Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Reasoning, Learning & Decision-Making Track

LAD-based Feature Selection for Optimal Decision Trees and Other Classifiers

David Ing , Said Jabbour , Lakhdar Sais , Fabien Delorme
CRIL, CNRS - Université d'Artois
{ing, jabbour, sais}@cril.fr, fabien.delorme@cnrs.fr

Abstract

The curse of dimensionality presents a significant challenge in data mining, pattern recognition, computer vision, and machine learning applications. Feature selection is a primary approach to address this challenge. It aims to eliminate irrelevant and redundant features while preserving the relevant ones to reduce computation time, improve prediction performance, and enhance the understanding of data. In this paper, we introduce a new feature selection (FS) technique based on the Logical Analysis of Data (LAD), a pattern learning framework that combines optimization, Boolean functions, and combinatorial theory. One of its main objectives is to generate minimal support sets of features (subsets of features) that discriminate between different groups of data. To generate such subsets, we first reduce the complexity of the LAD optimization task by transforming it into the problem of enumerating minimal hitting sets in a hypergraph, for which efficient implementations exist. Those feature subsets are then ranked based on a scoring method before selecting the highest quality one. Moreover, we explore the relationship between optimal Decision Trees (DTs) and LAD-based FS, introducing new optimality criteria, namely DTs involving a minimum number of features. Finally, we conduct comparative evaluations of LAD-based approach against several state-of-the-art (SOTA) FS methods on benchmark datasets, including two-class binary datasets and numerical datasets with two and multiple classes. Experiments reveal that our approach is competitive with SOTA methods, selecting high-quality feature subsets that maintain or enhance the performance of DTs and other classifiers like SVM, KNN, and Naive Bayes.

1 Introduction

The advancement of high-throughput technologies in this era has led to a rapid increase in the volume of harvested data with respect to both dimensionality and sample size. Managing these data efficiently and effectively poses an increasing challenge, making traditional manual management impractical. Consequently, data mining and machine learning techniques have been developed to automatically discover knowledge within this data. However, this collected data is frequently associated with a high level of noise due to imperfections in the technologies that collect such data and the sources of the data themselves. Unsurprisingly, extracting useful knowledge and patterns from such vast and noisy data is a challenging task, which can lead to the curse of dimensionality. Dimensionality reduction, one of the most popular techniques to remove irrelevant and redundant features, has received significant attention since the performance of machine learning algorithms can be noticeably affected by the number of features and dimensions of data. Dimensionality reduction techniques can be categorized primarily into feature extraction and feature selection (Khalid, Khalil, and Nasreen 2014). Feature extraction techniques involve projecting features into a new feature space of reduced dimensionality, typically creating new features that are combinations of the original features. On the contrary, feature selection approaches aim to find a small subset of most discriminative information that carries the main information of high-dimensional data. This aim can be achieved by removing irrelevant and redundant features while maintaining the most relevant features to the target such as the class labels in the classification problems. Both reduction techniques can improve learning algorithm performance, decrease computational complexity, enhance model generalization, and reduce storage requirements. Nevertheless, feature selection excels in readability and interpretability by selecting a feature subset from the original set of features without any transformation, thus preserving the meaning of the original features. In contrast, feature extraction maps the original feature space to a new feature space with lower dimensions. Yet, linking the features from original feature space to new features is difficult, making analysis of new features problematic due to the lack of real meaning in the transformed features. Therefore, feature selection has been widely used across various domains, including medical for disease diagnosis (Makimoto et al. 2023), oncology (Sayed et al. 2019), agriculture (Buyrukoğlu 2021), and more. Previous or classical FS approaches have focused on statistics (Hall 2000), similarity (Robnik-Sikonja and Kononenko 2003), correlation (Guyon and Elisseeff 2003), information theory such as mutual information (Kraskov, Stögbauer, and Grassberger 2004; Ross 2014), and other techniques to rank features according to their relevance to the target. However, feature selection technique based on logic has received almost no attention.

In this paper, we introduce a new feature selection (FS) approach based on LAD. The primary aspects of LAD involve identifying minimum support sets of features (subsets of features) necessary for explaining observations and uncovering hidden data patterns that differentiate observations among various groups of data. Our contributions focus on

three important aspects: (1). To generate subsets of features, we first mitigate the complexity of such LAD optimization problem by proposing a reduction to the problem of enumerating minimal hitting sets in a hypergraph for which efficient implementations exist. Then, those feature subsets are ranked based on a scoring method to select the highest quality one. (2). We highlight the relationship between Decision Trees (DTs) and LAD-based FS enabling us to define a new optimality criteria, focusing on the minimum number of features sufficient for constructing DTs, and (3). We conduct comparative evaluations between LAD-based and several state-of-the-art (SOTA) FS approaches on benchmark datasets, including two-class binary datasets and numerical datasets with two and multiple classes. Experiments reveal that our proposed approach is competitive with SOTA methods by selecting high-quality feature subsets that maintain or enhance the performance of DTs and other classifiers like SVM, KNN, and Naive Bayes.

2 Related Work

2.1 Feature Selection Techniques

Feature Selection (FS) is one of the main approaches for dimensionality reduction, with the aim of selecting a small subset of relevant features from the original dataset, thereby eliminating noisy (irrelevant) and redundant features. Such process typically results in enhanced learning performance, characterized by increased accuracy, reduced computational costs, and improved model interpretability. Depending on whether the training set is labelled or not, FS methods can be broadly categorized into supervised, unsupervised and semi-supervised techniques. In this paper, we focus mainly on supervised FS methods for classification problems. Indeed, due to the availability of class information, the relevance of features is gauged as the capability of distinguishing different classes. For instance, a feature f_i is said to be relevant to a class c_j if f_i and c_j are highly correlated with each other.

Supervised FS methods can be broadly categorized into three main approaches: filter, wrapper, and embedded methods. The filter method separates feature selection from classifier learning, ensuring that the bias of a learning algorithm does not collaborate with the bias of a FS algorithm. Therefore, it relies solely on some measures such as correlation, dependency, consistency, information, and distance. ReliefF (Robnik-Sikonja and Kononenko 2003), Correlation-based (Hall 2000), and Information Gain based methods (Kraskov, Stögbauer, and Grassberger 2004) are among the most classical algorithms of the filter method. The wrapper methods exploit the predictive accuracy of a predefined learning algorithm to evaluate the quality of selected features. They are known for their significant computational costs when processing data with a high number of features. Sequential forward, sequential backward, floating search (Pudil, Novovičová, and Kittler 1994) and recursive support vector machine (R-SVM) (Zhang et al. 2006) are some examples of the wrapper method. Incorporation of feature selection as part of the training process is the main difference of the embedded method from filter and wrapper methods. In other words, it achieves model fitting and feature selection simultaneously. Methods such as C4.5 (Quinlan 1993), recursive feature elimination using support vector machines (RFE-SVM) (Guyon et al. 2002), and Lasso Regularization (Tibshirani 1996) are among the known embedded methods.

In contrast to the previous approaches, our logic-based method, namely the Logical Analysis of Data (LAD), aims to differentiate between various groups of data. It is a supervised FS method utilizing the interaction between features and classes. During training, it generates feature subsets that discriminate between different groups of data in the given training set. Those feature subsets are then ranked using a scoring method, and the highest quality subset is selected. Finally, the highest quality one is utilized for training with a learning algorithm, enabling prediction on the testing set.

Our FS approach can be categorized into the filter method, as it relies on logic and a scoring method to select a feature subset without any interaction with the learning algorithms.

2.2 Learning Optimal Decision Trees

Decision trees (DTs) are widely used in machine learning, primarily due to their interpretability, which can be crucial in certain scenarios, even at the cost of lower accuracy. The predominant methods for computing DTs, such as CART (Breiman et al. 1984), ID3 (Quinlan 1986) and C4.5 (Quinlan 1993) are greedy heuristics that usually construct trees from top to bottom by selecting the feature with the highest information gain. Recently, there have been introductions of exact methods aimed at finding optimal DTs, considering criteria like size (nodes), depth, and empirical accuracy for certain combinations. For instance, (Bessiere, Hebrard, and O'Sullivan 2009; Narodytska et al. 2018; Avellaneda 2020) minimize either number of nodes or maximum depth, subject to the constraint that the tree is perfectly accurate on the training set. Similarly, (Hu et al. 2020) show how the SAT model can be adaptable to a MaxSAT approach as its formulation allows to use any linear combination of size, depth and accuracy. Other techniques (Nijssen and Fromont 2007; Bertsimas and Dunn 2017; Verwer and Zhang 2019; Hu, Rudin, and Seltzer 2019; Aglin, Nijssen, and Schaus 2020) choose to optimize the training accuracy to improve generalization performance, subject to constraints on the maximum depth. Exact algorithms have not supplanted greedy heuristics as the prevailing method because the scalability remains a concern.

Despite advancements in this field, learning optimal DTs remains NP-hard for various optimality criteria (Hyafil and Rivest 1976). Most studies focus solely on binary datasets of reasonable size (Narodytska et al. 2018), and rarely on numerical datasets (Schidler and Szeider 2021). In addition to these limitations, this paper addresses a notable gap in optimality criteria, particularly the minimum number of features sufficient for learning DTs. LAD-based FS is proposed to identify such highly discriminant feature subsets.

3 Preliminaries

3.1 Machine Learning Classification

We consider a Machine Learning (ML) classification problem, defined by a set of features $\mathcal{F} = \{X_1, \ldots, X_s\}$, and a

set of classes $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$, where $\mathcal{C}$ is associated with a class attribute denoted as C. Each feature $X_i \in \mathcal{F}$ takes values from a domain $\mathbb{D}_i$ (Domains may correspond to Discrete or Continuous data). Thus, feature space is defined as $\mathbb{F} = \prod_{i=1}^{s} \mathbb{D}_i$. To refer to an observation vector in $\mathbb{F}$, we use the notation $\mathbf{u} = (u_1, \ldots, u_s)$, with $u_i \in \mathbb{D}_i$, $i = 1, \ldots, s$. An instance denotes a pair $(\mathbf{u}_q, c_q)$, where $\mathbf{u}_q \in \mathbb{F}$ and $c_q \in \mathcal{C}$. We define a numerical dataset E as a set of instances; $|E|$ denotes its size. An ML classifier is characterized by a classification function τ that maps $\mathbb{F}$ into $\mathcal{C}$, i.e. $\tau : \mathbb{F} \to \mathcal{C}$. To learn a classifier, a set of instances $\{(\mathbf{u}_1, c_1), \ldots, (\mathbf{u}_K, c_K)\}$ is used by a learning algorithm that returns a function that best fits the training data and tries to generalize well on unseen test data.

For binary classification with data containing binary features, let $\mathcal{F}_\mathcal{B} = \{x_1, \ldots, x_t\}$ be a set of t binary features, and $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ be a binary dataset partitioned into a set of positive instances $\mathcal{E}^+$ and a set of negative instances $\mathcal{E}^-$. A binary instance denotes a pair $\epsilon_q = (\mathbf{v}_q, c_q) \in \mathcal{E}$, where $\mathbf{v}_q \in \{0,1\}^t$ denotes an observation vector of t binary features, and $c_q \in \{0,1\}$ is the class label. We have $c_q = 1$ if $\epsilon_q \in \mathcal{E}^+$ and $c_q = 0$ if $\epsilon_q \in \mathcal{E}^-$.

3.2 Logical Analysis of Data

Logical Analysis of Data (LAD) is a logic-based data analysis methodology combining ideas and concepts from discrete optimization, combinatorics and the theory of Boolean functions. It was first introduced over 30 years ago by Peter L. Hammer (Hammer 1986; Crama, Hammer, and Ibaraki 1988). The central concepts behind LAD are the computation of minimum support sets of features for explaining all observations, and the discovery of hidden patterns capable of distinguishing positive and negative sets of observations. A collection of such patterns is used for building a classification procedure, clustering, feature selection and other related problems. LAD has undergone continuous development, leading to numerous applications in various domains including medicine, business, seismology, oil exploration, etc. For more details, we refer to (Boros et al. 2000a).

Despite its widespread popularity in the discrete mathematics community, and its connections with several approaches developed in machine learning and data mining, LAD has not attracted the curiosity of the AI community.

Our goal in this paper is to bridge this gap, by using the main first step of LAD, which involves discovering relevant subsets of features that distinguish observations among different groups of data, while using them for training and testing ML classifiers. In the sequel, we use similar notations and definitions to those of (Hammer and Bonates 2006).

From Section 3.1, we have a set of t binary features $\mathcal{F}_\mathcal{B} = \{x_1, \ldots, x_t\}$, and a binary dataset $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$. Given a set of binary features $\mathcal{S} \subseteq \mathcal{F}_\mathcal{B}$, we denote by $\mathcal{E}^+_\mathcal{S}$ (resp. $\mathcal{E}^-_\mathcal{S}$) as the projection of $\mathcal{E}^+$ (resp. $\mathcal{E}^-$) on $\mathcal{S}$. $\mathcal{S}$ is called a *support set* if $\mathcal{E}^+_\mathcal{S} \cap \mathcal{E}^-_\mathcal{S} = \emptyset$. Moreover, $\mathcal{S}$ is called *irredundant* or *minimal* if no proper subset of it is a support set.

Example 1. *Consider the dataset in Fig. 1. The projection of $\mathcal{E}^+$ and $\mathcal{E}^-$ on $\mathcal{S} = \{x_1, x_2\}$ is $\mathcal{E}^+_\mathcal{S} = \{(0,0), (1,1)\}$ and $\mathcal{E}^-_\mathcal{S} = \{(1,0)\}$, respectively. Moreover, $\{x_1, x_2\}$, $\{x_2, x_3\}$, $\{x_3, x_4, x_5\}$, and $\{x_1, x_4, x_5\}$ are minimal (w.r.t. inclusion) support sets. Concretely, $\{x_1, x_2\}$ and $\{x_2, x_3\}$ are the minimum size support sets (w.r.t. cardinality).*

x_1	x_2	x_3	x_4	x_5	$class$
0	0	0	1	1	1 (c_1)
0	0	0	0	1	1 (c_1)
1	1	1	1	0	1 (c_1)
1	0	1	0	0	0 (c_2)
1	0	1	1	1	0 (c_2)

Figure 1: Binary dataset

To compute minimum support sets in $\mathcal{E}$, we associate with every feature $x_k \in \mathcal{F}_\mathcal{B}$ a new binary variable y_k, where $k = 1, \ldots, t$, $y_k = 1$ if x_k is part of the support set, and $y_k = 0$ otherwise. Let $\mathbf{v} = (v_1, \ldots, v_t)$ and $\mathbf{v}' = (v'_1, \ldots, v'_t)$ be the binary observation vectors of t features associated with $\mathcal{E}^+$ and $\mathcal{E}^-$, respectively. We further associate the vectors $\mathbf{v}$ and $\mathbf{v}'$ with a vector $w(\mathbf{v}, \mathbf{v}') = (w_1(\mathbf{v}, \mathbf{v}'), \ldots, w_t(\mathbf{v}, \mathbf{v}'))$, where $w_k(\mathbf{v}, \mathbf{v}') = v_k \oplus v'_k (mod\ 2)$, i.e. $w_k(\mathbf{v}, \mathbf{v}') = 1$ if $v_k \neq v'_k$, and $w_k(\mathbf{v}, \mathbf{v}') = 0$ otherwise. We can obtain the minimum support sets by solving the following set covering problem:

$$\begin{aligned} min \quad & \sum_{k=1,\ldots,t} y_k \\ s.t. \quad & \\ \sum_{k=1,\ldots,t} & w_k(\mathbf{v}, \mathbf{v}') y_k \geq 1,\ \forall (\mathbf{v}, c) \in \mathcal{E}^+,\ \forall (\mathbf{v}', c') \in \mathcal{E}^- \\ & y_k \in \{0,1\} \end{aligned} \tag{1}$$

Example 2. *For the dataset in Fig. 1, the formulation of LAD-based Minimum Support Sets (LAD-MSS) is given by:*

$$\begin{aligned} min \quad & y_1 + y_2 + y_3 + y_4 + y_5 \\ s.t. \quad & \\ & y_1 + y_3 + y_4 + y_5 \geq 1 \\ & y_1 + y_3 \geq 1 \\ & y_1 + y_3 + y_5 \geq 1 \\ & y_1 + y_3 + y_4 \geq 1 \\ & y_2 + y_4 \geq 1 \\ & y_2 + y_5 \geq 1 \end{aligned} \tag{2}$$

It is well-known that the above problem is NP-hard and involves a quadratic number of linear inequalities. For large datasets, solving this optimization problem is impractical. In Section 4.1, we circumvent this problem by proposing a more efficient alternative, formulating it as the problem of enumerating minimal hitting sets in a hypergraph.

3.3 Minimal Hitting Set Enumeration Problem

Given a collection $H = \{H_1, \ldots, H_m\}$ of subsets of the vertex set V. H is also called a *hypergraph* where each element H_i of H is a *hyperedge*. A *hitting set* (*traversal*) T of H is a subset of V that intersects (hits) every set $H_i \in H$

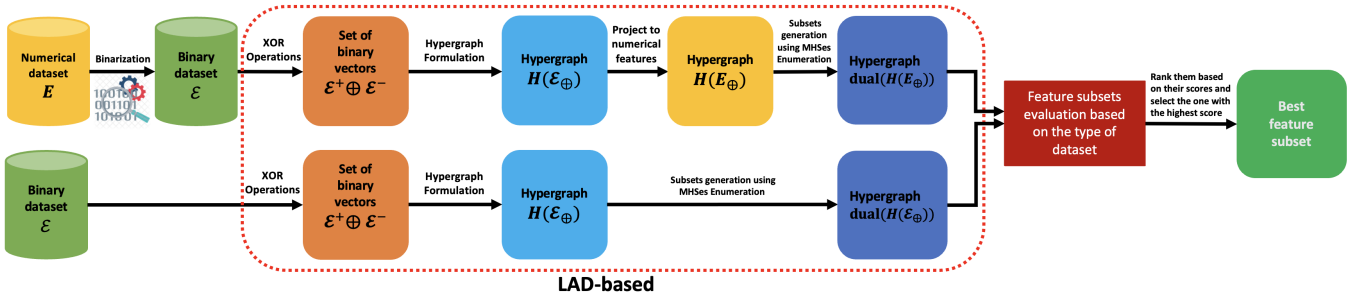

Figure 2: The overall proposed framework for LAD-based FS

i.e. $T \cap H_i \neq \emptyset$. T is called *Minimal Hitting Set* (MHS) of H if no proper subset of T is a hitting set of H. The dual of a hypergraph is the hypergraph whose hyperedge set is the set of all minimal hitting sets, and it is denoted by $dual(H)$. Hypergraph dualization refers to the problem of constructing the dual of a given hypergraph. Dualization is equivalent to many important problems, including minimal hitting set, minimal hypergraph transversal and minimal set covering enumeration problems. These related problems are fundamental in a wide variety of domains, including combinatorics, Boolean algebra, databases, computational biology, and artificial intelligence (AI). It finds important applications in AI such as diagnosis, machine learning, data mining, and explainable AI. Furthermore, several algorithms and efficient implementations for generating Minimal Hitting Sets (MHSes) have been proposed. For a complete review, we refer to (Gainer-Dewar and Vera-Licona 2016).

4 LAD-based Feature Selection

4.1 Feature Subsets Generation

As mentioned previously, LAD-based feature subsets generation is formulated as the problem of enumerating *minimum* support sets (w.r.t. cardinality). For complexity reasons, we consider the problem of enumerating *minimal* support sets (w.r.t. inclusion), reduced to the MHSes enumeration problem, for which efficient and scalable algorithms exist.

From Section 3.2, we have $\mathcal{F}_{\mathcal{B}} = \{x_1, \ldots, x_t\}$ a set of t binary features, $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ a binary dataset, $\mathbf{v}$ and $\mathbf{v}'$ the binary observation vectors associated with $\mathcal{E}^+$ and $\mathcal{E}^-$, respectively. We then derive a set of binary vectors from $\mathcal{E}$ denoted as $\mathcal{E}^+ \oplus \mathcal{E}^- = \mathcal{E}_\oplus = \{w(\mathbf{v}, \mathbf{v}'),\ \forall (\mathbf{v}, c) \in \mathcal{E}^+ \wedge \forall (\mathbf{v}', c') \in \mathcal{E}^-\}$. We define the hypergraph associated with $\mathcal{E}_\oplus$ as $H(\mathcal{E}_\oplus) = \{\mathcal{E}_\oplus^{(\mathbf{v},\mathbf{v}')},\ \forall (\mathbf{v}, c) \in \mathcal{E}^+ \wedge\ \forall (\mathbf{v}', c') \in \mathcal{E}^-\}$, where $\mathcal{E}_\oplus^{(\mathbf{v},\mathbf{v}')} = \{x_k \in \mathcal{F}_{\mathcal{B}} \mid k \in \{1, \ldots, t\} \text{ and } w_k(\mathbf{v}, \mathbf{v}') = 1\}$ is the hyperedge associated with $(\mathbf{v}, \mathbf{v}')$. Then we derive the following property:

Proposition 1. *The minimal support sets of $\mathcal{E}$ correspond to $dual(H(\mathcal{E}_\oplus))$.*

Proof. Let $h \in dual(H(\mathcal{E}_\oplus))$. As h is a minimal hitting set of the hypergraph $H(\mathcal{E}_\oplus)$, it intersects each hyperedge $\mathcal{E}_\oplus^{(\mathbf{v},\mathbf{v}')} \in H(\mathcal{E}_\oplus)$. Let $(\mathbf{v}, c) \in \mathcal{E}^+$ and $(\mathbf{v}', c') \in \mathcal{E}^-$ s.t. $\mathbf{v} = (v_1, \ldots, v_t)$ and $\mathbf{v}' = (v'_1, \ldots, v'_t)$. Then $\exists x_l \in h \cap \mathcal{E}_\oplus^{(\mathbf{v},\mathbf{v}')}$. Since $\mathcal{E}_\oplus^{(\mathbf{v},\mathbf{v}')} = \{x_k \mid k \in \{1, \ldots, t\} \text{ and } w_k(\mathbf{v}, \mathbf{v}') = 1\}$. This means that the projection of $\mathbf{v}$ and $\mathbf{v}'$ on x_k are different i.e. $v_l \neq v'_l$. Consequently, h is a minimal support sets of $\mathcal{E}$. The converse is also true. Indeed, if we take $\mathcal{S}$ a minimal support sets of $\mathcal{E}$, then by definition $\mathcal{E}_\mathcal{S}^+ \cap \mathcal{E}_\mathcal{S}^- = \emptyset$. Then $\mathcal{S} \in dual(H(\mathcal{E}_{\oplus\mathcal{S}}))$. Consequently, $\mathcal{S} \in dual(H(\mathcal{E}_\oplus))$.

Example 3. *From the formulation of LAD-MSS as a 0/1 linear program (Example 2), we can derive the following hypergraph, where each hyperedge corresponds to the set of features associated with the set of variables involved in each 0/1 linear inequality: $H(\mathcal{E}_\oplus) = \{\{x_1, x_3, x_4, x_5\}, \{x_1, x_3\}, \{x_1, x_3, x_5\}, \{x_1, x_3, x_4\}, \{x_2, x_4\}, \{x_2, x_5\}\}$. Then $dual(H(\mathcal{E}_\oplus)) = \{\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4, x_5\}, \{x_1, x_4, x_5\}\}$ corresponds to minimal supports sets. (see Example 1).*

Note that the number of MHSes is exponential in the worst case. Fredman et al. (Fredman and Khachiyan 1996) developed a quasi-polynomial time algorithm which runs in $O(N^{logN})$ time, where N is the *input size plus output size*.

4.2 Feature Subsets Evaluation

After enumerating MHSes as the feature subsets, we need to evaluate and rank them based on their relevance to the target class before selecting the subset with the highest score. To achieve this, we use a heuristic scoring method proposed in the test theory of Edwin Ghiselli (Ghiselli 1964). This theory considers the usefulness of individual features for predicting the class label along with the level of intercorrelation among them. It was also utilized by (Hall 2000) to calculate the merit of a feature subset, based on the claim that "*good feature subsets contain features highly correlated with the class, yet uncorrelated with each other*". The most significant strength of this method is that it evaluates and ranks feature subsets rather than individual features like ReliefF. For instance, to assess the quality of a student (class: poor, medium, high), it involves a variety of subjects' scores (features), such as math, physics, chemistry, etc., rather than any individual subject's score, which measures a restricted scope of features. The equation below characterizes this heuristic:

$$Merit_\mathcal{S} = \frac{|\mathcal{S}|\ \overline{r_{cf}}}{\sqrt{|\mathcal{S}| + |\mathcal{S}|\ (|\mathcal{S}| - 1)\ \overline{r_{ff}}}} \tag{3}$$

where $Merit_\mathcal{S}$ is the general measure of the "merit" of a feature subset $\mathcal{S}$, its size is denoted by $|\mathcal{S}|$, $\overline{r_{cf}}$ the average

feature-class correlation, and $\overline{r_{ff}}$ the average feature-feature correlation. The numerator indicates how predictive a group of features is whereas the denominator indicates how much redundancy exists among them. The heuristic deals with irrelevant features, as they tend to be poor predictors of the class. Redundant features are penalized, as they tend to be highly correlated with one or more of the other features.

To evaluate the "merit" of a feature subset $\mathcal{S}$, we instantiate the r_{cf} and r_{ff} in Equation (3) with different scoring functions depending on the type of dataset:

- *Numerical Dataset:* For a numerical dataset E where features contain continuous or discrete values, along with the target class considered as a categorical variable, we distinguish between two types of correlation as follows:
  1. *feature-feature correlation (r_{ff}):* Let $X, X' \in \mathcal{S} \subseteq \mathcal{F}$ be two numerical features. $E_X = (X_1, \ldots, X_{|E|})$ and $E_{X'} = (X'_1, \ldots, X'_{|E|})$ as the projections of E on X and X', respectively. To compute the correlation between E_X and $E_{X'}$, we use two measures: Pearson Correlation Coefficient and Symmetrical Uncertainty.
     - *Pearson Correlation Coefficient (PCC):* Named after Karl Pearson (1857–1936), it measures the linear correlation between E_X and $E_{X'}$. The PCC between E_X and $E_{X'}$ is defined as follows:

$$PCC = \frac{\sum_{i=1}^{|E|}(X_i - \bar{X})(X'_i - \bar{X}')}{\sqrt{\sum_{i=1}^{|E|}(X_i - \bar{X})^2 \sum_{i=1}^{|E|}(X'_i - \bar{X}')^2}}$$

       where $\bar{X} = \frac{\sum_{i=1}^{|E|} X_i}{|E|}$ is the sample mean; and likewise for $\bar{X}'$. $\overline{r_{ff}}$ is obtained by calculating the pairwise average of PCC between all pairs of features in $\mathcal{S}$.
     - *Symmetrical Uncertainty (SU):* A modified version of information gain to estimate the degree of association between discrete features (Press et al. 1988). This measure was also implemented in Correlation-based (Hall 2000) to measure the feature-feature correlation. The SU between E_X and $E_{X'}$ is defined as follows:

$$SU = 2.0 \times \left[\frac{\Delta(E_X) + \Delta(E_{X'}) - \Delta(E_X, E_{X'})}{\Delta(E_X) + \Delta(E_{X'})}\right]$$

       where Δ is the entropy (Hall 1999). Similarly, $\overline{r_{ff}}$ can be obtained by calculating the pairwise average of SU between all possible pairs of features in $\mathcal{S}$.
  2. *feature-class correlation (r_{cf}):* To calculate the correlation between a feature (numerical variable) and a target class (categorical variable), we use Correlation Ratio. It measures the curvilinear relationship between the dispersion in individual categories and the dispersion across the whole observation. Mathematically, it is defined as the weighted variance of the category means divided by the variance of all observations; it measures how well a continuous number can be classified into categories (score in $[0, 1]$). Let $X \in \mathcal{S}$ be a numerical feature and $E_X = (X_1, \ldots, X_{|E|})$ the projection of E on X. Let $E_C = (c'_1, \ldots, c'_{|E|})$ be the projection of E on C (class attribute). We define $V_i = (X_j \in E_X \mid 1 \leq j \leq |E| \text{ and } c'_j = c_i)$ as a vector to store E_X for all instances of class c_i for $1 \leq i \leq K$. The Correlation Ratio between E_X and E_C, denoted as η_{XC}, is defined as follows:

$$\eta_{XC} = \sqrt{\frac{\sigma^2_{\bar{X}}}{\sigma^2_X}}, \text{ where } \sigma^2_X = \frac{\sum_{i=1}^{K}\sum_{X_j \in V_i}(X_j - \bar{X})^2}{\sum_{i=1}^{K}|V_i|},$$
$$\sigma^2_{\bar{X}} = \frac{\sum_{i=1}^{K}|V_i|(\bar{X}_i - \bar{X})^2}{\sum_{i=1}^{K}|V_i|}, \bar{X} = \frac{\sum_{i=1}^{K}|V_i|\bar{X}_i}{\sum_{i=1}^{K}|V_i|},$$
$$\bar{X}_i = \frac{\sum_{X_j \in V_i} X_j}{|V_i|}$$

     where $|V_i|$ is the number of observations in class c_i, $\bar{X}_i$ is the sample mean of class c_i, $\bar{X}$ is the sample mean of all classes in $\mathcal{C}$. $\overline{r_{cf}}$ can be obtained by computing the average of η_{XC} for all features in $\mathcal{S}$.
- *Binary Dataset:* For binary dataset, we use Matthews Correlation Coefficient (MCC) (Matthews 1975) to compute both r_{ff} and r_{cf}. MCC is used as a measure of association between two binary variables. Explicitly, it is a contingency matrix method of calculating the Pearson product-moment correlation coefficient, and it shares the same interpretations as PCC (Powers 2020). Using true positive (TP), true negative (TN), false positive (FP), and false negative (FN) parameters, the MCC between two binary variables is defined as follows:

$$MCC = \frac{(TP*TN)-(FP*FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

  $\overline{r_{ff}}$ (resp. $\overline{r_{cf}}$) can be obtained by calculating the average of MCC between all possible pairs of features in $\mathcal{S}$ (resp. between each feature in $\mathcal{S}$ and a class variable). For example, consider $\{x_1, x_2\}$ in Fig. 1, we have $\mathcal{E}_{x_1} = (0, 0, 1, 1, 1)$, $\mathcal{E}_{x_2} = (0, 0, 1, 0, 0)$, and $\mathcal{E}_{class} = (c_1, c_1, c_1, c_2, c_2)$. Table 1 is the contingency matrix between $\mathcal{E}_{x_2}$ and $\mathcal{E}_{class}$. TN, FN, FP and TP corresponds respectively to the number of times $(0, 0)$, $(1, 0)$, $(0, 1)$ and $(1, 1)$, appears in $((x_{2_1}, c_1), (x_{2_2}, c_1), (x_{2_3}, c_1), (x_{2_4}, c_2), (x_{2_5}, c_2))$.

		$class$	
		0	1
x_2	0	2 (True Negative [TN])	2 (False Positive [FP])
	1	0 (False Negative [FN])	1 (True Positive [TP])

Table 1: Contingency matrix between x_2 and $class$

5 Optimal Decision Trees vs LAD-based FS

We first recall some necessary definitions and notations. We have $\mathcal{E}$ a binary dataset, and $\mathcal{F}_\mathcal{B}$ a set of t binary features. A Boolean decision tree is a binary tree $\mathcal{T}(\mathcal{E})$, each of whose internal nodes is labeled with one of the t input binary features, and whose leaves are labeled 0 or 1. Every variable is assumed to appear at most once on any root-to-leaf path. The size of $\mathcal{T}$, denoted $|\mathcal{T}|$, is given by the number of its nodes. The depth of the decision tree is the length of the longest path in the tree. If $\mathcal{T}$ is a decision tree, and $\mathcal{E}'$ is a set of training examples, we say that $\mathcal{T}$ is consistent with $\mathcal{E}'$,

if each example $e \in \mathcal{E}'$ is correctly classified by $\mathcal{T}$. Building optimal DTs represents a crucial stride in crafting precise and understandable ML models. The evaluation of optimality typically hinges on simplicity criteria, such as tree's depth and size. From the complexity side, learning an optimal decision tree is NP-complete, even for these two basic criteria (Avellaneda 2020). As previously stated, our objective is to build an optimal tree, where optimality refers to the minimum number of features (feature subsets) involved in the construction of a decision tree. Such feature subsets are computed by enumerating MHSes (see Section 4.1).

The following properties state the equivalence between the decision tree with the minimum depth and the decision tree build on the minimum number of features. Let us recall the well-known relationship between the depth and the tree size. Balanced or full binary trees admit $2^{d+1} - 1$ nodes, d is the depth of the tree. Consequently, minimising the depth is the most considered optimization criteria, since low depth DTs require fewer tests and are usually more accurate.

Proposition 2. *Let $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ a binary dataset, $\mathcal{S} \subseteq \mathcal{F}_{\mathcal{B}}$. If $\mathcal{S}$ is a minimum support set of $\mathcal{E}$ then the minimum depth of $\mathcal{T}(\mathcal{E})$ is (upper) bounded by $|\mathcal{S}|$.*

Proof. Let $\mathcal{S}$ be a minimum support set of $\mathcal{E}$ i.e. $\mathcal{E}^+_{\mathcal{S}} \cap \mathcal{E}^-_{\mathcal{S}} = \emptyset$. So every Boolean tuple $s \in \mathcal{E}_{\mathcal{S}}$ can be correctly classified. $\mathcal{T}((\mathcal{E}^+_{\mathcal{S}}, \mathcal{E}^-_{\mathcal{S}}))$ is then consistent with $\mathcal{E}_{\mathcal{S}}$. As the path from the root to the leaf in $\mathcal{T}((\mathcal{E}^+_{\mathcal{S}}, \mathcal{E}^-_{\mathcal{S}}))$ is labeled with the Boolean features in $\mathcal{S}$, its depth is bounded by $|\mathcal{S}|$. The result holds for $\mathcal{T}(\mathcal{E})$.

Proposition 3. *Let $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ a binary dataset and $\mathcal{S} \subseteq \mathcal{F}_{\mathcal{B}}$. $\mathcal{S}$ is a minimum support set of $\mathcal{E}$ iff $\mathcal{T}((\mathcal{E}^+_{\mathcal{S}}, \mathcal{E}^-_{\mathcal{S}}))$ is a minimum Boolean decision tree w.r.t. the number of features.*

Proof.
- $\Rightarrow$) - Let $\mathcal{S}$ be a minimum support set of $\mathcal{E}$. Then $\mathcal{E}^+_{\mathcal{S}} \cap \mathcal{E}^-_{\mathcal{S}} = \emptyset$. $\mathcal{T}((\mathcal{E}^+_{\mathcal{S}}, \mathcal{E}^-_{\mathcal{S}}))$ is then consistent with $\mathcal{E}_{\mathcal{S}}$. Suppose that $\mathcal{T}(\mathcal{E}_{\mathcal{S}})$ is not optimal w.r.t. minimum number of features. Then, there exists $\mathcal{S}' \subset \mathcal{S}$ such that $\mathcal{T}((\mathcal{E}^+_{\mathcal{S}'}, \mathcal{E}^-_{\mathcal{S}'}))$ is a consistent Boolean decision tree on $\mathcal{E}_{\mathcal{S}'}$. So $\mathcal{E}^+_{\mathcal{S}'} \cap \mathcal{E}^-_{\mathcal{S}'} = \emptyset$. Therefore, $\mathcal{S}$ is not a minimum support set of $\mathcal{E}$.
- $\Leftarrow$) - Let $\mathcal{T}((\mathcal{E}^+_{\mathcal{S}}, \mathcal{E}^-_{\mathcal{S}}))$a minimum Boolean decision tree w.r.t. the number of features. Suppose that $\mathcal{S}$ is not a minimum support set. Then there exists a support set $\mathcal{S}' \subset \mathcal{S}$. Consequently, there exists a consistent Boolean decision tree $\mathcal{T}((\mathcal{E}^+_{\mathcal{S}'}, \mathcal{E}^-_{\mathcal{S}'}))$. Then $\mathcal{T}((\mathcal{E}^+_{\mathcal{S}}, \mathcal{E}^-_{\mathcal{S}}))$ is not optimal.

Proposition 2 and 3 provide a means to connect optimal DTs (with minimum depth) and optimal DTs (with minimum number of features) to LAD-based FS (with minimum support sets).

Let $\{x_1, x_2\}$ and $\{x_2, x_3\}$ be two minimum support sets of Example 1. The optimal DTs (w.r.t. minimum number of features) are depicted in Fig. 3. The above propositions are important for two reasons. First, it allows to link recent works on optimal DTs to those of LAD-based FS. Second, it allows us to use minimum support sets of LAD-based FS to construct optimal DTs.

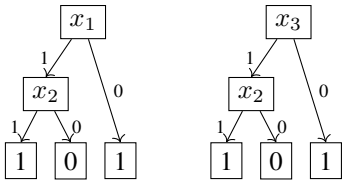

Figure 3: Optimal DTs with minimum number of features for $\{x_1, x_2\}$ (left side) and $\{x_2, x_3\}$ (right side)

6 Proposed Framework for LAD-based FS

This section provides a description of our approach for selecting the best feature subset in both binary and numerical datasets. For a numerical dataset, it consists of the following processing steps, as depicted in Fig. 2 :

1. *Binarization* ($E \rightarrow \mathcal{E}$) : To address a numerical dataset, the methodology of LAD is extended to a process called *binarization*, which involves transforming numerical (real value) data into binary (0,1) representations (Boros et al. 2000a; Boros et al. 2000b). Therefore, we utilize the binarization technique as described in (Boros et al. 2000a). This transformation maps each observation vector, denoted as $\mathbf{u_q} = (u_1, \ldots, u_s)$ of a given numerical dataset E to a binary vector of a binary dataset $\mathcal{E}$, denoted as $\mathbf{v_q} = (v_1, \ldots, v_t) \in \{0,1\}^t$, by associating each numerical feature X_i with a set of binary features $\{x_{i1}, \ldots, x_{ib_i}\}$, where b_i is the number of binary features associated with X_i. This is done in such a way that if $\mathbf{u_q}$ and $\mathbf{u_r}$ represent a positive and a negative observation vector, respectively, then $\mathbf{v_q} \neq \mathbf{v_r}$.
2. *XOR Operations* ($\mathcal{E} \rightarrow \mathcal{E}^+ \oplus \mathcal{E}^-$): We execute the XOR operations between every observation in $\mathcal{E}^+$ and every observation in $\mathcal{E}^-$ of $\mathcal{E}$ denoted as $\mathcal{E}_{\oplus} = \mathcal{E}^+ \oplus \mathcal{E}^-$, a set of binary vectors. For multiple classes, we execute the XOR operations between each pair of classes resulting in $\frac{K \times (K-1)}{2}$ pairs, where K is the number of classes.
3. *Hypergraph Formulation* ($\mathcal{E}^+ \oplus \mathcal{E}^- \rightarrow H(\mathcal{E}_{\oplus})$): From the binary set of vectors $\mathcal{E}_{\oplus}$, we derive a hypergraph $H(\mathcal{E}_{\oplus})$, where each binary vector is mapped to a hyperedge, made of a set of features with value 1.
4. *Project hypergraph to numerical features* ($H(\mathcal{E}_{\oplus}) \rightarrow H(E_{\oplus})$): A major drawback of binarization is that, it can yield millions of binary features. To alleviate this problem, we propose to project the hypergraph of binary features $H(\mathcal{E}_{\oplus})$ on the numerical features by linking back all those binary features to their corresponding numerical features, denoted as $H(E_{\oplus})$. To achieve this, a numerical feature exists as a vertex in a hyperedge if at least one binary feature is associated with it; otherwise, it does not exist. Moreover, each hyperedge only consists of distinct vertices although after the projection, there are many binary features corresponding to a single numerical feature. For illustration, assume that we have three numerical features $\{X_1, X_2, X_3\}$ associated with their respective binary features $\{x_{11}, x_{12}\}$, $\{x_{21}, x_{22}\}$, and $\{x_{31}, x_{32}, x_{33}\}$. If H_b = $\{x_{11}, x_{31}, x_{32}\}$ is a hyperedge of binary features,

then its projection on numerical features lead to the hyperedge $H_n = \{X_1, X_3\}$. Note that in this step, we only exploit all the binary features associated with all the numerical features to execute the XOR operations and then we project them back to their corresponding numerical features before generating subsets of features.

5. *Feature subsets generation using MHSes Enumeration* ($H(E_\oplus) \rightarrow dual(H(E_\oplus))$): To mitigate the complexity of LAD optimization problem (minimum size support sets of features), we formulate it as the problem of enumerating Minimal Hitting Sets (MHSes) for generating feature subsets; the dual of a hypergraph denoted as $dual(H(E_\oplus))$ (see Section 4.1). To dualize $H(E_\oplus)$, we use state-of-the-art algorithm called "pMMCS" (Gainer-Dewar and Vera-Licona 2016), a parallel implementation version of MMCS (Minimal-to-Maximal Conversion Search) algorithm, for solving the dualization problem (Murakami and Uno 2014), whose complexity is in $O(\|H\|)$ time per MHS and $O(\|H\|)$ memory, where $\|H\|$ denotes the sum of the sizes of hyperedges in H. We then obtain the MHSes (subsets of features) denoted as $dual(H(E_\oplus))$. The benefit of this algorithm lies in its capacity to handle very large-scale problems with up to millions of hyper-edges, generating numerous solutions in a short amount of time, which motivates our choice. Furthermore, it also supports multithreading implementations on computer with multiple CPUs. However, as mentioned in (Hall 2000; Guyon and Elisseeff 2003; Tang, Alelyani, and Liu 2014), the size of search space for s features is $O(2^s)$. Due to this reason, in all the practical experiments, we limit our search space to enumerate 10,000 subsets of features for each dataset.
6. *Feature subsets evaluation*: We need to evaluate the generated subsets based on their merit scores to select the highest quality one, as discussed in Section 4.2. The merit scores can be calculated depending on the type of dataset. After this step, we finally obtain the best feature subset that can be used for training and testing ML classifiers.

To handle a binary dataset, the process is similar to a numerical dataset, except steps (1) and (4), which do not apply.

7 Experiments

We conduct two types of comparative experiments on benchmark datasets, including two-class binary datasets and numerical datasets with two and multiple classes. In the first experiment, we compare our approach with state-of-the-art (SOTA) feature selection (FS) techniques, including ReliefF (Robnik-Sikonja and Kononenko 2003), Correlation-Based (CFS) (Hall 2000), Mutual Information (MI) (Kraskov, Stögbauer, and Grassberger 2004; Ross 2014) for *filter methods*, Sequential Forward Selection using Support Vector Machines (SFS-SVM) and Sequential Backward Selection using Support Vector Machines (SBS-SVM) (Pudil, Novovičová, and Kittler 1994; Ferri et al. 1994) for *wrapper methods*, as well as Recursive Feature Elimination using Support Vector Machines (RFE-SVM) (Guyon et al. 2002) and Lasso (Tibshirani 1996) for *embedded methods*. We conduct the experiments with decision trees using CART (Breiman et al. 1984) on two-class binary datasets mostly from CP4IM[1][2]. In the second experiment, we compare our approach with the same SOTA FS techniques, but using additional classifiers including SVM (Cortes and Vapnik 1995), KNN (Mucherino, Papajorgji, and Pardalos 2009) and Naive Bayes (Zhang 2004), alongside CART. We utilize 12 numerical datasets from *OpenML* (Bischl et al. 2021) and *datamicroarray* (Ramey 2016). The datasets from *datamicroarray* are high-dimensional two-class datasets with more features than instances. The FS techniques and ML classifiers can be found in the scikit-learn python library (Pedregosa et al. 2011). All the SOTA techniques are kept at their default settings. Following (Hall and Holmes 2003; Chandrashekar and Sahin 2014), we use accuracy as the evaluation metric. For all experiments, we use Repeated Stratified 10-fold cross-validation with 3 repetitions to maintain the class distribution, and they are conducted on an Intel XEON Gold 6248 @ 2.5 GHz with 768 Gib of memory.

7.1 Experiments on binary datasets

In this section, we compare our approach with other SOTA approaches using CART. Their parameters are kept at their default values, except for the maximum depth. Following the experiments in (Hu et al. 2020), we set three distinct depths, $d \in \{2, 3, 4\}$ in the second column. The results are reported in Table 2. The first column shows the name of the dataset, the size of dataset ($\#s$), and the number of binary features ($\#f_b$). The term "W/O FS" in the third column represents the accuracy without applying any FS techniques. The fourth column displays the result of our approach, while the remaining columns represent the results of the SOTA approaches. Each result contains the testing accuracy "Acc", and the average number of selected features ($\#nbf$).

Throughout, we consider a difference of 1% in accuracy for comparison. In Table 2, our approach maintains or improves the accuracy approximately around 70% of the overall settings. For "hepatitis", it significantly improves the accuracy in all depth settings around 3% and is competitive with other SOTA approaches especially when $d = 2$. Similarly, for "heart-cleveland", it enhances the accuracy in two depth settings particularly outperforming all the SOTA approaches when $d = 2$. For "mushroom" when $d = 2$, our approach outperforms the others and is competitive with RFE-SVM. However, it falls short in all depth settings by around 2%, particularly for "dorothea" and "german-credit". The accuracy of ReliefF is degraded on 8 out of 17 datasets and drastically degrades in all depth settings of "breast-cancer", but is improved for "anneal", "heart-cleveland" and "hepatitis", while other approaches could not enhance the accuracy for "anneal". The performances of CFS and MI are similar but CFS is better than MI in several cases. For the wrapper methods, SFS-SVM and SBS-SVM are very similar in terms of the improvement or degradation of accuracy and the average number of selected features. For the embedded methods, RFE-SVM and Lasso are also comparable,

[1] https://dtai.cs.kuleuven.be/CP4IM/datasets/

[2] Note: For certain datasets, instances appearing in both $\mathcal{E}^+$ and $\mathcal{E}^-$ are removed to maintain consistency.

Instance $\#s, \#f_b$	d	W/O FS	LAD-based		Filter Methods						Wrapper Methods				Embedded Methods			
					ReliefF		CFS		MI		SFS-SVM		SBS-SVM		RFE-SVM		Lasso	
		Acc	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$
	2	0.8539	0.8534		0.8670○		0.8548		0.8530		0.8539		0.8548		0.8539		0.8539	
anneal	3	0.8567	0.8562	23	0.8787○	10	0.8656	6.36	0.8572	10	0.8567	46	0.8581	47	0.8572	64	0.8567	48.8
712, 93	4	0.8609	0.8567		0.8791○		0.8885○		0.8623		0.8651		0.8586		0.8595		0.8581	
	2	0.9492	0.9075●		0.8660●		0.9492		0.9233●		0.9492		0.9492		0.9492		0.9492	
audiology	3	0.9615	0.9369●	15.37	0.8660●	10	0.9583	6	0.9366●	10	0.9552	74	0.9522	74	0.9614	11.47	0.9600	12.23
216, 148	4	0.9554	0.9508		0.8600●		0.9585		0.9322●		0.9538		0.9445●		0.9600		0.9523	
	2	0.9395	0.9165●		0.6935●		0.9424		0.9415		0.9414		0.9375		0.9405		0.9385	
breast-cancer	3	0.9385	0.9370	24.76	0.7043●	10	0.9444	6	0.9395	10	0.9400	44	0.9463	45	0.9429	35.63	0.9395	35.63
683, 89	4	0.9409	0.9390		0.7135●		0.9497		0.9400		0.9409		0.9463		0.9448		0.9463	
	2	0.8553	0.8553		0.8553		0.8553		0.8553		0.8240●		0.7885●		0.8553		0.8553	
car	3	0.8719	0.8719	14	0.8719	10	0.8719	6	0.8690	10	0.8495●	10	0.8333●	11	0.8719	17.6	0.8719	17.63
1728, 21	4	0.8719	0.8759		0.8701		0.8665		0.8749		0.8919○		0.8736		0.8728		0.8719	
	2	0.9307	0.9072●		0.9318		0.9275		0.9327		–		–		–		0.9324	
dorothea	3	0.9289	0.9081●	149	0.9339	10	0.9260	6	0.9292	10	–	–	–	–	–	–	0.9260	320.3
1150, 100000	4	0.9266	0.9072●		0.9356		0.9304		0.9304		–		–		–		0.9249	
	2	0.7120	0.6919●		0.6956●		0.6846●		0.6963●		0.7006●		0.7029		0.6933●		0.7046	
german-credit	3	0.7236	0.7020●	29.43	0.7099●	10	0.7036●	6	0.7143	10	0.7083●	56	0.7156	56	0.6996●	57.66	0.7106●	77.8
1000, 112	4	0.7249	0.6913●		0.7119●		0.7143●		0.7090●		0.7119●		0.7080●		0.7086●		0.7093●	
	2	0.7147	0.7747○		0.7170		0.7147		0.7170		0.7158		0.7147		0.7419○		0.7431○	
heart-cleveland	3	0.8004	0.7713●	17.13	0.8388○	10	0.8253○	6.96	0.8083	10	0.7902●	47	0.8059	48	0.7915	20.16	0.7982	49.66
296, 95	4	0.7568	0.7747○		0.8231○		0.8186○		0.7880○		0.7622		0.7702○		0.7813○		0.7713○	
	2	0.7813	0.8173○		0.7963○		0.7941○		0.7794		0.7989○		0.7990○		0.7789		0.7862	
hepatitis	3	0.7672	0.7956○	13	0.8272○	10	0.8038○	6	0.7838○	10	0.7970○	34	0.7844○	34	0.7789○	23.33	0.7820○	32
137, 68	4	0.7545	0.8025○		0.8034○		0.8082○		0.7967○		0.7771○		0.7846○		0.7666○		0.7842○	
	2	0.9837	0.9837		0.9685●		0.9837		0.9687●		0.9837		0.9837		0.9837		0.9837	
hypothyroid	3	0.9840	0.9840	24.46	0.9685●	10	0.9828	6	0.9687●	10	0.9846	44	0.9833	44	0.9849	4	0.9840	20.23
3199, 88	4	0.9865	0.9862		0.9685●		0.9825		0.9687●		0.9871		0.9841		0.9878		0.9867	
	2	0.7708	0.7708		0.7546●		0.7708		0.7708		0.7708		0.7708		0.7708		0.7708	
kr-vs-kp	3	0.9042	0.9042	29	0.7593●	10	0.9042	7	0.9042	10	0.9042	36	0.9042	37	0.9042	42.7	0.9042	55.53
3196, 73	4	0.9408	0.9408		0.7612●		0.9408		0.9250●		0.9408		0.9408		0.9408		0.9408	
	2	0.8192	0.8038●		0.7433●		0.7498●		0.7946●		0.8193		0.8053●		0.8125		0.8192	
lymph	3	0.7815	0.7903	13.5	0.7653●	10	0.7522●	6	0.7678●	10	0.7793	34	0.7884	34	0.7971○	33.73	0.7926○	26.37
148, 68	4	0.8441	0.7992●		0.7544●		0.7704●		0.7876●		0.8188●		0.8323●		0.8304●		0.8438	
	2	0.9271	0.9472○		0.9271		0.9182		0.9046●		0.9271		–		0.9488○		0.9283	
mushroom	3	0.9645	0.9544●	12.3	0.9588	10	0.9182●	6.36	0.9471●	10	0.9644	59	–	–	0.9608	10.27	0.9674	20.7
8124, 119	4	0.9989	0.9611●		0.9583●		0.9256●		0.9780●		0.9968		–		0.9639●		0.9874●	
	2	0.8408	0.8408		0.7911●		0.7842●		0.8173●		0.8408		0.8408		0.8408		0.8408	
primary-tumor	3	0.8186	0.8221	13.73	0.7561●	10	0.8385○	6	0.8266	10	0.8185	15	0.8174	16	0.8186	18.53	0.8210	25.9
283, 31	4	0.8256	0.8397○		0.7656●		0.8314		0.8314		0.8397○		0.8456○		0.8315		0.8268	
	2	0.8842	0.8842		0.8576●		0.8661●		0.8762		0.8864		0.8938		0.8805		0.8842	
soybean	3	0.9232	0.9227	16.76	0.8613●	10	0.8778●	8.27	0.8809●	10	0.9205	25	0.9146	25	0.8949●	17.8	0.9237	25.87
625, 50	4	0.9413	0.9402		0.8645●		0.8735●		0.8869●		0.9376		0.9259●		0.9082●		0.9429	
	2	0.6736	0.6809		0.6753		0.6931○		0.6896○		0.6763		0.6739		0.6736		0.6736	
tic-tac-toe	3	0.7222	0.7191	14	0.7254	10	0.7163	6	0.7097●	10	0.7369○	13	0.7268	14	0.7222	18	0.7222	18
958, 27	4	0.8183	0.7991●		0.7525●		0.7505●		0.7515●		0.7957●		0.7995●		0.8201		0.8201	
	2	0.9564	0.9556		0.9548		0.9564		0.9564		0.9564		0.9517		0.9564		0.9564	
vote	3	0.9441	0.9518	11.6	0.9480	10	0.9502	6	0.9471	10	0.9464	24	0.9471	24	0.9433	16.56	0.9448	15.5
435, 48	4	0.9456	0.9502		0.9464		0.9425		0.9479		0.9464		0.9426		0.9456		0.9540	
	2	1.0	1.0		1.0		1.0		1.0		1.0		1.0		1.0		1.0	
zoo	3	1.0	1.0	1	1.0	10	1.0	6	1.0	10	1.0	18	1.0	18	1.0	1	1.0	2
101, 36	4	1.0	1.0		1.0		1.0		1.0		1.0		1.0		1.0		1.0	

○, ●: 1% improvement or degradation of accuracy compared to column "W/O FS", respectively.

Table 2: Experimental results on two-class binary datasets using CART

but Lasso maintains accuracy in more cases than RFE-SVM, and it performs better in preserving accuracy compared to the others. However, the wrapper methods and RFE-SVM seem to struggle with high-dimensional datasets in terms of both dimensionality and sample size. The bar chart in Fig. 4 shows the comparison times of different FS methods across five largest datasets (in terms of dimensionality and sample size), presented from left to right: "german-credit", "hypothyroid", "kr-vs-kp", "mushroom", and "dorothea". For each dataset, the methods are listed in the following order on the x-axis from left to right: LAD-based, ReliefF, CFS, MI, SF-SVM, SB-SVM, RFE-SVM, and Lasso. The y-axis represents the time in seconds on a logarithmic scale. As we can see, SF-SVM and SB-SVM require a significant amount of time to generate a feature subset compared to the others, especially for "mushroom" and "dorothea" (the bars are missing as we could not report the time due to a significant amount of time needed to generate a feature subset in each iteration). LAD-based requires a similar duration to CFS but is slightly longer in most cases. Lasso requires the shortest amount of time compared to other methods. Surprisingly, for "dorothea", LAD-based requires a shorter amount of time compared to CFS and Lasso, despite its slight loss in accuracy of around 2%. In terms of the significant improvement, despite not having any interaction with the learning algorithms, our method outperforms filter methods (ReliefF and MI) and is competitive with CFS, wrapper and embedded methods. In terms of the average number of selected features, our approach selects fewer features compared to the wrapper and embedded methods across many datasets. This suggests that our approach can eliminate irrelevant and redundant features while preserving or enhancing the accuracy of DTs, making them optimal with respect to the minimum number of features sufficient for constructing such trees.

Instance $\#s, \#f_n, \#c$	Classifier	W/O FS	LAD-based		Filter Methods						Wrapper Methods				Embedded Methods			
					ReliefF		CFS		MI		SFS-SVM		SBS-SVM		RFE-SVM		Lasso	
		Acc	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$	Acc	$\#nbf$
	CART($d=3$)	0.7436	0.7634○ (SU)		0.6396●		0.6182●		0.7595○		–		–		0.6984●		0.7276●	
	SVM	0.8277	0.8357 (SU)	1	0.6778●	10	0.6428●	6	0.7492●	10	–	–	–	–	0.8309	81	0.8428○	1311.27
alon	KNN	0.8158	0.8357○ (SU)		0.6523●		0.6612●		0.7665●		–		–		0.8150		0.8261○	
62, 2000, 2	Naive Bayes	0.5849	0.8365○ (SU)		0.4928●		0.4793●		0.6889○		–		–		0.7087○		0.5857	
	CART($d=10$)	0.8919	0.7917● (SU)		0.5138●		0.6138●		0.9250○		–		–		0.9694○		0.9472○	
	SVM	0.6556	0.8251○ (SU)	1	0.5917●	10	0.6083●	6	0.9557○	10	–	–	–	–	0.7388○	23.9	0.6944○	327.96
borovecki	KNN	0.5917	0.8251○ (SU)		0.6112○		0.5917		0.9474○		–		–		0.6888○		0.5944	
31, 22283, 2	Naive Bayes	0.8361	0.8361 (SU)		0.6138●		0.7638●		0.9779○		–		–		0.8889○		0.9668○	
	CART($d=5$)	0.9291	0.9256 (PCC)		0.9262		0.9309		0.9279		0.9314		0.9344		0.9291		0.9373	
	SVM	0.9174	0.9402○ (PCC)	3.83	0.8969●	10	0.9004●	6	0.9180	10	0.9280○	15	0.9321○	15	0.9186	18.37	0.9174	17.2
breast-cancer-wisconsin	KNN	0.9309	0.9426○ (PCC)		0.9296		0.9244		0.9262		0.9356		0.9279		0.9308		0.9297	
569, 30, 2	Naive Bayes	0.9397	0.9238● (PCC)		0.9344		0.9297●		0.9209●		0.9274●		0.9373		0.9309		0.9373	
	CART($d=3$)	0.9075	0.8769● (PCC)		0.6666●		0.9239○		0.9490○		–		–		0.9181○		0.9269○	
	SVM	0.8109	0.8915○ (PCC)	2	0.7209●	10	0.8884○	6	0.9078○	10	–	–	–	–	0.9263○	44.64	0.8681○	2342.76
chowdary	KNN	0.9163	0.9232 (PCC)		0.7142●		0.9298○		0.9681○		–		–		0.9487○		0.9196	
104, 22283, 2	Naive Bayes	0.9142	0.8987● (PCC)		0.6789●		0.9518○		0.9460○		–		–		0.9239		0.9718○	
	CART($d=3$)	0.9030	0.9098 (PCC)		0.8981		0.8987		0.9049		0.8946		0.9179○		0.9080		0.9061	
	SVM	0.9148	0.9148 (PCC)	2	0.9148	10	0.9148	6	0.9148	10	0.9148	10	0.9148	10	0.9148	1	0.9148	19
climate-simulation-crashes	KNN	0.8938	0.9135○ (PCC)		0.8938		0.8969		0.8993		0.8938		0.9098○		0.9061○		0.8938	
540, 20, 2	Naive Bayes	0.8870	0.9148○ (PCC)		0.8185●		0.8061●		0.8709●		0.8098●		0.9148○		0.9148○		0.9030○	
	CART($d=3$)	0.7938	0.8123○ (SU)	3.97	0.8061○		0.8407○		0.7950		0.8012		0.8111○		0.7997		0.7938∅	
	SVM	0.6493	0.6098● (PCC)	6.47	0.6728○	10	0.7691○	6.77	0.6543	10	0.7975○	6	0.7901○	7	0.7691○	9.63	0.6493∅	12.73
heart-statlog	KNN	0.6654	0.7679○ (PCC)	6.47	0.7740○		0.8123○		0.6580		0.7886○		0.7987○		0.7629○		0.6654∅	
270, 13, 2	Naive Bayes	0.8407	0.8209● (PCC)	6.47	0.8407		0.8283●		0.8456		0.8111●		0.8271●		0.8493		0.8407∅	
	CART($d=20$)	0.8788	0.8143● (PCC)		0.8892○		0.8557●		0.8826		0.8804		0.8819		0.8783∅		0.8786∅	
	SVM	0.9295	0.7375● (PCC)	5.8	0.9021●	10	0.8030●	7	0.8881●	10	0.8628●	8	0.8762●	8	0.9295∅	16	0.9295∅	16
letter	KNN	0.9560	0.8364● (PCC)		0.9584		0.9000●		0.9529		0.9370●		0.9450●		0.9560∅		0.9560∅	
20000, 16, 26	Naive Bayes	0.6426	0.5603● (PCC)		0.6544○		0.5916●		0.6547○		0.6240●		0.6142●		0.6426∅		0.6426∅	
	CART($d=3$)	0.7715	0.7891○ (PCC)	3.2	0.7642		0.7791		0.7454●		0.7492●		0.7766		0.7926○		0.7701	
	SVM	0.7966	0.7929 (SU)	3	0.7941	10	0.8041	6	0.7890	10	0.7865●	22	0.7916	22	0.7877	10	0.7966	42.9
spectf	KNN	0.7356	0.7741○ (SU)	3	0.7632○		0.7990○		0.7577○		0.7492○		0.7880○		0.7864○		0.7394	
267, 44, 2	Naive Bayes	0.6857	0.7513○ (PCC)	3.2	0.5857●		0.7230○		0.6892		0.6396●		0.6809		0.7081○		0.6882	
	CART($d=3$)	0.7340	0.7629○ (PCC)	2.93	0.7416		0.7266		0.7366		–		–		0.7575○		0.6940●	
	SVM	0.7921	0.7903 (PCC)	2.93	0.7921	10	0.7863	6	0.7921	10	–	–	–	–	0.7923	23.26	0.7922	160.94
tian	KNN	0.8041	0.7708● (PCC)	2.93	0.7705●		0.7440●		0.7338●		–		–		0.7715●		0.7867●	
173, 12625, 2	Naive Bayes	0.7810	0.7614● (SU)	1	0.5738●		0.7114●		0.7053●		–		–		0.7443●		0.7392●	
	CART($d=15$)	0.6981	0.6099● (SU)	2.93	0.7084○		0.6670●		0.6607●		0.6867●		0.6896		0.7131○		0.7044∅	
	SVM	0.4983	0.6213○ (SU)	2.93	0.4964	10	0.4625●	6	0.4810●	10	0.6016○	9	0.6528○	9	0.5437○	17	0.4983∅	18
vehicle	KNN	0.6481	0.6013● (SU)	2.93	0.6283●		0.6411		0.6158●		0.6737○		0.6832○		0.6584○		0.6481∅	
846, 18, 4	Naive Bayes	0.4593	0.4841○ (PCC)	3.9	0.4825○		0.4243●		0.4156●		0.4691		0.4956○		0.4684		0.4593∅	
	CART($d=5$)	0.8952	0.9010 (PCC)		0.9192○		0.9084○		0.9082○		0.9174○		0.8821●		0.8952		0.8952∅	
	SVM	0.6873	0.8496○ (PCC)	3.65	0.6031●	10	0.6857	6	0.6836	10	0.9004○	6	0.8598○	7	0.7956○	10.56	0.6873∅	12.76
wine recognition	KNN	0.6989	0.8727○ (PCC)		0.8273○		0.7080		0.7133○		0.9361○		0.8706○		0.8488○		0.6989∅	
178, 13, 3	Naive Bayes	0.9776	0.9398● (PCC)		0.9419●		0.9472●		0.9587●		0.9476●		0.9212●		0.9473●		0.9739∅	
	CART($d=3$)	0.7107	0.7601○ (PCC)	2.87	0.6923●		0.7237○		0.7390○		0.6904●		0.7047		0.7057		0.7061	
	SVM	0.7597	0.7682 (SU)	2	0.7631	10	0.7631	6	0.7631	10	0.7614	16	0.7597	17	0.7631	29.5	0.7597	13.53
wpbc	KNN	0.7164	0.7391○ (SU)	2	0.7561○		0.6978●		0.7264○		0.7439○		0.7007●		0.7442○		0.7164	
134, 93, 2	Naive Bayes	0.6958	0.7581○ (PCC)	2.87	0.7182○		0.7385○		0.7000		0.7093○		0.6991		0.6851●		0.6994	

○, ●: 1% improvement or degradation of accuracy compared to column "W/O FS", respectively. ∅: no feature elimination

Table 3: Experimental results on numerical datasets using CART, SVM, KNN and Naive Bayes

7.2 Experiments on numerical datasets

We compare our approach with other SOTA FS approaches using CART, SVM, KNN and Naive Bayes. Their parameters are kept at their default values, except for the maximum depth of CART, we conducted preliminary experiments to determine a best depth $d \in \{3, 5, 10, 15, 20\}$ that obtains the highest accuracy for each dataset (e.g. if the best depth found for a dataset is 5, we take CART($d = 5$) for the comparison). The results are reported in Table 3. The first column shows the dataset's name, the size of dataset ($\#s$), the number of numerical features ($\#f_n$), and the number of classes ($\#c$). The second column represents ML classifiers. The fourth column displays the results of LAD-based, reporting accuracy with either PCC or SU. We conduct the experiments with both of them, and report the better one. We consider a difference of 1% in accuracy for comparison. As shown in Table 3, our approach either maintains or enhances the accuracy roughly around 70% of the overall cases across different datasets and ML classifiers. PCC achieves higher accuracy in more cases than SU, implying that PCC is better suited for calculating the coefficient of feature-feature correlation for the datasets used in the experiments, thereby retaining the highest quality feature subset at the top. According to the experiments, LAD-based outperforms filter methods in terms of improvement and with less degradation. Despite not having any interaction with the learning algorithms, it still enhances the accuracy in many cases compared to wrapper and embedding methods, albeit with more degradation in several cases. However, the wrapper methods seem to struggle with larger dimensional datasets, particularly those from *datamicroarray*. The bar chart in Fig. 4 shows the comparison times of different FS methods across four largest datasets in terms of dimensionality, presented from left to right: "alon", "borovecki", "chowdary", and "tian". For each dataset, the methods are listed on the x-axis from left to right: LAD-based, ReliefF, CFS, MI, RFE-SVM, and Lasso. The y-axis represents the time in seconds on a logarithmic scale. Note that for these four datasets, we could not report the accuracy and the time for SF-SVM and SB-SVM due to the significant amount of time required to generate a feature subset in each iteration. In Fig. 4, RFE-SVM requires a significant amount of time to generate a feature subset compared to other methods. LAD-based requires a similar duration to CFS but is slightly longer in most cases. Lasso requires the shortest amount of time compared to other methods. For the embedded methods, especially Lasso, we observed that they could not eliminate irrelevant or redundant features in "heart-statlog", "letter", "vehicle", and "wine recognition" datasets. Thus, evaluations with the original features yield the same accuracy as the col-

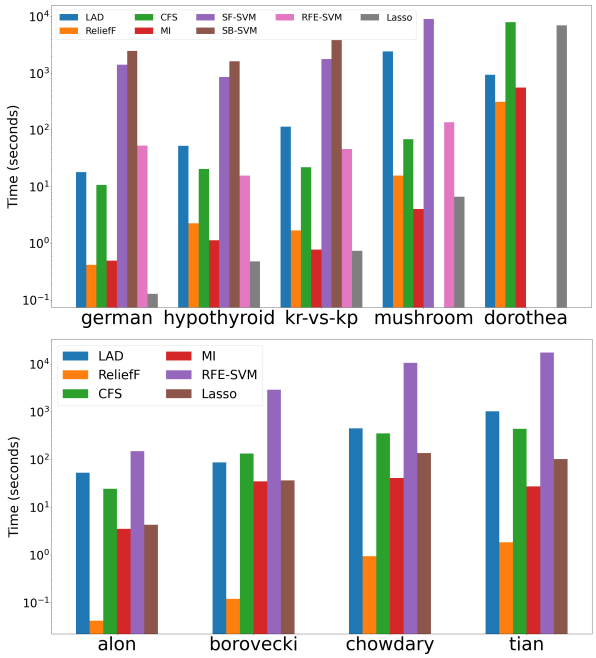

Figure 4: Time comparison between different FS techniques for binary datasets and numerical datasets

umn (W/O FS). In terms of the average number of selected features ($\#nbf$), LAD-based selects the fewest yet highest-quality features compared to the rest of the FS approaches. This suggests that LAD generates high-quality feature subsets, followed by an effective scoring method to select the highest quality one. Overall, this implies that while the majority of features are useful for predicting the target class, only a small subset is essential in practice for building an accurate and interpretable model.

8 Conclusion and Future Works

In this paper, we have presented a new feature selection (FS) approach based on the Logical Analysis of Data (LAD). To generate feature subsets, we first alleviate the complexity of the LAD optimization problem by converting it into the problem of enumerating minimal hitting sets in a hypergraph, for which efficient implementations exist. Those feature subsets are then ranked based on a scoring method to select the highest quality one. We then shed light on the relationship between optimal DTs and LAD-based FS, allowing us to construct the optimal DTs with respect to the minimum number of features. Experiments on benchmark datasets reveal that our approach is competitive with SOTA methods by selecting high-quality feature subsets that enhance or maintain the performance of DTs, SVM, KNN, and Naive Bayes.

In the future, it would be interesting to reduce the binary dataset resulting from numerical ones by considering alternative binarization techniques. Another research direction consists of exploring new scoring functions to better evaluate feature subsets for both binary and numerical datasets.

Acknowledgments

This work has benefited from the support of the region Haut de France and ANR HYCI project: ANR-22-CE55-0010.

References

Aglin, G.; Nijssen, S.; and Schaus, P. 2020. Learning Optimal Decision Trees Using Caching Branch-and-Bound Search. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(04):3146–3153.

Avellaneda, F. 2020. Efficient Inference of Optimal Decision Trees. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(04):3195–3202.

Bertsimas, D., and Dunn, J. 2017. Optimal classification trees. *Machine Learning* 106(7):1039–1082.

Bessiere, C.; Hebrard, E.; and O'Sullivan, B. 2009. Minimising Decision Tree Size as Combinatorial Optimisation. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, CP'09, 173–187. Berlin, Heidelberg: Springer-Verlag.

Bischl, B.; Casalicchio, G.; Feurer, M.; Gijsbers, P.; Hutter, F.; Lang, M.; Mantovani, R. G.; van Rijn, J. N.; and Vanschoren, J. 2021. OpenML Benchmarking Suites.

Boros, E.; Hammer, P.; Ibaraki, T.; Kogan, A.; Mayoraz, E.; and Muchnik, I. 2000a. An Implementation of Logical Analysis of Data. *IEEE Trans. Knowl. Data Eng.* 12(2):292–306.

Boros, E.; Hammer, P.; Ibaraki, T.; and Kogan, A. 2000b. Logical Analysis of Numerical Data. *Math. Program.* 79.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. Classification and Regression Trees.

Buyrukoğlu, S. 2021. New hybrid data mining model for prediction of Salmonella presence in agricultural waters based on ensemble feature selection and machine learning algorithms. *Journal of Food Safety* 41(4):e12903.

Chandrashekar, G., and Sahin, F. 2014. A survey on feature selection methods. *Computers Electrical Engineering* 40(1):16–28. 40th-year commemorative issue.

Cortes, C., and Vapnik, V. 1995. Support-Vector Networks. *Machine learning* 20(3):273–297.

Crama, Y.; Hammer, P. L.; and Ibaraki, T. 1988. Cause-effect relationships and partially defined Boolean functions. *Annals of Operations Research* 16:299–325.

Ferri, F. J.; Pudil, P.; Hatef, M.; and Kittler, J. 1994. Comparative study of techniques for large-scale feature selection. In *Machine intelligence and pattern recognition*, volume 16. Elsevier. 403–413.

Fredman, M. L., and Khachiyan, L. 1996. On the Complexity of Dualization of Monotone Disjunctive Normal Forms. *Journal of Algorithms* 21(3):618–628.

Gainer-Dewar, A., and Vera-Licona, P. 2016. The minimal hitting set generation problem: algorithms and computation. *CoRR* abs/1601.02939.

Ghiselli, E. E. 1964. Theory of psychological measurement. mcgraw-hill.

Guyon, I., and Elisseeff, A. 2003. An Introduction to Variable and Feature Selection. *Journal of machine learning research* 3(Mar):1157–1182.

Guyon, I.; Weston, J.; Barnhill, S.; and Vapnik, V. 2002. Gene Selection for Cancer Classification Using Support Vector Machines. *Machine Learning* 46:389–422.

Hall, M., and Holmes, G. 2003. Benchmarking Attribute Selection Techniques for Discrete Class Data Mining. *IEEE Trans. Knowl. Data Eng.* 15(6):1437–1447.

Hall, M. A. 1999. *Correlation-based Feature Selection for Machine Learning*. Ph.D. Dissertation, The University of Waikato.

Hall, M. 2000. Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. 359–366.

Hammer, P. L., and Bonates, T. O. 2006. Logical analysis of data - An overview: From combinatorial optimization to medical applications. *Ann. Oper. Res.* 148(1):203–225.

Hammer, P. L. 1986. Partially defined boolean functions and cause-effect relationships. *Proceedings of the International Conference on Multi-Attribute Decision Making via OR-Based Expert Systems*.

Hu, H.; Siala, M.; Hebrard, E.; and Huguet, M.-J. 2020. Learning Optimal Decision Trees with MaxSAT and its Integration in AdaBoost. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 1170–1176.

Hu, X.; Rudin, C.; and Seltzer, M. 2019. Optimal Sparse Decision Trees. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Hyafil, L., and Rivest, R. L. 1976. Constructing Optimal Binary Decision Trees is NP-Complete. *Inf. Process. Lett.* 5(1):15–17.

Khalid, S.; Khalil, T.; and Nasreen, S. 2014. A Survey of Feature Selection and Feature Extraction Techniques in Machine Learning. In *2014 Science and Information Conference*, 372–378.

Kraskov, A.; Stögbauer, H.; and Grassberger, P. 2004. Estimating Mutual Information. *Physical Review E* 69(6).

Makimoto, K.; Au, R.; Moslemi, A.; Hogg, J. C.; Bourbeau, J.; Tan, W. C.; and Kirby, M. 2023. Comparison of Feature Selection Methods and Machine Learning Classifiers for Predicting Chronic Obstructive Pulmonary Disease Using Texture-Based CT Lung Radiomic Features. *Academic Radiology* 30(5):900–910.

Matthews, B. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *BBA - Protein Structure* 405(2):442–451.

Mucherino, A.; Papajorgji, P. J.; and Pardalos, P. M. 2009. *k-Nearest Neighbor Classification*. Springer NY. 83–106.

Murakami, K., and Uno, T. 2014. Efficient algorithms for dualizing large-scale hypergraphs. *Discrete Applied Mathematics* 170:83–94.

Narodytska, N.; Ignatiev, A.; Pereira, F.; and Marques-Silva, J. 2018. Learning Optimal Decision Trees with SAT. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 1362–1368.

Nijssen, S., and Fromont, E. 2007. Mining Optimal Decision Trees from Itemset Lattices. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, 530–539.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Powers, D. M. W. 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *CoRR* abs/2010.16061.

Press, W. H.; Vetterling, W. T.; Teukolsky, S. A.; and Flannery, B. P. 1988. *Numerical recipes*. Citeseer.

Pudil, P.; Novovičová, J.; and Kittler, J. 1994. Floating search methods in feature selection. *Pattern Recognition Letters* 15(11):1119–1125.

Quinlan, J. R. 1986. Induction of Decision Trees. *Machine learning* 1:81–106.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Fran., CA, USA: Morgan Kaufmann Publishers Inc.

Ramey, J. 2016. Datamicroarray: collection of data sets for classification.

Robnik-Sikonja, M., and Kononenko, I. 2003. Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning* 53:23–69.

Ross, B. C. 2014. Mutual Information between Discrete and Continuous Data Sets. *PloS one* 9(2):e87357.

Sayed, S.; Nassef, M.; Badr, A.; and Farag, I. 2019. A Nested Genetic Algorithm for feature selection in high-dimensional cancer Microarray datasets. *Expert Systems with Applications* 121:233–243.

Schidler, A., and Szeider, S. 2021. SAT-based Decision Tree Learning for Large Data Sets. *Proceedings of the AAAI Conference on Artificial Intelligence* 35(5):3904–3912.

Tang, J.; Alelyani, S.; and Liu, H. 2014. Feature selection for classification: A review. *Data Classification: Algorithms and Applications* 37–64.

Tibshirani, R. 1996. Regression Shrinkage and Selection via the Lasso. *J. R. Stat. Soc. (Series B)* 58:267–288.

Verwer, S., and Zhang, Y. 2019. Learning Optimal Classification Trees Using a Binary Linear Program Formulation. *Proceedings of the AAAI Conference on Artificial Intelligence* 33(01):1625–1632.

Zhang, X.; Lu, X.; Shi, Q.; Xu, X.-q.; Leung, H.-C. E.; Harris, L. N.; Iglehart, J. D.; Miron, A.; Liu, J. S.; and Wong, W. H. 2006. Recursive SVM feature selection and sample classification for mass-spectrometry and microarray data. *BMC bioinformatics* 7:1–13.

Zhang, H. 2004. The Optimality of Naive Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004)*.