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 classification datasets. In our experiments, we find that our approaches allow finding alternative feature sets with similar prediction quality as the original ones.**

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

I. Introduction

a) Motivation: Feature-selection methods are ubiquitous for a variety of reasons. By reducing the dimensionality of the dataset, 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 just 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 multiple, differently composed feature sets with similar quality. For the user, these other feature sets might be interesting alternatives for multiple reasons. First, some of the originally selected features might be costly to obtain, so the user would prefer cheaper alternatives, while maintaining prediction quality. 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 explaining the prediction target, rather than just making good predictions. In such a situation, knowing alternatives allows for formulating multiple hypotheses and broadens the understanding.

b) Problem statement: This article addresses the problem of alternative feature selection, which we informally 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 a similar quality. Depending on how different the alternative feature set should be, one might have to compromise on quality. E.g., if there are only a few highly predictive features and most of them are part of the original feature set, the alternative feature set might have significantly lower prediction 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]. Ensemble feature-selection techniques [5], [6] produce multiple feature sets, but do not focus on optimal alternatives. The same goes for the concept of statistically equivalent feature subsets [7]. 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 with different features, but similar feature-set quality, e.g., prediction performance.

d) Contributions: Our contribution is threefold. First, we define an optimization problem that formalizes alternative feature selection. We foster the search for alternatives via constraints on feature sets. This formulation also allows integrating other constraint types on feature sets [10], [11], [12]. 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) [13], [14] 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].

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 [15], [1] for comprehensive surveys of feature selection. A typical categorization of feature selection is into filter, wrapper, and embedded methods [16].

[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

a) Filter methods: Filter methods evaluate feature sets without training a prediction model. Univariate filters assess each feature on its own, often assigning a numeric score ot each feature, while multivariate filters evaluate feature sets. Examples for univariate filters are the absolute Pearson correlation or the mutual information between a feature and the prediction target. Such methods ignore potential interaction between features, e.g., if they are redundant to each other. Multivariate methods often combine a measure of feature relevance with a measure of feature redundancy. Examples for such a method include CFS [17], FCBF [18], and mRMR [19]. Another interesting filter method is Relief [20], for which multiple extensions exist. While Relief assigns quality to individual features rather than feature sets, it still uses other features indirectly via nearest-neighbor computations between data objects.

b) Wrapper methods: Wrapper methods [21] employ a search strategy over feature sets and evaluate each of these feature set by training a prediction model. Thus, feature-set quality is equal to the prediction quality of the model trained with this feature set. Various black-box optimization techniques can serve as search strategy, e.g., genetic algorithms.

c) Embedded methods: Embedded methods train prediction models with built-in feature selection, e.g., decision trees [22] or random forests [23]. Thus, the criterion to evaluate the quality of features or feature sets is model-specific. For example, tree-based models often use information gain or the Gini index to select features during training.

d) Post-hoc feature-importance methods: Apart from traditional feature selection, there are various methods that assess feature importance after training a model. These methods range range from local explanation methods like LIME [24] or SHAP [25] to global importance methods like permutation importance [23] or SAGE [26]. In particular, assessing feature importance plays a crucial role in the emerging fields of ML interpretability [27].

III. Alternative Feature Selection

In this section, we present the problem and approaches for alternative feature selection. First, we define the structure of the optimization problem, i.e., objective and constraints. 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 being alternative as one or

multiple constraints. Third, one can consider being alternative as objective and introduce a constraint on feature-set quality, e.g., a lower bound. Fourth, one can define constraints for both feature-set quality and being alternative, searching for any feasible solution instead of optimizing. Depending on the use case, any of these four formulations might be appropriate.

In accordance with 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.

B. Constraints – Defining Alternative Feature Sets

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

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 [28]. Most of these measures consider how strongly two sets overlap and set this in relation to the size of the sets. For example, a simple and well-known set-dissimilarity measure is the Jaccard distance. Given two feature sets F_1, F_2, the Jaccard distance is defined as

$$\begin{aligned} d_{Jacc}(F_1, F_2) &= 1 - \frac{|F_1 \cap F_2|}{|F_1 \cup F_2|} \\ &= 1 - \frac{|F_1 \cap F_2|}{|F_1| + |F_2| - |F_1 \cap F_2|} \end{aligned} \tag{2}$$

We leverage such a 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. The choice of τ depends on how alternative the new feature set should be. Also, a feature set that differs strongly might yield a large change in feature-set quality. Thus, we leave τ as a parameter of our approach. If the choice of τ is unclear a priori, users can try out different values and compare results. If the set-dissimilarity measure is normalized to $[0, 1]$, 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 and adapting Definition 2 accordingly:

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

The latter formulation avoids having the feature-set sizes, and thereby the decision variables, in a quotient. However, we can see that the effect of setting τ on the overlap size is non-linear. Thus, we resort to another set overlap measure in this article, the using Dice coefficient:

$$\begin{aligned} d_{Dice}(F_1, F_2) = 1 - \frac{2 \cdot |F_1 \cap F_2|}{|F_1| + |F_2|} \geq \tau \\ \Leftrightarrow |F_1 \cap F_2| \leq \frac{1-\tau}{2} \cdot (|F_1| + |F_2|) \end{aligned} \tag{4}$$

If $|F_1| = |F_2|$, this measure is also identical to several further overlap measures. In that case, τ directly controls which fraction of features in one set needs to differ in the other set, and vice versa:

$$\begin{aligned} \text{If } |F_1| = |F_2| : \quad d_{Dice}(F_1, F_2) \geq \tau \\ \Leftrightarrow |F_1 \cap F_2| \leq (1-\tau) \cdot |F_1| = (1-\tau) \cdot |F_2| \end{aligned} \tag{5}$$

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:

$$\begin{aligned} |F_s| &= \sum_{j=1}^{n} s_j \\ |F_{s_1} \cap F_{s_2}| &= \sum_{j=1}^{n} s_{1,j} \cdot s_{2,j} \end{aligned} \tag{6}$$

Combining Equation 3 or Equation 4 with Equation 6 yields an inequality only involving sums and products of the binary decision variables and constant values. In fact, one can replace the product $s_{1,j} \cdot s_{2,j}$ with linear constraints by introducing an auxiliary variable t_j [29]:

$$\begin{aligned} t_j &\leq s_{1,j} \\ t_j &\leq s_{2,j} \\ 1 + t_j &\geq s_{1,j} + s_{2,j} \\ t_j &\in \{0, 1\} \end{aligned} \tag{7}$$

Thus, one can express alternative features sets according to Definition 2 with 0-1 integer linear constraints. This simple constraint type allows for using a broad range of solvers, given the right objective function, which we discuss in Section III-C. If there is an existing feature set, i.e., one either knows s_1 or s_2, Equation 6 already is linear without applying Equation 7.

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, the user gets several alternatives iteratively. In each iteration, we determine one alternative feature set. 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 effectively just duplicates the constraint from the base case and instantiates it for different existing feature sets. The objective function remains the same as in the base case, i.e., we optimize the quality of the new feature set F_2. This means the number of variables in the optimization problem remains constant, independent from the number of alternatives. Each alternative only adds one new constraint to the optimization problem. As we are alway comparing a new, variable feature set to existing feature sets, we also do not need to introduce interaction variables as in Equation 7. Thus, we expect the runtime of sequential search to scale well with the number of alternatives.

As the solution space becomes narrower over iterations, feature-set quality can drop with each further alternative. The user can decide after each iteration if feature-set quality is too low or if another alternative should be found. Also, a solver for this problem might benefit runtime-wise from iteratively adding constraints, provided the solver keeps some internal state between iterations and can warm-start.

Instead of Definition 3, one could also use a less strict constraint, e.g., requiring only the average dissimilarity to all existing feature sets to pass the threshold. We do not consider this case in our article.

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 average quality of all feature sets as objective. The number of decision variables for feature selection increases linearly with the number of alternatives. Also, for each feature and each pair of alternatives, we need to introduce an interaction variable according to Equation 7. The number of constraints grows quadratically in the number of alternatives as well. Thus, we expect the runtime of simultaneous search to scale worse with the number of alternatives than of sequential search.

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 be better in terms of 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 simultaneous 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 an appropriate white-box solver. In Section III-B, we already showed that our notion of alternative feature sets results in 0-1 integer linear constraints. In this section, we present white-box optimization objectives for different feature-selection methods.

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:

$$Q_{uni}(s, X, y) = \sum_{j=1}^{n} s_j \cdot q(X_{\cdot j}, y) \tag{8}$$

Here, q typically is a bivariate dependency measure, e.g., mutual information [30] or absolute value of Pearson 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. For example, one can pre-compute permutation importance or SAGE scores for each feature and use them in Equation 8. However, note that 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 8 does not hold. In theory, one would have to re-calculate feature importance for each feature set. However, such a procedure makes a white-box approach infeasible. In practice, one can still use Equation 8 with importance scores only computed on the full dataset. One only needs to be aware that such an approach might not represent feature importances in feature subsets faithfully.

c) Multivariate filter feature selection: Several multivariate filter methods allow for a simple white-box formulation as well, though not necessarily a linear one. In the following, we discuss CFS, mRMR, FCBF, and Relief.

d) CFS: Correlation-based Feature Selection (CFS) [17] considers both relevance and redundancy of features. Relevance is the correlation between features and prediction target, similar to the univariate filter. Redundancy is the correlation

between features, acting as a normalization term. Using a bivariate dependency measure q to quantify correlation, the objective is as follows:

$$Q_{CFS}(s, X, y) = \frac{\sum_{j=1}^{n} s_j \cdot q(X_{\cdot j}, y)}{\sqrt{\sum_{j=1}^{n} s_j + \sum_{j_1=1}^{n} \sum_{\substack{j_2=1 \\ j_2 \neq j_1}}^{n} s_{j_1} \cdot s_{j_2} \cdot q(X_{\cdot j_1}, X_{\cdot j_2})}} \quad (9)$$

e) mRMR: Minimal Redundancy Maximum Relevance (mRMR) [19] follows a similar principle as CFS, but uses the difference instead of the ratio between relevance and redundancy:

$$Q_{mRMR}(s, X, y) = \frac{\sum_{j=1}^{n} s_j \cdot q(X_{\cdot j}, y)}{\sum_{j=1}^{n} s_j} - \frac{\sum_{j_1=1}^{n} \sum_{j_2=1}^{n} s_{j_1} \cdot s_{j_2} \cdot q(X_{\cdot j_1}, X_{\cdot j_2})}{(\sum_{j=1}^{n} s_j)^2} \quad (10)$$

Note that if one knows the feature-set size, he denominators are constant and thus, Equation 10 is linear if one replaces the product terms according to Equation 7. Without this replacement, it is a quadratic-programming problem [31], [32].

f) FCBF: The Fast Correlation-Based Filter (FCBF) [18] 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:

$$\begin{aligned} \max_{s} \quad & Q_{FCBF}(s, X, y) = \sum_{j=1}^{n} s_j \cdot q(X_{\cdot j}, y) \\ \text{subject to:} \quad & \forall (j_1, j_2) \in \{1, \ldots, n\}^2, j_1 \neq j_2 : \\ & s_{j_1} \cdot s_{j_2} \cdot q(X_{\cdot j_1}, X_{\cdot j_2}) < q(X_{\cdot j_1}, y) \end{aligned} \quad (11)$$

In particular, we drop the threshold parameter on feature-target correlation and rather maximize the latter, as in the univariate filter case. To limit the size of the resulting feature set, one can put a constraint on the feature set size. Also, if one still wants to definitely prevent selection of features with low correlation to the prediction target, one can filter out these features before optimization. With the constraint in Equation 11, 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.

g) Relief: Relief [20] assigns a score to each features by sampling data objects and considering the difference in feature values compared to their nearest neighbors. The idea is that data objects with a similar value of the prediction target should have similar feature values. In contrast, data objects that differ in their prediction target should differ in their feature values as well. We consider this method multivariate as nearest-neighbor computations involve all features instead of considering features independently. However, the resulting scores for each feature can be put into the univariate objective from Equation 8. Also, one can combine Relief with CFS to consider feature redundancy [17], which the default Relief does not.

2) Black-Box Optimization: If feature-set quality has not simple closed-form expression, one needs to 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 iterating over candidate feature sets in a systematic manner. 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.

a) Enumerating feature sets: One can relinquish the idea of a search heuristic and just enumerate all feature sets that fulfill the constraints for being alternative. This most naive idea is iterating over all feature sets and sorting out those not alternative enough. A bit more focused is using a solver to enumerate all valid alternatives. While such an approach is technically simple, it might very inefficient, as there can be a huge number of alternative feature sets.

b) Sampling feature sets: Instead of considering all possible alternatives, one can also sample a limited number of them. Again, the naive way is sampling from all feature sets, but removing those samples that are not alternative enough. If the number of valid alternatives is low to the total number of feature sets, this approach might need a lot of samples. In fact, uniform sampling from a constrained space is a computationally hard problem, believed to be harder than only determining if a valid solution exists or not [33].

c) Multi-objective optimization: If one considers alternative feature selection as multi-objective problem, as mentioned in Section III-A, there are no hard constraints any more. Given a suitable multi-objective search procedure, the decision on trading off being alternative against feature-set quality might be postponed till after the search.

d) Adapting search: One can adapt a search heuristic to consider the constraints for being alternative. 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 [34], [35], [36].

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 [21], our search procedure only proposes and evaluates feature sets that satisfy the constraints for alternatives. First, the algorithm uses a solver to find one solution that is alternative enough, given the current constraints. 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 one feature, i.e., selecting the

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        // No valid alternative
    return ∅
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
     // Swap if improved
        s ← s'
        j ← 1
    else                  // Try next feature
        j ← j + 1
return s
```

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

feature if it was deselected or deselecting it if it was selected. The algorithm applies the solver again to find a solution s' containing this change and satisfying the other constraints. If such a solution exists and its quality $Q(s', X, y)$ improves on the current solution, the algorithm continuous from the new solution. Else, it attempts to swap the next feature. The algorithm terminates if no swap of a feature leads to an improvement or a fixed amount of iterations max_iters is reached. We define the iteration count as the number of calls to the solver, i.e., attempts to generate feature sets, as this step might be costly. 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 rather specific to certain prediction models. Thus, we leave formulation of concrete approaches open for future work. For example, in decision trees, one could prevent the tree from splitting on certain features if the resulting feature set was too similar to a feature set of a previously trained tree.

IV. Related Work

In this section, we review related work from four areas that are closely or loosely related to alternative feature selection: alternative clustering, subspace clustering and 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. Nevertheless, the concrete problem definition for a clustering scenario and our feature-selection scenario is different. [3] also draws the distinction between singular alternatives and multiple alternatives, found sequentially or simultaneously. Besides constraint-based search for alternatives, they discuss other solution paradigms as well. For example, feature selection and data transformation can help to find alternative clusterings [37].

Two examples of constrained-based alternative clustering are *COALA* [38] and *MAXIMUS* [39]. COALA [38] is a hierarchical algorithm that imposes *cannot-link constraints* on pairs of data objects: Data objects assigned to the same cluster in the original clustering should be assigned to different clusters in the alternative clustering. If observing the constraints is infeasible or violates a quality threshold, the algorithms proceeds like unconstrained hierarchical clustering. MAXIMUS [39] employs an integer program to formulates dissimilarity between clusterings, building on the distributions of feature values in clusters. The solutions of this problem are not concrete alternative clusterings, but constrain a subsequent clustering procedure.

b) Subspace clustering and subspace search: Finding multiple useful feature set plays a role in subspace clustering [40], [41], [42] and subspace search in general [43], [44], [45]. These fields 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 of the existing subspace approaches only consider if features are useful in their subspace, but do not consider the relationship between subspaces. In contrast, other approaches explicitly try to remove redundancy between subspaces [40], [43] or foster subspace diversity [44], [45], 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: In the field of feature selection, most methods only yield one solution [2], though there are some exceptions. As an example, [4] adapts a genetic algorithm to ensure diversity of feature sets in the population. To be concrete, they modify the fitness criterion to penalize overlap of candidate feature sets in each iteration. [46] uses multi-objective genetic algorithms to obtain prediction models of different complexity, i.e., using feature sets of different sizes. [47] clusters features based on their values and then forms all combinations of picking one feature from each cluster. They do this as a pre-processing step to reduce the number of features,

not as a guided search for alternative feature sets.

[48] presents six strategies to foster diversity in subgroup set discovery, which attempts to find regions in the feature space where the distribution of the target variable is different from the rest of the data. They integrate these strategies into beam search, i.e, a heuristic search procedure, while we also consider exact solutions. Also, the selection space is different to ours, as subgroup set discovery does not only select features, but also subsets of the features' values.

d) Ensemble feature selection: There are techniques for ensemble feature selection [5], [6], which run feature selection on different versions, e.g., samples, of the data, or run different feature-selection techniques. While fostering diverse feature sets might be a sub-goal to improve prediction performance, the results of the different feature selectors are combined at the end. Thus, this focus is different from our target of finding optimal alternative feature sets.

[49] uses k-medoid clustering and frequent-itemset mining to reduce a set of feature-selection results to a smaller, yet diverse subset. To obtain the initial set of feature sets, the run a feature-selection method on multiple bootstrap samples of the original dataset. [50] builds an ensemble prediction model from classifiers trained on different feature sets and uses an evolutionary algorithm that ensures diversity of feature sets. [51] 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 individual features or feature sets can be used interchangeably to predict a target variable. For example, one can test whether a feature is independent from the target given another feature. A search algorithm combines conducts multiple such tests. The output either is a set of equivalent feature sets or a feature grouping structure that allows to form such non-redundant feature sets.

While equivalent feature sets are related to our notion of alternative feature sets, there are differences as well. We do not require equivalence based on statistical tests. Depending on how the test are configured, such an approach might yield an arbitrary number of alternatives. I.e., it is not straightforward to build an optimal solution for the alternative feature-selection problem from the output of an algorithm for finding statistically equivalent feature sets. Instead, we always provide a fixed number of alternatives in the form of feature sets, unless the problem is infeasible. These alternatives do not need to be equivalent in terms of their quality or another criterion, but should be optimal under constraints. Our dissimilarity threshold allows to control overlap between feature sets, instead of eliminating all feature redundancies. Nevertheless, if one does definitely not want certain redundant features or feature sets to be selected simultaneously, one can achieve this with additional constraints in our optimization problem as well.

f) Constrained feature selection: Our article is not the first to formulate constrained feature-selection problems. In fact, there is work on considering various kinds of constraints in feature selection, e.g. cost constraints [10], group constraints [11], or constraints from domain knowledge [12]. These approaches are orthogonal to our work, as such constraints can be added to our optimization problem as well.

g) Counterfactual explanations: The problem of alternative feature selection shares some similarities with the concept of counterfactual explanations for predictions [8], [9]. Counterfactual explanations provide alternative solutions as well, working with the features of the datasets. Also, the search for counterfactual might be formulated as optimization problem, e.g., as an integer problem [52] or via Satisfiability Modulo Theories (SMT) [53]. [54] presents an approach to find multiple, diverse counterfactual explanations for differentiable prediction models.

However, there are crucial differences between counterfactuals and alternative feature sets: Counterfactual explanations target at predictions on single data objects and the feature values relevant for the prediction, rather than predictive quality of features on the whole dataset. 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. Approaches

Our experimental design comprises multiple approaches for feature selection, defining alternatives, and making predictions.

1) Feature Selection (Objective Functions): We search for alternatives under different notions of feature-set quality as objective function. To represent a diverse set of feature-selection methods, we consider the different categories from Section II-B. We choose four feature selection-methods that are both 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. For the reason explained in Section III-C3, we do not include an embedded method for feature selection.

For each feature-selection method, we set the number of selected features to $k \in \{5, 10\}$. This yields small feature sets, fostering interpretability. As we use a solver to find feature sets anyway, we can enforce the desired k with a simple constraint. However, 0.03% of the feature sets from our evaluation violate their prescribed k for unknown reasons, i.e., the solver seemingly did not enforce the cardinality constraint properly. These few outliers originate from a variety of experimental settings, so we could not detect a cause for this behavior. As a consequence, we remove the affected search runs of alternatives completely, which constitute 0.12% of all search runs.

a) MI: As univariate filter method, we consider mutual information [30] between each feature and the prediction target. This measure allows to capture arbitrary dependencies rather than, say, just linear correlations. We add up the univariate feature qualities according to Equation 8. 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 constraint formulation of FCBF [18] according to Equation 11. 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. Like all wrapper methods, the search algorithm uses the performance of a prediction model to evaluate feature-set quality. In particular, we choose a decision tree as model and Matthews correlation coefficient (MCC) [55] 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*. For the decision-tree model we use, 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 8, i.e., treat the importances like univariate filter scores. Note that the interpretation is different, though. In univariate filters, the scores consider features independently. Here, the scores originate from trees involving multiple features and hierarchical splits on these features. The model-based feature 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 larger size of the optimization problem for simultaneous compared to sequential search, i.e., more decision variables and more constraints. 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 might have found an optimal solution, or might have proven that no solution exists. In our evaluation, these two outcomes apply to 94% of the feature sets. In the remaining cases, the solver might have had an error, found a feasible but suboptimal solution, or found no solution though one might exist.

3) Prediction: As prediction models, we use decision trees [22] as well as random forests with 100 trees [23]. Both types of models allow to learn complex, non-linear dependencies from the data. We leave the hyperparameters of the models mostly at their defaults, apart from two changes: First, we choose information gain instead of Gini impurity as split criterion, to be consistent to our filter feature-selection methods. Second, we specify a random seed for reproducibility reasons.

Note that tree models also carry out feature selection themselves, i.e., they are embedded approaches. Thus, they might 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

To analyze how well feature-selection and prediction models generalize, 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. When we report quality metrics in the following, we focus on test-data results.

We use two kinds of metrics for feature-set quality. We focus on on how these metrics evolve for different configurations of alternative feature selection, cf. Section V-B2. First, we evaluate the objective functions of the individual 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.

D. Datasets

We evaluate alternative feature selection on a variety of datasets from the Penn Machine Learning Benchmarks (PMLB) [13], [14]. To focus the evaluation on one machine-learning task type, 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

to search for alternatives. Next, we remove one dataset with 1000 features, as this dataset would dominate the runtime of the experiments compared to the other datasets, which all have significantly lower dimensionality. 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 experimental pipeline in Python. For machine learning, we use *scikit-learn* [56]. To express and solve the constraint optimization problems, we use *Python-MIP* [57].

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 datasets used, and several effects could occur. 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.

In our experiments, we indeed see a broad variation of feature-set quality over datasets. Further, we do not observe a systematic relationship between dataset dimensionality n and feature-set quality.

B. Prediction Models

As one can expect, the average prediction performance of random forests is higher than that of 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 on training set as well as test set, we only focus on one prediction model in the following: We choose decision trees, as they always consider all features during training, while random forests involve random sampling of features.

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 might be caused by different degrees of overfitting, depending on the experimental settings. Further, the correlation between optimization objective Q and prediction 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.

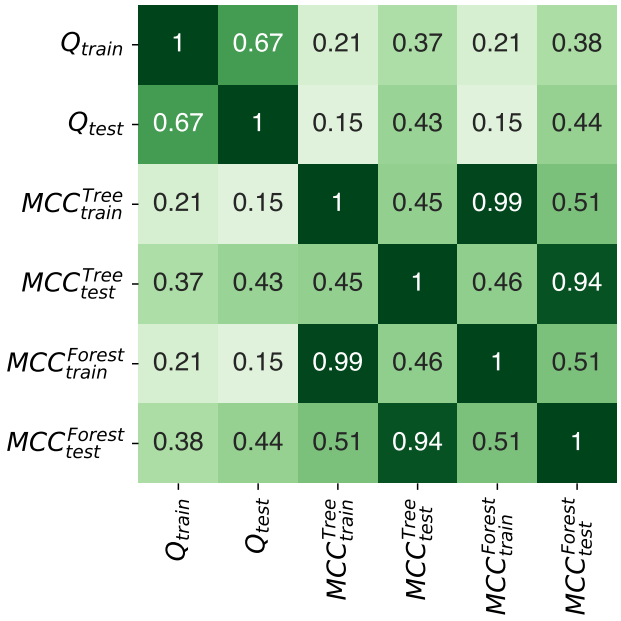


Fig. 2. Correlation between evaluation metrics over all experimental settings.

C. Feature-Selection Methods

As the different feature-selection methods have different functions as optimization objectives Q, it does not make sense to compare objective values between feature-selection methods. Regarding 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%. Thus, besides the constraints for alternatives, the constraints in our formulation of *FCBF* (c.f. Equation 11) made it hard to find valid feature sets. This point needs to be considered when applying our *FCBF* feature selector.

Regarding test-set prediction performance, *Model Gain* beats *MI* and *Greedy Wrapper* 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 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 the expected effect: Larger feature sets, i.e., $k = 10$, exhibit higher feature-set quality than smaller feature sets, i.e., $k = 5$. However, doubling the number of features does not double the prediction performance. Over all experimental settings, the median test-set MCC for decision trees is 0.50 for $k = 5$ and 0.60 for $k = 10$. This displays that small feature sets can already achieve a large share of the possible 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 VI-D1 visualizes this observation. Also, this variation 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. On the test set, the objective value Q of simultaneous-search feature sets still varies less than for sequential search, but increases with the number of alternatives. 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 VI-D1 shows this behavior for the mean training-set objective of search runs. Depending on the concrete search setting, either of the two search types might yield better-performing feature sets. An explanation for this phenomenon is that search does not always yield the exact optimum. For *Greedy Wrapper*, the search is heuristic per se. Additionally, the exact optimization of the other three feature-selection methods can time out. As simultaneous search is harder than sequential search, it is affected stronger by this phenomenon. In particular, 11.65% of feature sets from simultaneous search might not be optimal, as the solver timed out. This figure strongly depends on the number of alternatives: While there are no or only a few such potentially sub-optimal feature sets for up to three alternatives in simultaneous search, 4% of the feature sets might be suboptimal for four alternatives and 35% of the feature sets for five alternatives. For sequential search, we do not have any feasible but sub-optimal solution.

Based on these results, we focus on sequential search in the following.

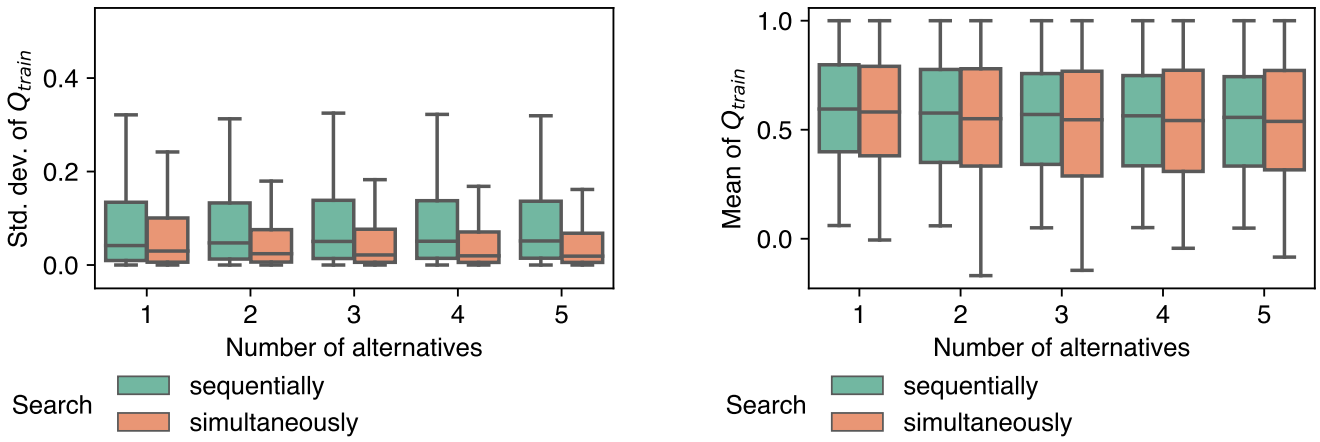
2) Number of Alternatives: For sequential search, the training-set objective value Q naturally decreases with the number of alternatives, at least for the feature-selection criteria optimized exactly. Figure 4 illustrates this for *MI*-based feature selection. The other feature-selection methods, including the heuristic *Greedy Wrapper*, exhibit similar effects. As feature-set quality varies between datasets and number of features k, we apply normalization: In Figure VI-D2, we have max-normalized the objective value for each search of alternatives, i.e., one feature set from a search run gets an objective of 1 and the other feature-set qualities are scaled accordingly. This figure shows that there might be multiple alternatives of similar quality, as the median feature-set quality remains relatively stable over the number of alternatives, and is close to the maximum of 1. Figure VI-D2 uses min-max normalization, i.e., the worst of the alternatives gets an objective of 0. This figure highlights that the decrease training-set objective value is most prominent from the original feature set to the first alternative, but becomes smaller beyond.

Figure 4 also shows that test-set objective value drops the most from the original feature set to the first alternative as well. The decrease in median 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 of the optimization procedure. Similar findings hold for test-set prediction performance. Thus, the alternative feature sets fulfill their purpose of being different solutions with similar quality.

However, one should note that these observations refer to the quality of the feature sets actually found. In particular, the more alternatives are desired, the more likely it is that the optimization problem becomes infeasible. For example, the *MI* feature selector in sequential search always returns a solution for the original feature set, but the problem is infeasible in 2% of the cases for the first alternative, 14% of the cases for the second alternative, 24% of the cases for the fifth alternative, and 30% of the cases for the tenth alternative. Thus, while the quality of feature-sets remains relatively stable with increased number of alternatives, valid alternatives might simply not exist. This outcome 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. If the dissimilarity requirement is low, e.g., $\tau = 0.1$, the objective value barely drops over the number of alternatives. In contrast, if the dissimilarity threshold is be high, 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 for prediction performance. Also, the the number of infeasible optimization problems increases with tau: For a higher dissimilarity threshold, the likelihood is higher that there is no feature set that is alternative enough.

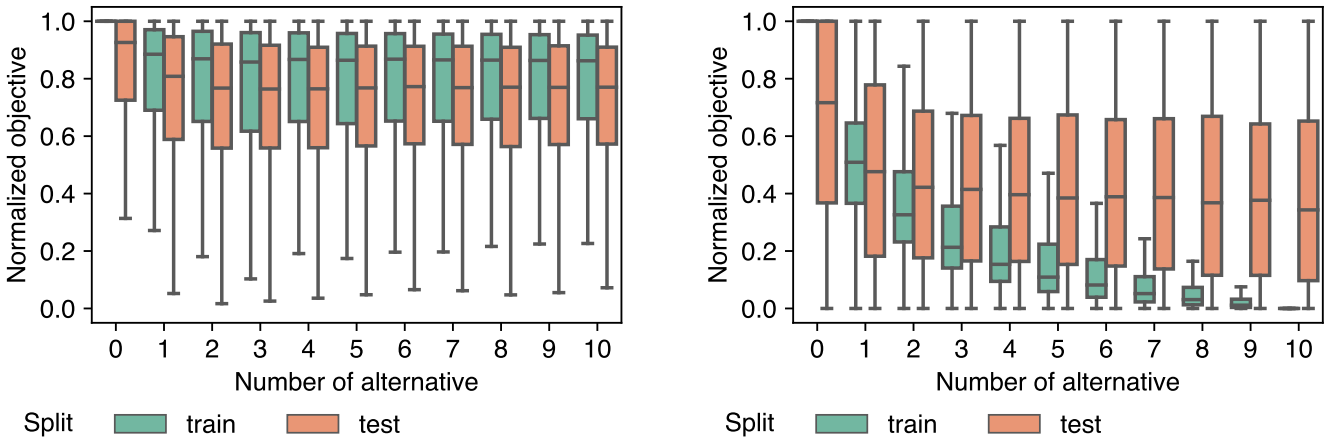
While the previous observation were made for *MI* as feature selector, they do not hold universally. Besides *MI*, the results for *Model Gain* strongly depend on *tau* as well. In contrast, *FCBF* and *Greedy Wrapper* exhibit less influence of τ on feature-set quality. In case of *Greedy Wrapper*, this can be explained by its heuristic search procedure. In case of *FCBF*, the fraction of infeasible solutions is much higher than for *MI*



(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. Development of objective value Q, normalized per experimental setting, over the alternatives in sequential search, using *MI* as feature selector. Outliers removed for readability reasons.

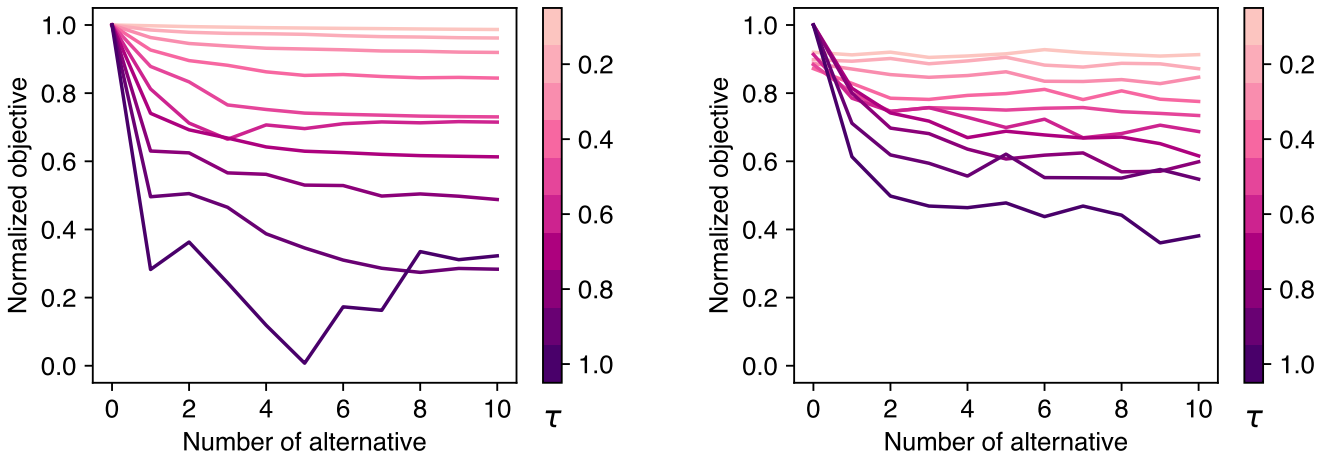
in general, so adding further constraints has less impact.

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 prediction. Traditional feature-selection algorithms focus on yielding *one* feature set with high quality, e.g., regarding prediction performance. 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 from the data.

In this paper, we leveraged constraints to find alternative feature sets that ideally should have similar quality. We introduced a general, formal notion of alternatives, which is independent from the concrete feature-selection method. Additionally, it is combinable with other domain-independent or domain-specific constraints on feature selection. Further, we discussed how different categories of feature-selection methods can integrate this notion. In particular, we presented exact and heuristic approaches for the constrained optimization problem of alternative feature selection. Finally, we evaluated our approach with 30 classification datasets and four feature-selection techniques. We compared two search strategies for alternatives and varied a threshold how strongly alternatives should differ from existing solutions.



(a) Training set.

(b) Test set.

Fig. 5. Development of median objective value Q, max-normalized per experimental setting, over the alternatives and dissimilarity threshold τ in sequential search, using *MI* as feature selector and $k = 10$.

In our experiments, we found that one can indeed obtain alternative feature sets with similar quality as the original one. This worked with simultaneous search for multiple alternatives as well as sequential search for single alternatives. The latter search procedure turned out to be significantly faster, and allows stopping the search once one does not want any more alternatives. As one could expect, the quality of alternatives drops the more alternatives one desires and the more alternatives should differ from existing feature sets. 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 and making a use-case-specific decision.

B. Future Work

While our work introduced alternative feature selection, one might vary several points of our approach. For example, one can combine the search for alternatives with other feature-selection methods. For embedded feature selection, one would need to customize our search routines and push them into model training. Further, one can analyze different notions of alternatives, e.g., other dissimilarity measures for feature sets and other definitions for the case of multiple alternatives. Additionally, one can express alternatives differently than with hard constraints. Instead, one may use soft constraints or a multi-objective formulation of the optimization problem. 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. [Online]. Available: https://arxiv.org/abs/2010.10596

[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] 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.

[11] 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.

[12] W. C. Groves, "Toward automating and systematizing the use of domain knowledge in feature selection," Ph.D. dissertation, University of Minnesota, 2015. [Online]. Available: https://hdl.handle.net/11299/175444

[13] 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.

[14] 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. [Online]. Available: https://arxiv.org/abs/2012.00058v3

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

[16] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, no. Mar, pp. 1157–1182, 2003. [Online]. Available: https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf

[17] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, University of Waikato, Hamilton, New Zealand, 1999. [Online]. Available: https://www.cs.waikato.ac.nz/~ml/publications/1999/99MH-Thesis.pdf

[18] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proc. ICML*, 2003, pp. 856–863. [Online]. Available: https://aaai.org/Papers/ICML/2003/ICML03-111.pdf

[19] 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.

[20] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *Proc. ML*, 1992, pp. 129–134. [Online]. Available: https://www.aaai.org/Papers/AAAI/1992/AAAI92-020.pdf

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

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

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

[24] 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.

[25] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. NIPS*, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[26] 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. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/c7bf0b7c1a86d5eb3be2c722cf2cf746-Paper.pdf

[27] 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.

[28] 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.

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

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

[31] I. Rodriguez-Lujan, R. Huerta, C. Elkan, and C. S. Cruz, "Quadratic programming feature selection," *J. Mach. Learn. Res.*, vol. 11, no. 49, pp. 1491–1516, 2010. [Online]. Available: http://jmlr.org/papers/v11/rodriguez-lujan10a.html

[32] 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.

[33] S. Ermon, C. Gomes, and B. Selman, "Uniform solution sampling using a constraint solver as an oracle," in *Proc. UAI*, 2012, pp. 255–264. [Online]. Available: https://www.auai.org/uai2012/papers/160.pdf

[34] 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.

[35] 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.

[36] 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.

[37] V. T. Tao and J. Lee, "A novel approach for finding alternative clusterings using feature selection," in *Proc. DASFAA*, 2012, pp. 482–493.

[38] E. Bae and J. Bailey, "COALA: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity," in *Proc. ICDM*, 2006, pp. 53–62.

[39] E. Bae, J. Bailey, and G. Dong, "A clustering comparison measure using density profiles and its application to the discovery of alternate clusterings," *Data Min. Knowl. Disc.*, vol. 21, no. 3, pp. 427–471, 2010.

[40] 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.

[41] 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. [Online]. Available: https://icml.cc/Conferences/2011/papers/546_icmlpaper.pdf

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

[43] 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.

[44] 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.

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

[46] 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.

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

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

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

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

[51] 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.

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

[53] A.-H. Karimi, G. Barthe, B. Balle, and I. Valera, "Model-agnostic counterfactual explanations for consequential decisions," in *Proc. AISTATS*, 2020, pp. 895–905. [Online]. Available: https://proceedings.mlr.press/v108/karimi20a.html

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

[55] 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.

[56] 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. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[57] H. G. Santos and T. A. M. Toffolo, "Python-MIP," 2022, accessed: 2022-03-11. [Online]. Available: https://python-mip.com/