, X, y)$ is a sufficiently simple function, one can tackle alternative feature selection with a white-box solver.

a) Univariate filter feature selection: For univariate filter feature selection, the objective function is linear. This makes the optimization problem of alternative feature selection an 0-1 integer linear problem. In particular, univariate filter methods decompose the quality of a feature set into the quality of the individual features:

$$Q_{uni}(s, X, y) = \sum_{j=1}^{n} s_j \cdot q(X_{\cdot j}, y) \tag{7}$$

Here, q typically is a bivariate dependency measure, e.g., mutual information [26] or correlation, to quantify the relationship between a feature and the prediction target.

b) Post-hoc feature importance: From the technical perspective, one can also insert values of post-hoc feature importance scores for q in Equation 7. For example, one can precompute permutation importance [19] or SAGE scores [23] for each feature. However, such post-hoc importance scores often evaluate the usefulness of features under the presence of other features. In consequence, the feature-independence assumption underlying Equation 7 does not hold. However, re-calculating feature importance for each possible feature set makes a white-box approach infeasible. In practice, one can still use Equation 7 with importance scores computed on the full dataset. One only needs to be aware that such an approach might not represent feature interactions faithfully.

c) Multivariate filter feature selection: Several multivariate filter methods also enable a simple white-box formulation, though not necessarily a linear one. In the following, we discuss the Fast Correlation-Based Filter (FCBF) [15], which we use in our experiments. Correlation-based Feature Selection (CFS) [14] and Minimal Redundancy Maximum Relevance (mRMR) [16] allow white-box optimization [27] as well.

FCBF [15] bases on the notion of predominance: It strives to select features that have at least a certain correlation to the prediction target, but a lower correlation to all other features. While the original FCBF algorithm uses a heuristic search, we propose a formulation as a constrained optimization problem:

$$\begin{aligned} \max_{s} \quad & Q_{FCBF}(s, X, y) = \sum_{j=1}^{n} s_j \cdot q(X_{\cdot j}, y) \\ \text{subject to:} \quad & \forall (j_1, j_2) \in \{1, \ldots, n\}^2, j_1 \neq j_2 : \\ & s_{j_1} \cdot s_{j_2} \cdot q(X_{\cdot j_1}, X_{\cdot j_2}) < q(X_{\cdot j_1}, y) \end{aligned} \tag{8}$$

We drop the original FCBF's threshold parameter on feature-target correlation and instead maximize the latter, as in the univariate filter case. With the constraint in Equation 8, we enforce that if two feature are selected, their correlation is lower than each feature's target correlation. Overall, the optimization problem is linear again if one appropriately handles the product term between decision variables.

2) Black-Box Optimization: If feature-set quality has not closed-form expression, one can treat it as a black-box function when searching for alternative feature sets. This applies to the category of wrapper feature selection. Using search heuristics, one can optimize such black-box functions by systematically iterating over candidate feature sets.

However, search heuristics often assume an unconstrained search space. For example, they might propose a candidate feature set that is not alternative enough. We see multiple ways to address this challenge: First, one can enumerate and evaluate all feature sets satisfying the constraints, which is inefficient. Second, one can sample valid feature sets, which is not easy if the procedure should sample uniformly [28]. Third, one can formulate a multi-objective problem, so there are no hard constraints on alternatives any more. Fourth, one can adapt a search heuristic to consider constraints; this is what we describe in the following.

There are several approaches for adapting search heuristics to constraints. One idea is to prevent a search heuristic from producing feature sets that violate the constraints, or at least making the latter less likely, e.g., be penalizing the objective function accordingly. Another idea is to 'repair' feature sets proposed by the search that are not alternative enough. For example, one can replace a feature set violating the constraint with the most similar feature set satisfying the constraints. Such solver-assisted search approaches are common in search procedure for software configuration [29], [30], [31].

As a simple wrapper approach for our experiments, we use a greedy hill-climbing strategy, as displayed in Figure 1. Compared to standard hill-climbing [17], our search only evaluates feature sets that satisfy the constraints for alternatives. First, the algorithm uses a solver to find one solution that is sufficiently alternative. Next, it tries 'swapping' a feature, i.e., selecting the feature if it was deselected or deselecting it if it was selected. The algorithm applies the solver again to find a

Input: Dataset X, Prediction target y,
Feature-set quality function $Q(\cdot)$,
Constraints for alternatives $Cons$,
Maximum number of iterations max_iters
Output: Feature-selection decision vector s

```
s ← Solve(Cons)      // Initial alternative
iters ← 1            // Number of solver calls
if s = ∅ then return ∅      // No alternative
j ← 1                // Index to be swapped
while iters < max_iters and j ≤ |s| do
    s' ← Solve(Cons ∪ {¬s_j})  // Swap feature
    iters ← iters + 1
    if s' ≠ ∅ and Q(s', X, y) > Q(s, X, y) then
        s ← s'           // Swap if improved
        j ← 1
    else j ← j + 1       // Try next feature
return s
```

Fig. 1. Constraint-aware greedy wrapper feature selection.

solution s' containing this change and satisfying the other constraints. If such a solution s' exists and its quality $Q(s', X, y)$ improves the current solution, the algorithm continuous from the new solution. Else, it attempts to swap the next feature. The algorithm terminates if no swap leads to an improvement or a fixed number of iterations max_iters is reached. Each iteration corresponds to a solver call. This also is an upper bound on the number of prediction models trained. However, not all solver calls might yield a valid feature set.

3) Embedding Alternatives: If feature selection is embedded into training a prediction model, there is no general approach for finding alternative feature sets. Instead, one would need to embed the search for alternatives into training as well. In consequence, approaches for embedding alternatives will be specific to prediction models. For example, one could prevent decision trees from splitting on a feature if the resulting feature set was too similar to an existing feature set. We leave the formulation of such approaches open for future work.

IV. Related Work

In this section, we review related work from areas related to alternative feature selection: alternative clustering, subspace search, feature selection, and counterfactual explanations. To the best of our knowledge, searching for optimal alternative feature sets is novel. However, there is literature on optimal alternatives outside the field of feature selection. Also, there are works on finding multiple, diverse feature sets.

a) Alternative clustering: Finding alternative solutions has been addressed extensively in the field of clustering. [3] gives a taxonomy of alternative clustering and describes algorithms from different categories of alternative clustering. Our problem definition in Sections III-A and III-B is, on a high level, inspired by the one in [3]: find one or multiple solutions that maximize quality while minimizing similarity. [3] also draws the distinction between singular alternatives and multiple alternatives, found sequentially or simultaneously. Nevertheless, the concrete problem definition for our feature-selection scenario is different.

b) Subspace search: Finding multiple feature set plays a role in subspace clustering [32], [33], [34] and subspace search [35], [36], [37]. These approaches strive to improve the results of data-mining algorithms by using subspaces, i.e., feature sets, rather than the full space, i.e., all features. Some subspace approaches explicitly try to remove redundancy between subspaces [32], [35] or foster subspace diversity [36], [37], which bears some resemble to alternative feature selection. Nevertheless, these unsupervised scenarios have fundamentally different goals than our approaches for supervised feature selection.

c) Feature selection: Most feature-selection methods only yield one solution [2], though there are some exceptions. Nevertheless, none of the following approaches explicitly searches for optimal alternatives. [4] adapts a genetic algorithm to ensure diversity of feature sets by penalizing overlap of feature sets. [38] uses multi-objective genetic algorithms to obtain prediction models that use feature sets of different sizes. [39] clusters features based on their values and then forms all combinations of picking one feature from each cluster. [40] presents six strategies to foster diversity in subgroup set discovery, which searches for interesting regions in the data space rather than selecting features.

d) Ensemble feature selection: Ensemble feature selection [5], [6] runs feature selection on different versions, e.g., samples, of the data, or uses different feature-selection methods. While fostering diverse feature sets might be a subgoal, the results of the different feature selectors are combined at the end. Thus, this focus is different from our ours. As an example, [41] uses clustering and frequent-itemset mining to reduce a set of feature-selection results, obtained on bootstrap samples of the data, to a smaller, yet diverse set. [42] builds an ensemble prediction model from classifiers trained on different feature sets and uses an evolutionary algorithm that ensures diversity of feature sets. [43] selects features separately for each class and then combines the results.

e) Statistically equivalent feature sets: A subfield of feature selection is the search for statistically equivalent feature sets [7], [2]. These approaches use statistical tests to determine which features or feature sets can be used interchangeably to predict a target. While equivalent feature sets are related to our notion of alternative feature sets, there are differences as well. Depending on the configuration of statistical tests, there can be an arbitrary number of equivalent feature sets, without a clear ordering. Instead, we always provide a fixed number of alternatives. Rather than striving for equivalence between feature sets, we optimize under constraints. Our dissimilarity threshold allows to control overlap of feature sets, instead of eliminating all feature redundancies. Nevertheless, if one wants to definitely prevent selection of certain redundant features or feature sets, one can achieve this with additional constraints in our optimization problem as well.

f) Constrained feature selection: There is work on considering various kinds of constraints in feature selection, e.g., for costs [44], feature group [45], or domain-knowledge [46]. These approaches are orthogonal to our work, as such constraints can be added to our optimization problem as well.

g) Counterfactual explanations: Alternative feature selection shares some similarities with the concept of counterfactual explanations for predictions [8], [9]. Counterfactual explanations also provide alternative solutions and might be formulated as optimization problem [47], [48]. In particular, [49] presents an approach to find diverse counterfactual explanations. However, there are crucial differences between counterfactuals and alternative feature sets: Counterfactual explanations target at predictions on single data objects rather than global predictive quality of features. Also, counterfactuals want to alter the prediction outcome by changing feature values little, while alternative feature sets should keep the prediction quality while changing the feature set significantly.

V. Experimental Design

In this section, we describe our experimental design. First, we give a brief overview of its goal and components. Next, we elaborate on different components of the design in detail.

A. Overview

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

B. Methods

1) Feature Selection (Objective Functions): To have a diverse set of feature-selection methods, we consider the different categories from Section II-B. We choose four feature selection-methods that are well-known and either parameter-free or easy to parameterize. See Section III-C for details on the objective function of each method. One method (*Greedy Wrapper*) requires black-box optimization, while the other three methods have white-box objectives. As explained in Section III-C3, we do not include an embedded method.

For each feature-selection method, we set the number of selected features to $k \in \{5, 10\}$. This yields small feature sets, fostering interpretability. In our optimization problem, we can enforce the desired k with a simple constraint.

a) MI: For univariate filtering, we use Equation 7 with mutual information [26] as dependency measure $q()$. To improve comparability between datasets and values of k, we normalize the qualities such that selecting all features yields a quality of 1 and selecting no feature yields a quality of 0.

b) FCBF: As multivariate filter method, we use our constrained formulation of FCBF [15] according to Equation 8. As bivariate quality measure within FCBF, we rely on mutual information again, normalized to sum up to 1.

c) Greedy Wrapper: As wrapper method, we employ a greedy hill-climbing strategy according to Algorithm 1. We set max_iters to 1000. To evaluate feature-set quality within the wrapper, we choose a decision tree as model and Matthews correlation coefficient (MCC) [50] as performance metric. Also, we apply a stratified 80:20 holdout split to test generalization performance. Thus, $Q(s, X, y)$ corresponds to the MCC on the 20% validation part of the data.

d) Model Gain: As post-hoc importance measure, we take model-based feature importance provided by *scikit-learn*. Again, we use a decision tree as model. There, importance expresses a feature's contribution towards optimizing the split criterion of the tree, which for which we choose information gain, i.e., mutual information. We plug the importances into Equation 7, i.e., treat the importances like univariate filter scores. Note that the interpretation is different, though. Here, the scores originate from trees involving multiple features, rather than assessing features in isolation. The model-based importances are normalized to sum up to 1 by default.

2) Alternatives (Constraints): We employ *sequential* as well as *simultaneous* search for alternatives, using the corresponding definitions from Section III-B2. For sequential search, we evaluate 1 to 10 alternatives, while for simultaneous search, we examine 1 to 5 alternatives. This difference stems from the increased runtime for simultaneous search. For the dissimilarity threshold τ, we analyze all possible sizes of the overlap, or rather the difference between feature sets. For $k = 5$, this means we consider values of τ from 0.2 to 1 with a step size of 0.2, corresponding to an overlap of four to zero features. For $k = 10$ we consider values of τ from 0.1 to 1 with a step size of 0.1. Naturally, we exclude $\tau = 0$, which would allow returning multiple identical feature sets.

As solver runtime may vary considerably even for optimization problems of the same size, we set a timeout of 60 s on each solver run. If the solver finishes before the timeout, it either found an optimal solution or proved that no solution exists. In our evaluation, these two outcomes apply to 94% of the feature sets. In the remaining cases, the solver had an error, found a feasible but potentially suboptimal solution, or found no solution though one might exist.

3) Prediction: As prediction models, we use decision trees [18] as well as random forests with 100 trees [19]. Both types of models allow to learn complex, non-linear dependencies from the data. We mostly use default hyperparameters of the models, apart from choosing information gain as split criterion and setting a random seed for reproducibility.

Note that tree models carry out feature selection themselves, i.e., they are embedded approaches. Thus, they need not use all features that we provide to them. However, this is no problem for our study. We are interested which performance the models achieve if they are limited to small alternative feature sets, not how they use each feature from an alternative.

C. Evaluation Metrics

We use two kinds of metrics for feature-set quality. First, we evaluate the objective functions of the feature-selection methods, which guided the search for feature sets. Second, we train predictions models with the found feature sets. We report prediction performance in terms of the Matthews correlation coefficient (MCC), which is insensitive to class imbalance. This coefficient reaches its maximum of 1 for perfect predictions, and is 0 for random guessing.

To analyze generalization, we conduct stratified 10-fold cross-validation. Not only model training, but also the search for alternative feature sets is limited to the training data.

D. Datasets

We evaluate alternative feature selection on a variety of datasets from the Penn Machine Learning Benchmarks (PMLB) [10], [11]. To focus the evaluation on one machine-learning task, we only consider binary-classification datasets. However, alternative feature selection is possible for regression and multi-class problems as well. We filter out datasets containing less than 100 instances. Also, we filter out datasets with less than 15 features, to leave some room for alternatives. Next, we remove one dataset with 1000 features, which would dominate the overall runtime of the experiments. Finally, we manually remove datasets that seem to be duplicates or modified versions of other datasets from the benchmark.

In consequence, we obtain 30 datasets. The number of data objects varies from 106 to 9822, and the number of features ranges from 15 to 168. The datasets contain no missing values. Categorical features have an ordinal encoding by default.

E. Implementation

We implement our experiments in Python. For machine learning, we use the package *scikit-learn* [51]. To express and solve the constraint optimization problems, we use the solver *CBC* via the package *OR-Tools* [52].

VI. Evaluation

In this section, we evaluate our experiments. In particular, we discuss results for the different dimensions of our experimental design: datasets, prediction models, feature-selection methods, and approaches to search for alternatives.

A. Datasets

Naturally, feature-set quality depends on the dataset. For example, distribution of feature-set quality in datasets might be relatively uniform or relatively skewed. Datasets with more features n generally allow for more alternative feature sets. At the same time, the (normalized) feature quality $q(X_{\cdot j}, y)$ can be spread over more features than for smaller datasets, making it harder to compose a high-quality feature set.

Indeed, our experiments show a broad variation of feature-set quality over datasets. Further, we do not observe a clear relationship between dimensionality n and feature-set quality.

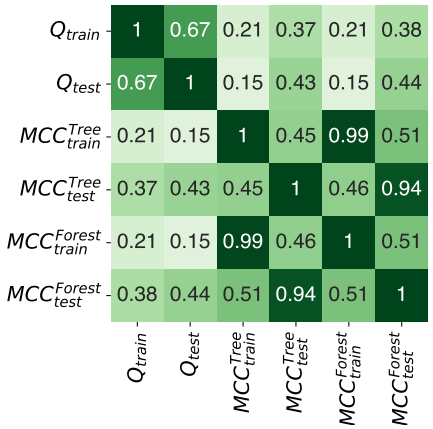


Fig. 2. Correlation between evaluation metrics over all experimental settings.

B. Prediction Models

As one can expect, the average prediction performance of random forests is higher than for decision trees. Also, overfitting occurs for both models, i.e., there is a gap between training-set and test-set prediction performance. As we do not limit the growth of trees or prune them after training, this observation is expected. However, that will not prevent our analysis how prediction performance develops over alternative feature sets, where we normalize feature-sets quality. To a lesser extent, the optimization objective Q also shows overfitting for all feature-selection methods.

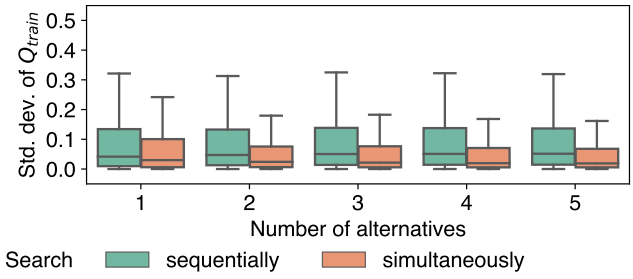
Figure 2 shows the Spearman correlation between different evaluation metrics. As the performance of decision trees and random forests is highly correlated, we focus on decision trees in the following, which always consider all features rather than randomly subsampling them.

Figure 2 also shows that the correlation between training-set quality and test-set quality is moderate, but not high, for the optimization objective Q as well as prediction performance in terms of MCC. This observation might be an effect of overfitting. Further, the correlation between Q and MCC is only weak to moderate. This means that the quality criterion of feature selection is only partially indicative of prediction performance. In particular, only one of our four feature-selection methods uses prediction performance to assess feature sets.

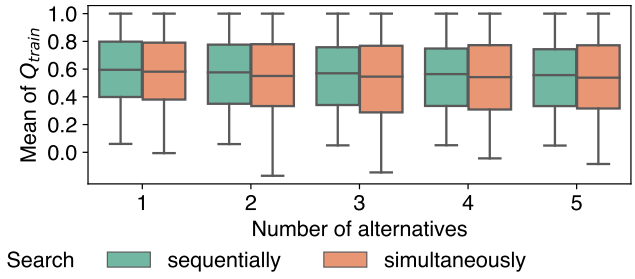
C. Feature-Selection Methods

As the different feature-selection methods have different objective functions, it does not make sense to compare objective values between them. For the optimization status, we note that 89% of feature sets for *FCBF* could not be determined, as the solver timed out or the optimization problem was infeasible. For MI, this figure only is 19%. In particular, the additional constraints in our formulation of *FCBF* (c.f. Equation 8) seemingly made it hard to find valid feature sets.

Regarding test-set prediction performance, *Model Gain* is best on average. The median test-set MCC for decision trees is 0.57 for *Model Gain*, 0.54 for *MI*, and 0.51 for *Greedy Wrapper*. It is not surprising that model-based importance

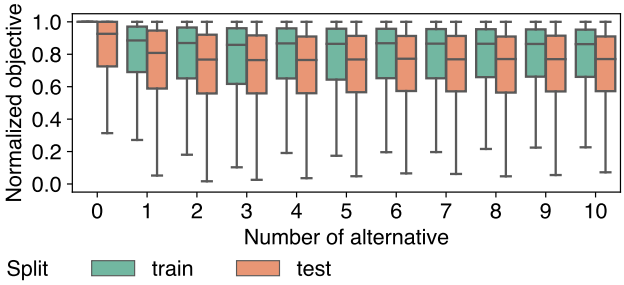


(a) Standard deviation of objective.

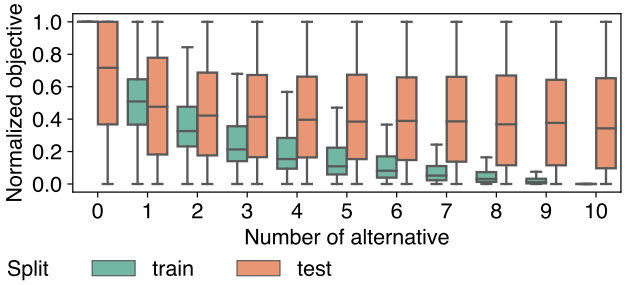


(b) Mean objective.

Fig. 3. Training-set objective value Q in experimental runs of sequential or simultaneous search. Outliers removed for readability reasons.



(a) Max-normalization.



(b) Min-max-normalization.

Fig. 4. Objective value Q, normalized per experimental setting, in sequential search, using *MI* as feature selector. Outliers removed for readability reasons.

yields better feature sets than univariate, model-free feature scoring with *MI*. The relatively bad performance of *Greedy Wrapper* might result from its heuristic nature: The wrapper search can only evaluate a fraction of all feasible feature sets and might get stuck in local optima, while the remaining feature-selection methods optimize globally.

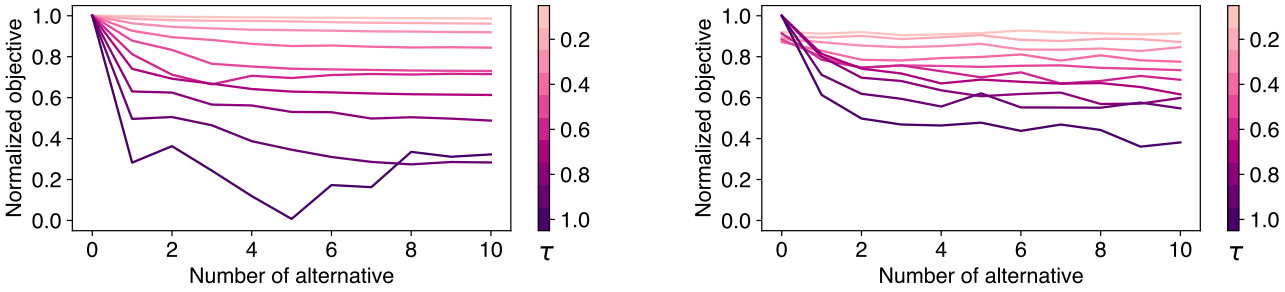
The feature-set size k shows expected effects: Larger feature sets, i.e., $k = 10$, yield higher feature-set quality than smaller feature sets, i.e., $k = 5$. However, doubling the number of features does not double prediction performance. Over all experimental settings, the median test-set MCC of decision trees is 0.50 for $k = 5$ and 0.60 for $k = 10$. This shows that small feature sets can already achieve relatively high prediction performance. For the optimization objective Q, the increase from $k = 5$ to $k = 10$ is higher than for prediction performance, but still less than proportional with k.

D. Searching Alternatives

1) Search Method: As we expected, the training-set objective value Q of feature sets within a particular search run varies more for sequential search than for simultaneous search. Figure 3a visualizes this observation. Also, the variance increases with the number of alternatives for sequential search, but decreases for simultaneous search. I.e., simultaneous search finds relatively homogeneous feature sets, in particular, if many alternatives are desired. However, on the test set, the variance also increases with the number of alternatives for simultaneous search. This effect might be a result of overfitting. Regarding test-set prediction performance, simultaneous search does not show lower variance than sequential search any more.

Surprisingly, the average feature-set quality of simultaneous search is not higher than for sequential search, neither regarding objective value Q nor regarding prediction performance. Figure 3b shows this behavior for the mean training-set objective of search runs. Depending on the search setting, either of the two searches might yield better results. An explanation for this phenomenon is that search results might be sub-optimal. Exact optimization can time out, and *Greedy Wrapper* uses a heuristic search per se. As simultaneous search is harder than sequential search, it is affected more by timeouts. In particular, 11.65% of feature sets from simultaneous search were feasible but potentially sub-optimal, but no feature sets for sequential search. The former figure strongly depends on the number of alternatives: For up to three alternatives in simultaneous search, no or only a few feature sets are potentially sub-optimal, but 4% of the feature sets for four alternatives and 35% of the feature sets for five alternatives. Based on these results, we focus on sequential search in the following.

2) Number of Alternatives: For sequential search, the training-set objective value Q naturally decreases with the number of alternatives, at least for the feature-selection criteria optimized exactly. Figure 4 illustrates this for *MI*-based feature selection. The other feature-selection methods, including the heuristic *Greedy Wrapper*, exhibit similar effects. As feature-set quality varies between datasets and number of features k, we apply normalization: In Figure 4a, we have max-normalized the objective value for each search of alternatives, i.e., the highest objective value is scaled to 1. This figure shows that there are multiple alternatives of similar quality, as the



(a) Training set.

(b) Test set.

Fig. 5. Median objective value Q, max-normalized per experimental setting, in sequential search, using *MI* as feature selector and $k = 10$.

median feature-set quality remains relatively stable over the number of alternatives, and is close to the maximum. Figure 4b uses min-max normalization, i.e., the worst of the alternatives gets 0 as objective. This figure highlights that the training-set objective value decreases most from the original feature set to the first alternative, but less beyond.

Additionally, Figure 4 shows that the test-set objective value drops most to the first alternative as well. However, the decrease in objective value over the alternatives is less visible than on the training set. In particular, alternatives can even have a higher objective value than the original feature set, due to overfitting. Similar findings hold for test-set prediction performance. Thus, the alternative feature sets fulfill their purpose of being different solutions with similar quality.

Be aware that these observations refer to the quality of the found feature sets. However, the more alternatives are desired, the more likely is an infeasible optimization problem. For example, the *MI* feature selector in sequential search always finds an original feature set, but the problem is infeasible in 2% of the cases for the first alternative, 14% for the second alternative, 24% for the fifth alternative, and 30% for the tenth alternative. This outcome also depends on the dissimilarity, threshold τ, which we analyze in the following.

3) Dissimilarity Threshold: As Figure 5 shows for *MI* as feature selector, the decrease of the objective value Q over the number of alternatives can strongly depend on the dissimilarity threshold τ. For a low dissimilarity threshold, e.g., $\tau = 0.1$, the objective value barely drops over the number of alternatives. In contrast, for dissimilarity threshold, e.g, $\tau = 1$, the objective decrease significantly. This phenomenon also holds for test-set objective and test-set prediction performance, though the decrease with τ is less prominent. Also, the the number of infeasible optimization problems, i.e., lack of valid alternative feature sets, increases with τ.

While the previous observation were made for *MI* as feature selector, they do not hold universally. The results for *Model Gain* strongly depend on τ as well, but the picture for *FCBF* and *Greedy Wrapper* is less clear. In case of *Greedy Wrapper*, this can be explained by its heuristic search. In case of *FCBF*, the fraction of infeasible solutions is much higher than for *MI* in general, so adding further constraints has less impact.

VII. Conclusions and Future Work

A. Conclusions

Obtaining interpretable solutions is a highly active area of research in machine learning. One way to foster interpretability is selecting a small set of features for predictions. Traditional feature-selection methods yield *one* feature set with high quality. However, users might be interested in obtaining multiple, sufficiently different feature sets that, at the same time, have high quality. Such alternative feature set might provide alternative explanations for predictions.

In this paper, we defined alternative feature selection as an optimization problem. We formalized alternatives via constraints, independent from the feature-selection method. Further, we presented approaches to solve the optimization problem for different categories of feature selection. Finally, we ran an evaluation with 30 classification datasets and four feature-selection methods. We varied a threshold for how strongly alternatives should differ from existing solutions.

In our experiments, we found that one can obtain alternative feature sets with similar quality as the original ones. This worked with simultaneous search as well as sequential search for alternatives. The latter procedure was significantly faster, and allows stopping the search once one does not want any more alternatives. As expected, the quality of alternatives drops the more alternatives one desires and the more alternatives should differ. The ultimate decision which trade-off between feature-set quality and alternatives is acceptable lies at the user. Thus, we recommend evaluating different dissimilarity thresholds for alternatives.

B. Future Work

While our work introduced alternative feature selection, one might vary several points of our approach. For example, one can integrate other feature-selection methods, in particular, embedded methods. Further, one can vary the notion of alternatives, e.g., set-dissimilarity measure, definition for multiple alternatives, or soft constraints instead of hard constraints. Finally, while our evaluation was domain-independent, the benefit of alternative feature set for explainable predictions should be analyzed in domain-specific case studies.

References

[1] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–45, 2017.

[2] G. Borboudakis and I. Tsamardinos, "Extending greedy feature selection algorithms to multiple solutions," *Data Min. Knowl. Disc.*, vol. 35, no. 4, pp. 1393–1434, 2021.

[3] J. Bailey, "Alternative clustering analysis: A review," in *Data Clustering: Algorithms and Applications*. CRC Press, 2014, ch. 21, pp. 535–550.

[4] U. F. Siddiqi, S. M. Sait, and O. Kaynak, "Genetic algorithm for the mutual information-based feature selection in univariate time series data," *IEEE Access*, vol. 8, pp. 9597–9609, 2020.

[5] Y. Saeys, T. Abeel, and Y. V. d. Peer, "Robust feature selection using ensemble feature selection techniques," in *Proc. ECML PKDD*, 2008, pp. 313–325.

[6] B. Seijo-Pardo, I. Porto-Díaz, V. Bolón-Canedo, and A. Alonso-Betanzos, "Ensemble feature selection: Homogeneous and heterogeneous approaches," *Knowl.-Based Syst.*, vol. 118, pp. 124–139, 2017.

[7] V. Lagani, G. Athineou, A. Farcomeni, M. Tsagris, and I. Tsamardinos, "Feature selection with the R package MXM: Discovering statistically equivalent feature subsets," *J. Stat. Software*, vol. 80, no. 7, pp. 1—-25, 2017.

[8] S. Verma, J. Dickerson, and K. Hines, "Counterfactual explanations for machine learning: A review," arXiv:2010.10596 [cs.LG], 2020.

[9] I. Stepin, J. M. Alonso, A. Catala, and M. Pereira-Fariña, "A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence," *IEEE Access*, vol. 9, pp. 11 974–12 001, 2021.

[10] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, "PMLB: a large benchmark suite for machine learning evaluation and comparison," *Biodata Min.*, vol. 10, 2017.

[11] J. D. Romano, T. T. Le, W. La Cava, J. T. Gregg, D. J. Goldberg, N. L. Ray, P. Chakraborty, D. Himmelstein, W. Fu, and J. H. Moore, "PMLB v1.0: An open source dataset collection for benchmarking machine learning methods," arXiv:2012.00058v3 [cs.LG], 2021.

[12] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, 2014.

[13] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[14] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, University of Waikato, Hamilton, New Zealand, 1999.

[15] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proc. ICML*, 2003, pp. 856–863.

[16] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, 2005.

[17] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, no. 1-2, pp. 273–324, 1997.

[18] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth, 1984.

[19] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[20] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine learning interpretability: A survey on methods and metrics," *Electronics*, vol. 8, no. 8, 2019.

[21] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proc. KDD*, 2016, pp. 1135–1144.

[22] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. NIPS*, 2017.

[23] I. Covert, S. M. Lundberg, and S.-I. Lee, "Understanding global feature contributions with additive importance measures," in *Proc. NeurIPS*, 2020, pp. 17 212–17 223.

[24] L. Egghe, "New relations between similarity measures for vectors based on vector norms," *J. Am. Soc. Inf. Sci. Technol.*, vol. 60, no. 2, pp. 232–239, 2009.

[25] "MOSEK modeling cookbook : Mixed integer optimzation," MOSEK ApS, 2021, accessed: 2022-02-25. [Online]. Available: https://docs.mosek.com/modeling-cookbook/mio.html

[26] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Phys. Rev. E*, vol. 69, no. 6, 2004.

[27] X. V. Nguyen, J. Chan, S. Romano, and J. Bailey, "Effective global approaches for mutual information based feature selection," in *Proc. KDD*, 2014, pp. 512–521.

[28] S. Ermon, C. Gomes, and B. Selman, "Uniform solution sampling using a constraint solver as an oracle," in *Proc. UAI*, 2012, pp. 255–264.

[29] J. White, D. Benavides, D. C. Schmidt, P. Trinidad, B. Dougherty, and A. Ruiz-Cortés, "Automated diagnosis of feature model configurations," *J. Syst. Software*, vol. 83, no. 7, pp. 1094–1107, 2010.

[30] C. Henard, M. Papadakis, M. Harman, and Y. Le Traon, "Combining multi-objective search and constraint solving for configuring large software product lines," in *Proc. ICSE*, 2015, pp. 517–528.

[31] J. Guo and K. Shi, "To preserve or not to preserve invalid solutions in search-based software engineering: A case study in software product lines," in *Pro. ICSE*, 2018, pp. 1027–1038.

[32] E. Müller, I. Assent, S. Günnemann, R. Krieger, and T. Seidl, "Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data," in *Proc. ICDM*, 2009, pp. 377–386.

[33] Y. Guan, J. G. Dy, and M. I. Jordan, "A unified probabilistic model for global and local unsupervised feature selection," in *Proc. ICML*, 2011, pp. 1073–1080.

[34] J. Hu and J. Pei, "Subspace multi-clustering: a review," *Knowl. Inf. Sys.*, vol. 56, no. 2, pp. 257–284, 2018.

[35] H. V. Nguyen, E. Müller, and K. Böhm, "4S: Scalable subspace search scheme overcoming traditional Apriori processing," in *Proc. Big Data*, 2013, pp. 359–367.

[36] H. Trittenbach and K. Böhm, "Dimension-based subspace search for outlier detection," *Int. J. Data Sci. Anal.*, vol. 7, no. 2, pp. 87–101, 2019.

[37] E. Fouché, F. Kalinke, and K. Böhm, "Efficient subspace search in data streams," *Inf. Syst.*, vol. 97, 2021.

[38] C. Emmanouilidis, A. Hunter, J. MacIntyre, and C. Cox, "Selecting features in neurofuzzy modelling by multiobjective genetic algorithms," in *Proc. ICANN*, 1999, pp. 749–754.

[39] I. M. Müller, "Feature selection for energy system modeling: Identification of relevant time series information," *Energy AI*, vol. 4, 2021.

[40] M. van Leeuwen and A. Knobbe, "Diverse subgroup set discovery," *Data Min. Knowl. Disc.*, vol. 25, no. 2, pp. 208–242, 2012.

[41] A. Woznica, P. Nguyen, and A. Kalousis, "Model mining for robust feature selection," in *Proc. KDD*, 2012, pp. 913–921.

[42] K. Liu and J. Tian, "Subspace learning with an archive-based genetic algorithm," in *Proc. IEEM*, 2018, pp. 181–188.

[43] D. S. Guru, M. Suhil, L. N. Raju, and N. V. Kumar, "An alternative framework for univariate filter based feature selection for text categorization," *Pattern Recognit. Lett.*, vol. 103, pp. 23–31, 2018.

[44] P. Paclík, R. P. W. Duin, G. M. P. van Kempen, and R. Kohlus, "On feature selection with measurement cost and grouped features," in *Proc. SSPR /SPR*, 2002, pp. 461–469.

[45] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. R. Stat. Soc.: Ser. B (Stat. Methodol.)*, vol. 68, no. 1, pp. 49–67, 2006.

[46] W. C. Groves, "Toward automating and systematizing the use of domain knowledge in feature selection," Ph.D. dissertation, University of Minnesota, 2015.

[47] K. Mohammadi, A.-H. Karimi, G. Barthe, and I. Valera, "Scaling guarantees for nearest counterfactual explanations," in *Proc. AIES*, 2021, pp. 177–187.

[48] A.-H. Karimi, G. Barthe, B. Balle, and I. Valera, "Model-agnostic counterfactual explanations for consequential decisions," in *Proc. AISTATS*, 2020, pp. 895–905.

[49] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proc. FAT**, 2020, pp. 607–617.

[50] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochim. Biophys. Acta - Protein Struct.*, vol. 405, no. 2, pp. 442–451, 1975.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.

[52] L. Perron and V. Furnon, "OR-Tools," Google, 2022, accessed: 2022-02-25. [Online]. Available: https://developers.google.com/optimization/