Journal Pre-proof

Analysis and comparison of feature selection methods towards performance and stability

Matheus Cezimbra Barbieri, Bruno Iochins Grisci, Márcio Dorn

PII: S0957-4174(24)00533-5
DOI: https://doi.org/10.1016/j.eswa.2024.123667
Reference: ESWA 123667

To appear in: *Expert Systems With Applications*

Received date : 8 October 2023
Revised date : 28 February 2024
Accepted date : 11 March 2024

Please cite this article as: M.C. Barbieri, B.I. Grisci and M. Dorn, Analysis and comparison of feature selection methods towards performance and stability. *Expert Systems With Applications* (2024), doi: https://doi.org/10.1016/j.eswa.2024.123667.

Highlights

Analysis and Comparison of Feature Selection Methods Towards Performance and Stability

Matheus Cezimbra Barbieri,Bruno Iochins Grisci,Márcio Dorn

- Extensive and complete review on feature selection algorithms and evaluation.
- Comparison of different stability metrics.
- A new Python framework for implementing and benchmarking feature selection.
- Evaluation of several feature selectors regarding many metrics.

Revised manuscript (with changes marked)

Analysis and Comparison of Feature Selection Methods Towards Performance and Stability

Matheus Cezimbra Barbieri[a], Bruno Iochins Grisci[a] and Márcio Dorn[a,b,c,*]

[a] Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, 90040-060, RS, Brazil
[b] Center of Biotechnology, Federal University of Rio Grande do Sul, Porto Alegre, 90040-060, RS, Brazil
[c] National Institute of Forensic Sciences, Porto Alegre, RS, Brazil

ARTICLE INFO

Keywords:
feature selection
dimensionality reduction
machine learning
classification
stability
tabular data

Abstract

The amount of gathered data is increasing at unprecedented rates for machine learning applications such as natural language processing, computer vision, and bioinformatics. This increase implies a higher number of samples and features; thus, some problems regarding highly dimensional data arise. The curse of dimensionality, small samples, noisy or redundant features, and biased data are among them. Feature selection is fundamental to dealing with such problems. It reduces the data dimensionality by selecting the most relevant and less redundant features. Thus reducing the computational cost, improving accuracy, and enhancing the data's interpretability to machine learning models and domain experts. However, there are several selector options from which to choose. This work compares some of the most representative algorithms from different feature selection groups regarding a broad range of measures, several datasets, and different strategies from diverse perspectives. We employ metrics to appraise selection accuracy, selection redundancy, prediction performance, algorithmic stability, selection reliability, and computational time of several feature selection algorithms. We developed and shared a new open Python framework to benchmark the algorithms. The results highlight the strengths and weaknesses of these algorithms and can guide their application.

1. Introduction

In contemporary business and scientific endeavors, the reliance on high-dimensional datasets has become paramount. These datasets often manifest as tabular data, where rows denote instances (i.e., sampled data points' values), and columns represent various features characterizing the sampled data. Applications span diverse domains, including the analysis of gene expression in cancer patients (Feltes et al., 2019, 2021), hemogram examination data from COVID-19 patients (Avila et al., 2020; Formica et al., 2020; Yan et al., 2020; Dorn et al., 2021), e-commerce sales (Sakar et al., 2019), and product defect detection (Pes, 2019). These large tabular datasets may encompass dozens to millions of columns, each representing a feature, input, or dimension (Grisci et al., 2021). While our focus in this work primarily centers on continuous or numerical features, it is worth noting that features can also be ordinal or categorical.

The ever-increasing volume of data poses significant challenges, rendering traditional processing methods impractical for many machine learning applications. Challenges associated with working with high-dimensional data include the curse of dimensionality, data imbalance, computational complexity, overfitting, and noisy or redundant data (Ang et al., 2015). Consequently, substantial research efforts have been directed towards dimensionality reduction techniques, notably feature extraction and feature selection.

*Corresponding author
mcbarbieri@inf.ufrgs.br (M.C. Barbieri); bigrisci@inf.ufrgs.br (B.I. Grisci); mdorn@inf.ufrgs.br (M. Dorn)
https://sbcb.inf.ufrgs.br/ (M. Dorn)
ORCID(s): 0000-0002-5389-7064 (M.C. Barbieri); 0000-0003-4083-5881 (B.I. Grisci); 0000-0001-8534-3480 (M. Dorn)

While feature extraction algorithms aim to project high-dimensional data into lower-dimensional spaces, feature selection techniques concentrate on identifying and selecting a subset of relevant features from the original dataset. In this paper, we delve into the latter aspect. Feature selection methods play a crucial role in mitigating several aforementioned challenges by reducing the number of attributes in the dataset, thereby providing machine learning models and domain experts with a more concise, relevant, and less noisy subset of features. These methods employ various strategies, metrics, and criteria to ascertain the importance of each feature. However, selecting an appropriate feature selection algorithm remains challenging due to the absence of a universally applicable metric or ground truth.

Addressing this challenge necessitates quantitatively evaluating feature selection methods based on various metrics. These metrics may include selection accuracy (indicating how effectively relevant features are chosen) and stability (assessing whether the selected feature subset remains consistent under slight variations in the input data, thereby enhancing the algorithm's reliability). In this work, we conduct a comprehensive comparison and evaluation of popular feature selection methods across diverse metrics, including selection prediction performance, accuracy, redundancy, stability, reliability, and computational efficiency. Additionally, we introduce an extensible framework designed to facilitate the setup, execution, and evaluation of these techniques.

The inspiration for this study is that the escalating volume of data across diverse domains necessitates ongoing research in feature selection. Our study is motivated by the pressing need to address the challenges posed by high-dimensional datasets and to provide robust solutions that

enhance the effectiveness and efficiency of feature selection methodologies. This paper makes several key contributions to the field of feature selection. Firstly, we provide a comprehensive review of classical feature selection methods, highlighting their properties, inner workings, and applications. Secondly, we propose a systematic, modular, and expandable evaluation framework that enables a thorough comparison of these methods across multiple metrics. Thirdly, we present empirical results that offer insights into the performance and behavior of different feature selection algorithms. Lastly, we provide open-access resources, including datasets and the Python code of the proposed framework, to facilitate reproducibility and further research in the field. Moving forward, it is imperative to expand and refine the proposed framework to accommodate evolving methodologies and address emerging challenges in feature selection. We also highlight that the framework we propose is independent of which feature selection method is being tested so that even the most recent developments can be compared to other methods. This possibility is one of the paper's main contributions, and we encourage future authors to compare their new feature selection algorithms regarding performance and stability. By providing access to both summarized data and the underlying framework, we aim to foster collaboration and advancement in feature selection research.

This text is organized as follows. Section 2: *Related Work* presents some works that provide some of the concepts and fundamentals used in our work. Subsequently, Section 3: *Methods and Concepts* introduces the concepts of feature selection, stability, prediction, and statistical methods and describes the algorithms that are fundamental to the understanding of this work. Then, Section 4: *Framework* shows the architecture, execution flow, and components of the proposed framework. Next, Section 5: *Methodology* describes the datasets used, how measurements were taken, and how experimentation was conducted. Afterward, Section 6: *Results and Discussion* reports the results obtained and describes the analysis of how different methodologies performed. Finally, Section 7 summarizes the most relevant results and observations regarding this work and discusses future improvements and possible research suggestions.

2. Related Work

Previous works have been published to survey and compare the characteristics of many feature selection algorithms. This section presents a brief review of some studies that are closely related to feature selection and, thus, to our work. Lazar et al. (2012) provide a complete survey on filter feature selection approaches. This work presents filter methods' fundamental notions, classifications, and evaluation techniques. Although Lazar et al. (2012) do not give empirical results, the proposed taxonomy is very informative and thus served as a good guideline for the development of filter feature selection procedures. Vergara and Estévez (2014) present another work that provides fundamental concepts to some of our work's feature selection methods. They supply the core notions of Information Theory and put forward a unified mathematical framework for information-based feature selection techniques. Molnar (2020) reviews some feature selection algorithms in the context of interpretable machine learning, presenting yet another framework for analysis (Grisci et al., 2021; Gill et al., 2022).

Both Ang et al. (2015) and Miao and Niu (2016) introduce the notions of supervised, semi-supervised, and unsupervised feature selection. While Miao and Niu (2016) focus solely on evaluating the accuracy of some of the novel unsupervised methods, Ang et al. (2015) go further in presenting fundamentals, theory, and survey of feature selection methods on gene selection. Ang et al. (2015) bring up various ways to classify attribute selection approaches and enlist their main advantages and disadvantages. Besides, they extensively review feature selection methods and individual reported accuracy results published in recent years. Ang et al. (2015) also discuss some of the problems that arise when working on gene selection, such as the curse of dimensionality, imbalanced data, and redundancy issues. Lastly, they state that most studies treat high classification accuracy as the ultimate goal. Still, more effort should be employed on research that assesses different evaluation measures and validation on feature selection.

Moreover, there are similar studies to Ang et al. (2015), put up by Bolón-Canedo et al. (2014) and Tadist et al. (2019). Their works also present a review of some of the most popular approaches to attribute selection and describe some of the gene microarray databases[1] most present in feature selection literature. Afterward, the classification and feature selection challenges on such databases are described. Finally, they show experimentation results by evaluating the accuracy, sensitivity, and precision of prediction on feature selection procedures. Grisci et al. (2023) offer a different perspective by critically reviewing the gene expression datasets used in the feature selection literature. Njoku et al. (2022) also compare the impact of filter feature selection on classification performance, and Cilia et al. (2019) focus specifically on microarrays.

Our work goes even further. We survey and compare several feature selection approaches regarding prediction accuracy and other aspects, such as the proportion of informativeness, redundancy, stability, and reliability in their results. Also, we compare the computational time that feature selection approaches take.

As mentioned above, another important topic discussed in this work is the stability and reliability of feature selection methods. Khaire and Dhanalakshmi (2019) and Nogueira et al. (2017) provide and describe the core concepts and desired properties of feature selection stability. Also, alongside Mohana Chelvan and Perumal (2016), they survey and describe some metrics that can assess stability for different formats of feature selection results. Some of the presented

[1] Gene Expression Microarrays, also known as DNA microarrays, are datasets that contain encoded gene expression, usually represented by 6000 to 60000 features, of samples from different individuals of the same species. Usually, the number of samples lies about a hundred (Bolón-Canedo et al., 2014).

metrics have also been used in our work. The work of Awada et al. (2012) reviews methodologies to evaluate stability and reduce instability. It shows ways to sample data to promote small perturbations in the dataset to evaluate stability. Furthermore, Pes (2019) is one of the few other studies that evaluate and compare prediction performance and stability between different feature selection approaches. The research and analysis strategy in their work is very similar to ours. However, their work focuses on comparing the non-ensemble version versus the ensemble version of some specific feature selection algorithms. In contrast, our work brings up a more general and extendable approach. Similarly, Salman et al. (2022) review the stability of aggregation techniques in ensemble feature selection.

Finally, few implementations of these studies are made public, so further extending research on these topics usually takes more effort than it could. One exception is the *featsel* framework by Reis et al. (2017). It is a C++ implementation of algorithms and cost functions for benchmarking feature selection. In our work, we intend to provide reference results of a comparison between popular feature selection approaches and an extensible Python framework to aid further research on the topic.

It is also important to highlight that the research, development, and application of new feature selection algorithms and methods are still relevant in machine learning and data science. New developments are constantly being published, and it would be out of the scope of this work to review all of them. However, as an overview of the field's current state, we comment on some recent works below.

Sheikhpour et al. (2023) discuss the importance of semi-supervised feature selection (SSFS) in utilizing both labeled and unlabeled data to choose informative features, mainly focusing on addressing the limitations of graph Laplacian (GL) regularization in preserving data structure and achieving extrapolation. To overcome these issues, the authors propose two frameworks: Hessian-based SSFS with generalized uncorrelated constraint (HSFSGU) and Hessian–Laplacian-based SSFS with generalized uncorrelated constraint (HLSFSGU). These frameworks utilize Hessian regularization for maintaining data structure, mixed-norm regularization for suitable feature selection, and the generalized uncorrelated constraint for preventing excessive sparsity. A unified algorithm with proven convergence is presented for solving convex and non-convex cases.

Saberi-Movahed et al. (2022) introduce DR-FS-MFMR, a novel approach for feature selection in gene expression data analysis. Traditional methods based on matrix factorization for dimensionality reduction have limitations in efficiency and reliability. DR-FS-MFMR addresses these shortcomings by discarding redundant features from the original set. It formulates the feature selection problem with two key aspects: matrix factorization and correlation information of selected features. The method enriches the objective function with two data representation characteristics and an inner product regularization criterion to improve redundancy minimization and sparsity.

Han et al. (2024) present a novel method, RRFS, designed to tackle the challenges of feature selection in multi-view, multi-label data. Existing methods often overlook balanced information gain across labels and feature overlap and fail to exploit correlations between views, features, and labels. RRFS addresses these issues by integrating feature relevance and redundancy terms, considering each label's information gain and feature redundancy within and between views. This approach aims to balance label information acquisition while reducing feature redundancy.

Gao et al. (2021) discuss the challenges posed by high-dimensional multi-label data and propose a novel approach, Shared Latent Feature and Label Structure Feature Selection, to address these challenges. Existing methods often overlook the impact of latent feature structure on label correlations. To overcome this limitation, the SSFS method incorporates a Latent Structure Shared (LSS) term, which preserves latent feature and label structures. Additionally, graph regularization ensures consistency between the original and latent feature structure spaces. The SSFS method, derived from the constrained LSS term, is optimized using an effective scheme with provable convergence.

Gao et al. (2023) introduce a novel feature selection framework, STFS, to address critical issues in high-dimensional multi-label data analysis. STFS incorporates two probability distribution assumptions based on low-order variable correlations and proposes a unified framework comprising three low-order information-theoretic terms. This framework effectively captures high-order variable correlations.

Although outside the proposed scope of this work, another important front in feature selection research is the use of artificial neural networks and deep learning. Neural networks have shown promise in automating feature selection design, offering potential improvements in efficiency and effectiveness, and the comparison of classical feature selection with deep learning approaches using the framework proposed in this work is a promising future work. Whiteson et al. (2005) introduced FS-NEAT, a method extending the NEAT algorithm, which not only evolves neural network topology and weights but also determines the appropriate inputs. This approach circumvents the need for meta-learning or labeled data, demonstrating superior learning capabilities and smaller network sizes across various tasks. Grisci et al. (2018) further extended FS-NEAT's applicability to microarray analysis, a domain characterized by high-dimensional data and limited sample sizes. Their adaptation showcased the method's ability to handle both classification and feature selection simultaneously, yielding biologically relevant gene biomarkers for diseases like leukemia.

Watts et al. (2019) addressed the challenge of evolving compact models for high-dimensional datasets with noisy features. They proposed Blocky Net, a minimalist neural network incorporating built-in feature selection. Comparative evaluations against NEAT and FS-NEAT highlighted its superior performance, particularly on large datasets, suggesting promising avenues for future research. In a similar vein,

Grisci et al. (2019) leveraged neuroevolution to tackle microarray data analysis, emphasizing the simultaneous classification and selection of relevant genes. They identified biologically significant genes associated with cancer types through rigorous preprocessing and algorithmic enhancements, underscoring the method's potential for biomarker discovery.

Recognizing the need for model interpretability, Grisci et al. (2021) introduced relevance aggregation, a technique for elucidating neural network predictions by identifying important features. Through empirical validation across diverse datasets, including gene expression and online shopping behavior, they demonstrated the method's efficacy in feature selection and model comprehension, facilitating knowledge discovery and mitigating machine bias. Overall, these studies collectively underscore the potential of neural networks, particularly neuroevolution-based approaches like FS-NEAT and Blocky Net, in automating feature selection across various domains while addressing challenges related to interpretability and model compactness.

3. Methods and Concepts

This section presents the fundamental concepts and methodologies used throughout this work. First, in Subsection 3.1 and Subsection 3.2, we discuss the kinds and specifics of the feature selection algorithms used and their properties, respectively. Subsection 3.3 reviews the predictive models employed in the study. The metrics to assess prediction are explained in Subsection 3.4. Several statistical methods and concepts are substantiated in Subsection 3.5 and Subsection 3.6. Afterward, all the evaluated feature selection algorithms are introduced, followed by all the measurements taken into account, in Subsection 3.7. The concept of algorithmic stability is introduced in Subsection 3.8. Finally, the visualization techniques utilized in this work are described in Subsection 3.9. Figure 1 shows an overview of the concepts we will introduce here.

3.1. Feature Selection

High-dimensional datasets can often lead to problems when training machine learning models, such as the curse of dimensionality, the small sample problem, too imbalanced data, infeasible computational times, model overfitting, and non-relevant or redundant data (Ang et al., 2015). Given the necessity of working with such data types, many dimensionality reduction techniques have been studied over the past few years.

Feature Selection consists of selecting a smaller subset of features from the original data. These subsets are created based on some property or satisfying some criteria, such as maximizing prediction accuracy, informativeness, relevance, or minimizing redundant and noisy data. In contrast to feature extraction, feature selection does not transform nor mix pieces of information, thus providing much more interpretable results. For example, understanding which genes are differentially expressed for a genome is possible with feature selection but not feature extraction.

Regarding labeling information, feature selection algorithms can be roughly classified as supervised, unsupervised, or semi-supervised (Miao and Niu, 2016). In the presented work, we focus on evaluating algorithms that can perform supervised tasks, which only work for datasets in which every instance has been labeled beforehand. Classifying feature selection algorithms by their search strategy is also ubiquitous in the literature. According to Ang et al. (2015), algorithms are often grouped as filter, wrapper, embedded, hybrid, or ensemble methods. Besides labeling and search strategy, algorithms can be further categorized in terms of statistical analysis, amount of correlation between features, determinism, result type, and other properties that go beyond the scope of this work and so will not be taken into account.

3.1.1. Filter

Filter methods are the earliest approach to feature selection. They are algorithms that rely only on intrinsic data characteristics before any learning task, therefore having a better generalization property. They are mainly sub-grouped in ranking or space search methods (Lazar et al., 2012).

Ranking algorithms are usually simple, fast, and occur in two steps. Firstly, a relevance score is generated for every feature, often based on dependency, consistency, information, or statistical relevance. Finally, a subset of the highest-ranked features is selected. Later we will discuss more about ranking, with two ranking filter methods: *Kruskal Wallis H test Filter*, in Subsection 3.5.1, *Mutual Information Filter*, in Subsection 3.6.2, and ReliefF algorithm, presented ahead in Subsection 3.7.4.

The other group of filter techniques is space search methods. This group transforms the feature selection problem into an optimization problem with some defined cost function. A space search approach is then used to select the subset of features that optimizes the given cost function. This work evaluates Minimum Redundancy Maximum Relevance (mRMR), a space search approach later described in Subsection 3.7.3.

3.1.2. Wrapper

Like space search filtering methods, wrapper techniques also perform a space search. The main difference lies in the scoring function. In filtering approaches, the cost function is defined in terms of data intrinsic characteristics. In contrast, in wrapper approaches, the cost function is based on the predictive error of a single classifier.

This strategy often leads to excellent, if not optimal, selections to the wrapped classifier model because it tries to minimize the prediction error while considering the non-linear relation between features. Nevertheless, the main drawback of wrapper methods is that their cost function relies on training and predicting at least one model per iteration (Ang et al., 2015). Thus, this approach has a high computational cost. Also, wrappers are classifier-specific. Therefore, they may not yield satisfactory results if the resulting selection is used with different classifiers other than the one used in training and become very prone to overfit.

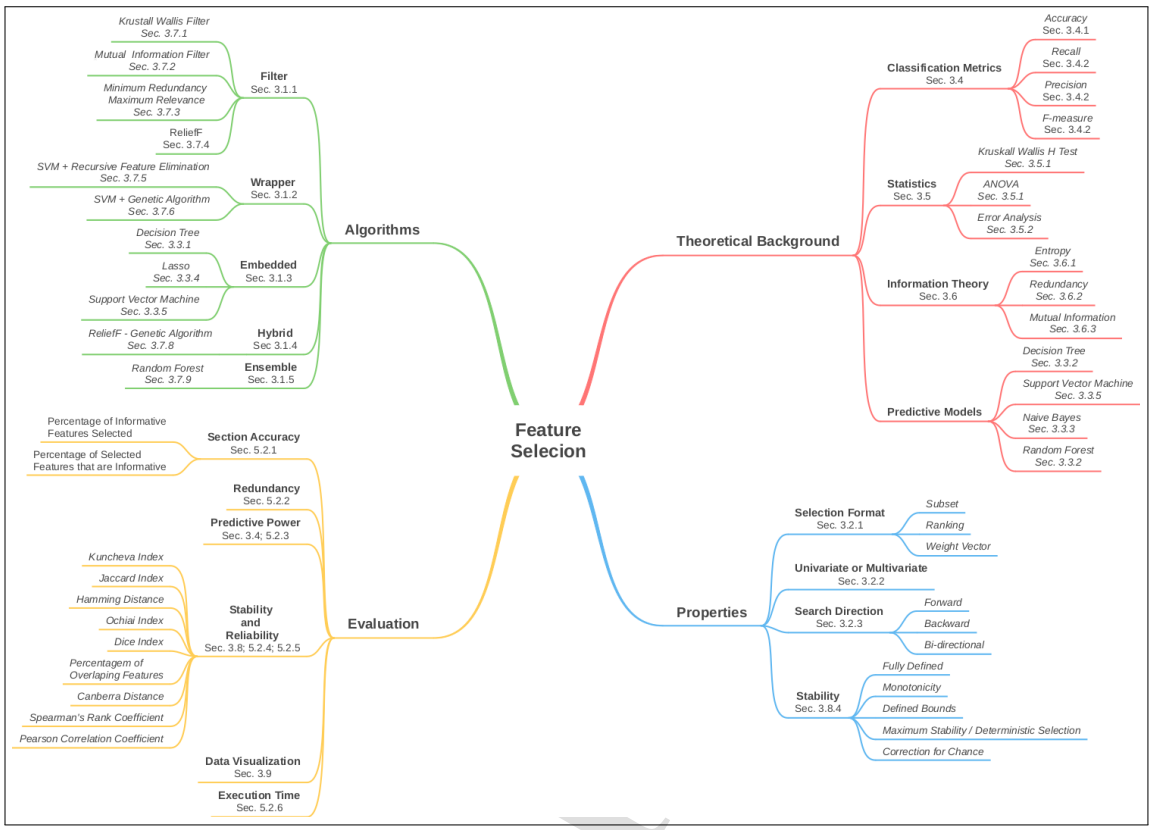


Figure 1: Structure of concepts and methods discussed in Section 3 and 5. This figure displays the taxonomy of the algorithms, evaluation metrics, theoretical background, and selection properties of feature selection. The text sections are indicated by the corresponding names.

Here, we evaluate the following wrapper algorithms' approaches: *SVM Recursive Feature Elimination* and *Genetic Algorithm as SVM wrapper*. Both methods are later shown in more detail in Subsection 3.7.5 and 3.7.6, respectively.

3.1.3. Embedded

This family of methods utilizes built-in model characteristics to select features. The most common approach is to rank features based on the weights a particular classifier attributes to each attribute. This approach suffers from the same problem of being specific to a single classifier (Ang et al., 2015). However, it does not require expensive computation as wrapper methods and, yet, tends to yield higher performance in classification accuracy than filter approaches. We evaluate three different predictors that perform embedded selection: *Decision Trees*, *Lasso*, and *Linear SVM*. All three predictors are forward defined in Subsection 3.3.

3.1.4. Hybrid

Hybrid methods are generated by combining any feature selection techniques, such as a filter and a wrapper. This differs from an embedded method because, in embedded selection, the ranking of the features is part of the inner workings of the classifier and not a separate algorithm. The main goal of hybrid methods is to leverage each method's complementary strengths, usually achieving better computational efficiency than wrapper and yet resulting in more general subsets with satisfactory prediction results (Ang et al., 2015). We test a mixture of ReliefF and Genetic Algorithm as SVM wrapper approaches, further described in Subsection 3.7.8.

3.1.5. Ensemble

Ensemble methods are methods based upon the concept of the wisdom of the crowds. The most common approach is to run a single feature selection method on a different number of feature-wise subsamples of the training data to generate additional subsets of selected features. Then, obtained subsets are aggregated into a single subset, which typically leads to good approximations of the optimal subset. This approach's main strength is that it is very robust for stability and reliability issues, as it is suitable for high-dimensional data. We use the Random Forest as an ensemble feature selector (Subsection 3.7.9).

Table 1
Feature selectors properties.

Algorithm	Algorithm Type	Selection Format			Attributes Correlation		Search Type
		Subset	Rank	Weights	Univariate	Multivariate	
KW Filter	Filter			✓	✓		-
MI Filter	Filter			✓	✓		-
mRMR	Filter			✓		✓	Additive
ReliefF	Filter			✓		✓	-
SVM-RFE	Wrapper		✓			✓	Subtractive
SVM-GA	Wrapper	✓				✓	Both
Decision Tree	Embedded			✓		✓	-
Lasso	Embedded			✓		✓	-
Linear SVM	Embedded			✓		✓	-
ReliefF-GA	Hybrid	✓				✓	Both
Random Forest	Ensemble			✓		✓	-

3.2. Properties of Feature Selectors

In this subsection, some properties that feature selectors may hold are described. Table 1 links these properties presented below with the feature algorithms that are later described (Subsection 3.7).

3.2.1. Selection Format

There are some formats that feature selectors may return concerning their results. They can be **subsets** when the selector can only return a vector of indexes to the selected features. Subsets are usually generated by stochastic algorithms that evaluate diverse subset alternatives through the feature selection process, such as genetic algorithms (described in Subsection 3.7.6). The selection format can also be a **rank** of the attributes, which has two possible representations: an **ordered list** of the feature indexes, which was sorted by some decision criteria, or a **permutation list**, that each position in the vector represents a feature and the value is its current position at the rank. A more in-depth discussion about rank representation can be found in the works by Grisci et al. (2022) and Lin (2010).

Finally, results can be feature **weights**, representing the relevance scores that the feature selector attributed to the features. Weight vectors can be used to generate ranked results by ordering the features by their respective weights. Also, ranked results can be seen as subsets by removing the notion of ordering. The way back from subset to ranked list and from ranked list to weight vector is not possible.

The selection format directly impacts how some stability metrics (described in Subsection 3.8) can be measured. Some metrics can only deal with the notion of weights (Subsection 3.8.3), so they can not evaluate selections of algorithms that yield rankings or subsets. Just as well, several metrics can only deal with rankings (Subsection 3.8.2) and can not evaluate selections of algorithms that are subsets.

3.2.2. Univariate or Multivariate

Correlation between two features means that one feature is dependent on the other. In the context of feature selectors, regarding correlation, we can classify methods as *univariate* or *multivariate* (Lazar et al., 2012).

Univariate methods are the ones that evaluate each feature's correlation to the target class independently from other features. The main advantage of these methods is that they are usually straightforward and fast. However, they may not be the most accurate techniques as they do not consider feature synergy.

In contrast to univariate methods, multivariate methods are approaches that account for the correlation between features. They often are more complex than univariate techniques. Nevertheless, they are also more robust. We further explore these concepts when experimenting with the XOR dataset (described in Subsection 5.1.1), which has a strong non-linear feature correlation, and Synth_B (described in Subsection 5.1.2), which contains redundant features.

3.2.3. Search Direction

Search direction indicates in what order feature selectors achieve a relevant subset regarding adding or removing features from the candidate subset. Feature selection can be *forward*, *backward*, *bi-directional*, or none of the before (Ang et al., 2015). *Forward* feature selection means that the algorithm starts with an empty set of features and iteratively adds features to this subset. Conversely, starting with the complete set of features and subtracting some at every algorithm iteration is also possible. This process is called *backward* selection. Besides, some strategies employ both addition and subtraction of features simultaneously. Those are called *bi-directional*. Finally, some algorithms perform neither addition nor subtraction of attributes but act by giving weights to all features simultaneously.

3.3. Predictive models

Predictive models or predictors are machine learning algorithms that aim to predict outcomes or classify new data samples concerning previously seen data. Here, we briefly review some of the predictors and metrics we use to perform feature selection in Subsection 3.7 and to evaluate prediction results, described in Subsection 5.2.3.

3.3.1. Decision Tree

This is a straightforward model of a tree in which every node, in a top-down manner, represents a decision path that can be taken based on certain conditions (the values of a sample attribute in prediction), and every leaf node represents some label for which that path leads. As presented by Loh (2011), tree generation algorithms recursively build the tree from top to bottom, generating splits (decisions) at each node based on an impurity function for attributes, thus generating scores for the attributes it considers discriminant.

3.3.2. Random Forest

Random Forest is an ensemble algorithm that leverages the concept of the wisdom of the crowd to minimize the variance in results produced by single classifiers. The random forest concept was first introduced by Breiman (2001). It is formed by a collection of decision trees trained on different subsets of instances by bootstrapping (random sampling of instances). For a single instance, the prediction process occurs for every tree in the forest, and then the predicted label is decided through a majority voting system.

3.3.3. Naïve Bayes

Naïve Bayes is a probabilistic classifier method based on Bayes' theorem under the "naïve" assumption that every pair of features is independent. It is a straightforward and efficient approach. It relies solely on estimating the maximum likelihood for each feature and *a priori* probability for each class (Rish et al., 2001). Then, it selects the class that better explains the instance being predicted, e.g., the one that maximizes the *a posteriori* probability for class y for the given x instance, as defined in Equation 1.

$$y_{MAP} = \underset{i}{\operatorname{argmax}}\, P(y_i \mid \mathbf{x}) \tag{1}$$

3.3.4. Least Absolute Shrinkage and Selection Operator

This model, also known as Lasso, is a linear regression method that uses $l1$-regularization to improve prediction accuracy and reduce overfitting. Its formulation consists of solving the formulation in Equation 2.

$$\min_{\beta_0,\beta} \left\{ \frac{1}{N} \|y - \beta_0 - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\} \tag{2}$$

Where y is the outcome, X is the independent variables, N is the number of instances in the data, β is the unknown parameters to be calculated, and λ is the regularization term.

3.3.5. Support Vector Machine

Support vector machines (SVM) are a powerful and efficient class of regressors and classifiers. The method constructs an n-dimensional hyper-plane or hyper-plane set that separates the instances by class with the highest possible margin, maximizing the distance from the hyper-planes to their closest instances from each class (Cortes and Vapnik, 1995). It uses the hinge loss function to the soft-margin approach (when data is not necessarily linearly separable) and applies Thikonov regularization, also known as L2-regularization. Computing the classifier is equivalent to minimizing the expression in Equation 3.

$$\left[\frac{1}{n} \sum_{i=1}^{N} \max\left(0, 1 - y_i \left(\vec{w} \cdot \vec{x}_i - b\right)\right) \right] + \lambda \|\vec{w}\|^2 \tag{3}$$

Where $\vec{w}$ is the normal vector to the hyper-plane, $\frac{b}{\vec{w}}$ is the plane offset to from origin along $\vec{w}$, n is the number of instances in the data, x_i is i-th instance in the data, and y_i is every x_i class, which can be either 1 or -1, each representing one class.

3.4. Classification Metrics

To analyze how good a selection of a feature selector is, one of the measures that are considered in this work is how good such selection is when used to train machine learning models. This subsection describes some of the most simple and popular approaches to evaluating some binary predictor's quality and, therefore, evaluating the relevance of the selected feature subset regarding this predictor.

3.4.1. Accuracy

Sometimes addressed as Rand Index or Rand Accuracy, prediction accuracy represents the percentage of correct predictions within all predictions (Powers, 2020) and can be calculated as shown in Equation 4.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Where TP is the count of true positives, TN is the count of true negatives, FP is the count of false positives, and FN is the count of false negatives.

This metric can range from zero to one, where one means all predictions are correct, and zero means the opposite. Accuracy is effortless to calculate, though one of its main drawbacks is that it falls short in terms of confidence when the data is too unbalanced[2].

3.4.2. F-measure

Precision is a metric that quantifies how many instances predicted as positive are actually positive (Powers, 2020). It can be calculated as follows in Equation 5.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Where TP is the count of true positives, and FP is the count of false positives.

Recall, also known as sensitivity or true positive rate of the model, is a metric that quantifies how much among the actually positive samples were predicted as positive (Powers, 2020) and is given by Equation 6.

[2] An unbalanced dataset is a dataset in which the number of instances of each class differs significantly.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

Where TP is the count of true positives, and FN is the count of false negatives.

Recall and precision are relevance measures for machine learning and statistical models. F-measure, also known as F-score, is the harmonic mean between recall and precision (Powers, 2020). It is defined in Equation 7.

$$F_{score} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \tag{7}$$

Where β is a parameter to regulate the weight of recall versus precision.

Like accuracy, recall, and precision, the F-measure is also bounded between zero and one, one being the best possible score and zero the worst. Both recall and precision also suffer from the same problem of being biased by unbalanced data; thus, the F-measure is affected, too. Also, it is essential to notice that the F-measure, even though it is a useful metric for measuring balanced data, does not consider the number of true negatives. That is, it is also biased towards the positive class.

3.5. Statistical Methods

This subsection introduces one-way analysis of variance (ANOVA), its non-parametric version (Kruskal Wallis H test), and how to address errors between continuous variables. The *Kruskal Wallis H test* will later be used as part of a filter feature selection approach in Subsection 3.7.1. In contrast, error analysis will be used in Subsection 6.4 to compare the error between stability estimators.

3.5.1. ANOVA and Kruskal Wallis H test

The one-way analysis of variance is a class of statistical methods that uses the concepts of variation among and between groups (within a random variable and compared to others, respectively) to compare whether a group of random variables comes from the same distribution. ANOVA can be generally defined as described in Equation 8.

$$F = \frac{between\ group\ variation}{within\ group\ variation} \tag{8}$$

In this work, we utilize the *Kruskal Wallis H test*, which is the non-parametric equivalent of ANOVA. It does not assume that the data comes from a normal distribution. This test is defined in Equation 9.

$$H = \left(\frac{12}{n(n+1)} \sum_{j=1}^{k} \frac{R_i^2}{n_i} \right) - 3(n+1) \tag{9}$$

Where k is the number of compared groups, n is the total number of samples, n_i is the number of samples in the i-th group, and R_i^2 is the sum of the values in the i-th group.

3.5.2. Error Analysis

One way to compare predictors and estimators is to calculate how far their results are from each other. This can be addressed by accounting for the total error between them. Here is presented *Mean Absolute Error*, which will be used further to analyze the stability results in Subsection 6.4.

Mean Absolute Error (MAE) is one of the most popular ways to calculate the difference between two continuous variables. It is a simple way to measure the error while preserving its magnitude (Willmott and Matsuura, 2005). The MAE between two continuous variables x and y is defined by Equation 10.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i| \tag{10}$$

Where n is the total number of instances, x_i is the i-th value in the variable x, and y_i is the i-th value in variable y.

3.6. Information Theory

Information theory is a branch of applied informatics that studies how information is stored, encoded, and transmitted. This subsection describes Entropy, Mutual Information, and Redundancy, which are core concepts for some of the feature selectors tested further in this work.

3.6.1. Entropy

One of the fundamental matters of information theory is called Entropy. Vergara and Estévez (2014) define Entropy (H) as the measure of uncertainty that a random variable holds towards the probability of an event's occurrence. A random variable's entropy $H(x)$ is formulated in Equation 11.

$$H(x) = -\sum_{i=1}^{n} P(x_i) \log_2(P(x_i)) \tag{11}$$

Where n is the total number of instances, x is a random variable, x_i is the i-th value in the variable x, and $P(x_i)$ is the probability of x_i occurring.

3.6.2. Mutual Information

Derived from the concept of Entropy, Mutual Information (MI) measures how much information one random variable has concerning another variable (Vergara and Estévez, 2014). It is defined in Equation 12.

$$I(x; y) = \sum_{i=1}^{n} \sum_{j=1}^{n} P(x_i, y_j) \cdot \log \left(\frac{P(x_i, y_j)}{P(x_i) \cdot p(y_j)} \right) \tag{12}$$

Where n is the total number of instances, x and y are random variables, x_i and y_i are the i-th value in the variable x and y, respectively, $P(x_i)$ and $P(y_i)$ are the probabilities of x_i and y_i occurring, respectively, and $P(x_i, y_i)$ is the joint probability of x_i and y_i occurring.

3.6.3. Redundancy

The concept of redundancy of a variable f regarding a set of other variables S can be defined simply as the total mutual information, described in Subsection 3.6.2, from that variable to the set of attributes, as shown in Equation 13.

$$R(f, S) = \frac{1}{|S|} \sum_{x_i \in S} I\left(f, x_i\right) \tag{13}$$

Where f is a random variable, S is a set of random variables, x_i is the i-th value in the variable x, and $I(x, f)$ is the Mutual Information between x and f.

3.7. Feature Selectors

Throughout this subsection, all the core methods of feature selection we have experimented with in this work and some of their properties are described. Table 1 shows how each algorithm relates to each property.

3.7.1. Kruskal Wallis Filter

In the context of feature selection, we want to select the features that best discriminate the target class, so, for each feature, we calculate the score yielded by the Kruskal Wallis H test (described in Subsection 3.5.1) applied to the feature values grouped by class. Next, we select the features with higher test values (Grisci et al., 2019). As the test is applied to every attribute alone regarding the target class, this method is univariate and thus does not consider any relation or dependency between features. Also, as stated before, it yields weighted results.

3.7.2. Mutual Information Filter

Here, we utilize the concept described in Subsection 3.6.2 as a ranking filter approach. With mutual information, it is possible to quantify a feature's relevance based on how much information it holds concerning the target class. Therefore, we can rank the features according to the obtained relevance. Similarly to the Kruskal Wallis filter, this method only looks at features individually without considering dependency among them and thus is a univariate approach. Mutual Information yields weighted results.

3.7.3. mRMR

Sometimes, we want to identify how much a new variable can contribute by adding it to a set of other variables. One way to do this is by determining how relevant and non-redundant the new variable is. The Minimum Redundancy Maximum Relevance (mRMR), presented by Peng et al. (2005), is a space search filter algorithm that selects, iteratively, the features that maximize mutual information (described in Subsection 3.6.2) to the target class and minimize the redundancy (described in Subsection 3.6.3) regarding all the features already in the selected subset. So, at every iteration, it aims to choose the feature x_j that maximizes the Equation 14.

$$\max_{x_j \in X - S} \left[I\left(x_j; c\right) - R(x_j, S)\right] \tag{14}$$

Where X is the set of all attributes, S is the set of already selected features, I is the mutual information function, and R is the redundancy function. This work uses a fixed number of K-selected features as the algorithm's stopping criteria. This approach yields a weighted ranking of the K-best features.

3.7.4. ReliefF

ReliefF is a popular choice of algorithm for feature selection. Its main strength is its core idea: for every instance in the dataset, it looks for the k-neighbor instances of each class. Then, it weights how much each attribute differs between instances, thus being a lightweight algorithm that takes into account feature-correlation (Robnik-Šikonja and Kononenko, 2003).

The algorithm initiates a vector of weights that represents the quality of each feature. Then it selects an instance and looks for the other k-nearest[3] instances for each class. The instances with the same class of the target instance are called *hits*, and the instances with different classes are called *misses*. The intuition is that if the target instance's value and a *miss* in a given attribute A differs, A is a good feature to separate these classes, and thus its weight must be increased. Similarly, if the target instance's value and a *hit* in the attribute A differs, then A is a good feature to separate these classes, but it should not be because they have the same class. Thus, the weight for feature A must be decreased.

Misses are factored by their class's occurrence in the dataset, denoted as $P(c)$, so all the *misses* can equal in weight to *hits*. The pseudo-code for ReliefF is presented in Algorithm 1. The return of this algorithm is a vector of weights representing feature relevance.

3.7.5. SVM Recursive feature elimination

Recursive feature elimination is a wrapper approach that recursively eliminates features from the subset of all features based on some criteria. The procedure is described in Algorithm 2. We utilize the *SVM-RFE* method (Guyon et al., 2002). The criteria of feature removal are based on selecting the feature with the SVM predictor's lowest weight (Subsection 3.3.5) internally to the features. This method is multivariate since the SVM classifier considers all points and is very stable since SVM's optimization function is convex. In other words, with enough time, it always converges to the same result for given inputs. The return of the algorithm is a ranked list of all features.

[3] k is a user-defined parameter.

Algorithm 1 ReliefF Pseudo-code

Inputs : **R**: Set of all instances, **C**: Set of classes, **k**: Number of neighbors
Output: **W**: Vector of feature weights

$n \leftarrow |R|$
$W \leftarrow vector\_of\_zeros(size = |R_1|)$
for $i = 1$ **to** $|R|$ **do**
  $H \leftarrow k\_nearest\_hits(k, R_i, R)$
  for c ***in*** $C \backslash class\,(R_i)$ **do**
    $M_c \leftarrow k\_nearest\_misses(c, k, R_i, R)$
  end
  for $a \leftarrow 1$ **to** $|A|$ **do**
    for hit ***in*** H **do**
      $W[a] \leftarrow W[a] - \frac{diff(R_i, hit)}{n*k}$
    end
    for c ***in*** $C \backslash class\,(r)$ **do**
      for $miss$ ***in*** M_c **do**
        $d \leftarrow \frac{P(c)}{1-P(class(R_i))} * \frac{diff(R_i, miss)}{n*k}$
        $W[a] \leftarrow W[a] + d$
      end
    end
  end
end
return W

Algorithm 2 RFE Pseudo Code

Input : **S**: The set of all attributes
Output: **E**: reverse ordered ranked list of features

$E \leftarrow emptyList$
for $i \leftarrow 1$ **to** $|S|$ **do**
  $w = select\_worst\_attribute(S)$
  $S \leftarrow S\ remove\ w$
  $E \leftarrow E\ append\ w$
end
return E

3.7.6. Genetic Algorithm as SVM wrapper

Genetic algorithms are a class of evolutionary algorithms mainly used to solve optimization problems and perform space search in a heuristic manner (Diaz-Gomez and Hougen, 2007). The core concepts of genetic algorithms are motivated by natural selection over some generations of a population of individuals. Its leading operators are the *selection of the fittest*, *mutation*, and *crossover*.

In the context of wrapper feature selection, every individual in the population is a candidate subset of features. The algorithm starts by first creating the initial population of individuals. In our approach, we heuristically create the first population so that the size of the individual solutions equals the number of features to select. Population size is the minimum necessary to have every feature in the input data at least once. Then, the algorithm iteratively creates a new generation of individuals until some stop criterion is reached. At every iteration, the algorithm performs its three primary operations. It selects the fittest individuals in the population. Our strategy is that, given a fitness function, we apply the roulette wheel algorithm so that every individual can go to the next generation proportional to their fitness. Higher scores yield a higher chance for them to survive. We also make an elitist selection of the best 5% of individuals. These individuals are directly copied to the next generation without suffering crossover.

We perform the crossover operation over the selected solutions two by two. Our definition of crossover is that for every two random pairs of features formed from both parents' features, without replacement, there is a 50% chance, exclusively, for this pair to be passed on to their offspring. The mutation operator is applied to every individual in the resulting population. Our approach is: for every feature of an individual, there is a random chance of 0.1% for it to mutate into another random feature from a set of all features minus the selected ones. This work uses the genetic algorithm presented above as a wrapper feature selector for the SVM predictor (Subsection 3.3.5). This approach is inspired by the work of Frohlich et al. (2003), but the parameters were not explicitly tuned for each dataset; thus, results might not be optimal. The fitness function is the F-measure (Subsection 3.4.2) obtained by training the SVM model in a 5-fold cross-validation manner. Our stop criteria are either reaching 200 generations or a perfect classification score. The genetic algorithm pseudo-code is shown in Algorithm 3.

Algorithm 3 GA Pseudo-code

Inputs : **S**: The set of all attributes, **n_gen**: Maximum number of generations, **max_fitness**: Maximum fitness threshold
Output: **P**: The last population to be generated

$P \leftarrow create\_initial\_population(S)$
$fitness \leftarrow evaluate\_fitness(P)$
for $i \leftarrow 1$ **to** $n\_gen\ or\ max(fitness) == max\_fitness$ **do**
  $fittest \leftarrow selection(P, fitness)$
  $offspring \leftarrow crossover(fittest)$
  $offspring \leftarrow mutate(offspring)$
  $elite\_individuals \leftarrow elite(P)$
  $P \leftarrow offspring + elite\_individuals$
  $fitness \leftarrow evaluate\_fitness(P)$
end
return P

3.7.7. Embedded feature selection

As mentioned in Subsection 3.1, some predictors can perform feature selection at the same time they are trained. In this work, we compare embedded feature selection on Decision Tree (Subsection 3.3.1), Lasso (Subsection 3.3.4), and SVM (Subsection 3.3.5) predictors.

3.7.8. Hybrid ReliefF filter with Genetic Algorithm wrapper

This method is a hybrid approach that consists of combining two other methods already presented here: ReliefF (Subsection 3.7.4) and Genetic Algorithm (Subsection 3.7.6). As stated by Shreem et al. (2012), combining techniques can create a feature selector that takes advantage of both eliminating noisy features by two different criteria and also achieves good prediction power at lower computational cost by applying a wrapper algorithm on a smaller set of features. In our setup, we utilize ReliefF to filter the top 1000 features for every used dataset, except for the XOR dataset (Subsection 5.1.1) that contains fewer attributes. Then, we select smaller subsets of features with the genetic algorithm.

3.7.9. Random Forest feature selection

At last, we utilize the Random Forest (Subsection 3.3.2) to perform an ensemble feature selection. One of the biggest strengths of this method is that by utilizing an ensemble, instability in results tends to fall (Ang et al., 2015). This approach involves recombining the feature weights that every tree in the random forest calculated in a single averaged features weight vector so that features can be ranked.

3.8. Stability and Reliability

In machine learning, stability is a measure of how much the outputs tend not to change - that is, are stable - after small perturbations in the data. Data perturbation is defined as changes in the training samples and can be simulated by either adding new instances, removing some instances, or re-sampling instances (Awada et al., 2012). Stability plays an essential role in selecting an appropriate approach to feature selection because as stability increases, so does the confidence of domain experts regarding the analysis of selected features. Stability is closely related to another important concept called *reliability*. The main difference is that an algorithm is considered reliable if the outputs are stable for repeated applications to the same inputs without perturbation (Boutsidis and Magdon-Ismail, 2013).

Different estimators can be used to address and quantify the stability of a feature selection method. Mohana Chelvan and Perumal (2016) present three categories of stability estimators segregated by the kind of results they can evaluate: *Stability by Index/Subset*, *Stability by Rank*, and *Stability by Weight*. Nogueira et al. (2017) describe some desirable properties for such estimators, such as being fully defined, having maximum stability, defined bounds, correction by chance, and monotonicity.

3.8.1. Stability by Index/Subset

This kind of estimator can evaluate subsets of features. It relies solely on comparing similarity (the amount of overlap between sets of selected features); sometimes, they need the total number of features before selection. One advantage of these methods is that most can calculate the stability for sets of different sizes. Also, as weighted vectors and rankings can be seen as subsets (explained in Subsection 3.2), these metrics can evaluate a broader range of feature selectors' results. Ahead, in Subsection 3.8.5, we introduce the Jaccard Index, Hamming Distance, Percentage of overlap, Dice, Ochiai Index, and Kuncheva Index, all metrics for subsets.

3.8.2. Stability by Rank

As its name implies, stability by rank is applied to data that encodes a ranking of features. Data can be presented as an ordered list, where features are ordered according to their rank, or as a permutation list, where each position represents a feature, and its value on the list is equal to its respective position in the ranking. The latter format is more suitable for these estimators, as ordered lists do not store the position where an element is in the list. Most of these methods cannot deal with pairs of rankings of different sizes. In Subsection 3.8.5, the estimators Canberra Distance, Kendall's τ, and Spearman's ρ are described.

3.8.3. Stability by Weight

This kind of estimator accounts for the weights of the feature set in consideration to calculate stability. It is usually defined by a measure of the correlation between the two lists of weights. The main drawback of this approach is the inability to deal with pairs of weight vectors of different sizes. This is the most restricted group of estimators. It can only evaluate feature selectors that yield feature weights or scores as their result. Later in Subsection 3.8.5, Pearson correlation, a stability by weight estimator, is explained.

3.8.4. Stability Properties

The main approaches of stability calculation used in this work are described below. Table 2 shows which properties each metric satisfies and the type of results they can evaluate.

Fully Defined

An estimator is considered fully defined if it is able to deal with different sizes of selected subsets.

Monotonicity

This property states that the estimator should be a strict function in terms of sample variance. In other words, the larger the intersection ratio between the feature subsets, the more significant the stability should be.

Defined Bounds

This property means that stability estimations must be bounded by constants that depend neither on the overall number of features nor the number of selected features.

Maximum Stability ↔ Deterministic Selection

This implies that an estimator should achieve the maximum stability score if and only if all compared subsets are identical.

Correction for Chance

This property states that estimated stability should be constant in expectation under the null model hypothesis for any randomly selected subset from a set of features.

Table 2
Properties of stability measures. Original by Nogueira et al. (2017) and Khaire and Dhanalakshmi (2019). The "result type" column was complemented by the authors.

Metric	Fully Defined	Bounds	Maximum	Correction for Chance	Monotonicity	Result Type
Jaccard	✓	✓	✓		✓	Subsets
Hamming	✓	✓	✓		✓	Subsets
Dice	✓	✓	✓		✓	Subsets
Ochiai	✓	✓	✓		✓	Subsets
POG	✓	✓	✓		✓	Subsets
Kuncheva		✓	✓	✓	✓	Subsets
Canberra	✓	✓		✓		Rank
Spearman	✓	✓		✓	✓	Rank
Pearson	✓	✓		✓	✓	Weights

3.8.5. Stability Metrics

This subsection presents some of the metrics studied and implemented during the development and execution of this work. Table 2 links the metrics described here to some of their properties.

Jaccard Index

Jaccard Index is a similarity measure between two sets (Mohana Chelvan and Perumal, 2016). Jaccard yields values between zero and one. The value zero states no common element between the sets, while one says that both sets are precisely equal. For two sets, A and B, it is defined by Equation 15.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{15}$$

Normalized Hamming Distance

The Hamming distance between two sets is equal to the number of replacements needed to change one set to the other (Mohana Chelvan and Perumal, 2016). It is the size of the symmetrical difference between sets A and B divided by the size of their intersection. The normalized Hamming distance version for sets A and B is linearly inverse to Jaccard and can be calculated as defined in Equation 16. Just as Jaccard, results lay between zero and one, but in this case, zero means there is no distance between the sets, that is, they are identical, and one means they have no element in common.

$$NHD = \frac{|A \triangle B|}{|A \cup B|} \tag{16}$$

Percentage of overlap

This metric calculates the normalized number of overlapping attributes from one set to another (He and Yu, 2010). For two sets, A and B, it is calculated as in Equation 17. It is bounded between zero and one. The resulting value of one means that both sets have 100% overlap and are identical. Alternatively, zero means no overlap, and both sets have no element in common.

$$POF = \frac{|A \cap B|}{|A|} \tag{17}$$

Sørensen-Dice

Sørensen-Dice, also known as Dice or f1-score metric, quantifies the amount of overlap between two given sets. It is similar to Jaccard, and it is even possible to calculate Jaccard from Dice or vice versa. For two sets, A and B, Dice is defined as the harmonic mean of the overlap from A to B and from B to A (He and Yu, 2010). Its bounds are the same as in the percentage of overlap and have the same meaning. It can be calculated as in Equation 18.

$$Dice = \frac{2 * A \cap B}{|A| + |B|} \tag{18}$$

Ochiai Index

Ochiai Index, a very similar metric to Jaccard and Dice, also quantifies the overlap between two given sets (He and Yu, 2010). It is also possible to calculate it from the Jaccard Index score or vice versa. For two sets, A and B, the Ochiai index is defined as the geometric means of the overlap from A to B and overlap from B to A. Its bounds are the same as in the percentage of overlap and have the same meaning. It is calculated by Equation 19.

$$Ochiai = \frac{2 * A \cap B}{\sqrt{|A||B|}} \tag{19}$$

Kuncheva Index

Kuncheva Index, also known as consistency index, is a measure of similarity between two sets of selected attributes (Kuncheva, 2007). Unlike the metrics for subsets presented above, the consistency index satisfies the property of correction by chance. However, it can only be defined for subsets with the same size as a trade-off. Kuncheva is bounded between zero and one. The value zero means there is no common element between the two sets, whereas one means that the elements in both sets are equal. Its calculation can be defined as in Equation 20.

$$I_C(A, B) = \frac{|A \cap B|n - k^2}{k(n-k)} \tag{20}$$

Where n is the total number of features, and k is the cardinality of A and B. This equation is derived by applying a correction factor of $frack^2n$ on the amount of overlap between two sets of size k. This factor is defined as the expected size of the intersection between two randomly drawn subsets of size k from a subset of size n without replacement.

Canberra distance

Canberra distance is a measure of the distance between two ranked lists. It is defined as the sum of differences between the two lists divided by the sum of their position in the rank (Jurman et al., 2009). Its values start at zero and increase for every non-matching pair. For two given ranks, p and q encoded as permutation lists, Canberra distance is defined by Equation 21.

$$D_{canberra}(p, q) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|} \tag{21}$$

Where p_i is the position of the feature i-th feature in rank p, and q_i is the position of the feature i-th feature in rank q. The main advantage of this metric is that it attributes higher values for differences at the top of the ranking. When p_i and q_i are smaller, the feature in consideration is highly ranked. Hence, the ranking difference is divided by their sum, yielding higher values for more important features than those ranked at the bottom of the ranking.

Pearson correlation coefficient

Pearson correlation measures the linear correlation between two random variables. It is used to evaluate algorithm results that yield weighted feature subsets. It results in values between -1 and 1, with 1 indicating that both variables are linearly correlated and -1 meaning that they are linearly inverse correlated. Zero means that there is no correlation at all. Pearson correlation coefficient is defined in Equation 22.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{22}$$

Where x and y are a random variables, x_i and y_i are the i-th values in the variables x and y, respectively, n is the number of values in x and y, and $\bar{x}$ and $\bar{y}$ are, respectively, the mean value of variables x and y. The dividend of Equation 22 is the covariance between variables x and y, whereas the quotient is the standard deviation of x multiplied by the standard deviation of y.

Spearman's ρ

Spearman's rank correlation coefficient or Spearman's ρ is a measure of the correlation between two given random variables. Spearman's ρ between two ranks equals Pearson's correlation between the rank values of the two ranks, and thus, it has the same boundaries as Pearson's correlation. In the case where there are no ties in the ranking, Kalousis et al. (2007) define its formula as follows in Equation 23.

$$r_s(p, q) = 1 - \frac{6\sum_{i=1}^{n}(p_i - q_i)^2}{n(n^2 - 1)} \tag{23}$$

Where p and q are two rankings presented as permutation lists and n is the number of elements in the rankings.

3.9. Visualization

In this work, to better understand how much a subset of features is able to differentiate classes in a dataset, we utilize *t-SNE* (Maaten and Hinton, 2008; Grisci et al., 2021). The name *t-SNE* stands for t-distributed Stochastic Neighbor Embedding. It is a dimensionality-reduction feature extraction technique that projects a high-dimensional space into a faithful lower-dimensional space so that it can be visualized.

The intuition of *t-SNE* is that given the conditional probability of an instance being the neighbor to another in a high-dimensional space, for all instances, a space with fewer dimensions should be a good representation of this high-dimensional space if instances have a similar conditional probability of being neighbor to one another to the conditional probability the instances in the higher-dimensional space have. This algorithm's details go beyond this work's scope; more information about it can be found in Maaten and Hinton (2008).

4. Framework

This section presents an overview of the proposed computational framework supporting the experimentation conducted in this work. The following subsections describe the framework's architecture created to run and evaluate the feature selection algorithms, its execution flow, its main components, and the technologies used to build it. The complete code and data developed for this implementation can be accessed in the following repository and can be used for experiments or as a basis for other projects: https://github.com/sbcblab/GenExpFS.

4.1. Architecture

We create a software architecture based on workers to run and evaluate the algorithms in a fair computational environment. A manager process is responsible for spawning workers and distributing tasks to them. Every worker is spawned in a different process and attributed to one CPU core, so every task runs in a single-core process during its life cycle. All results generated during execution are saved to CSV files. Figure 2 shows the proposed architecture.

4.2. Execution flow

All the processing development is executed in a few operations. The steps are described below:

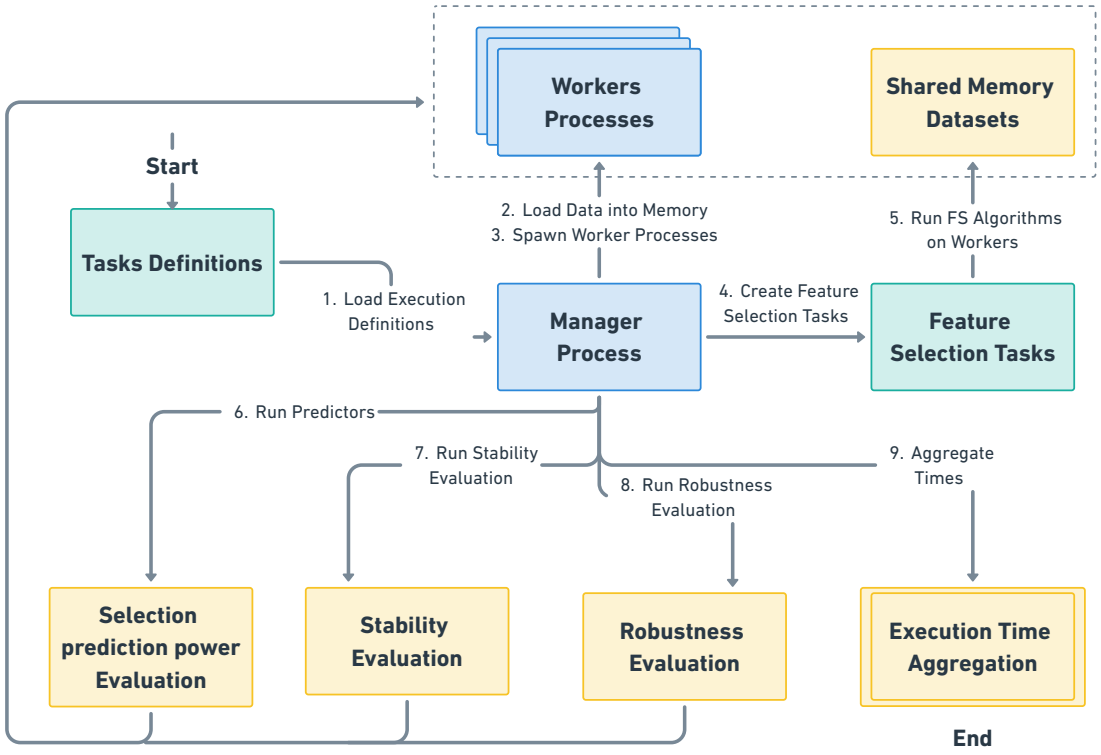


Figure 2: The pipeline architecture is built to perform feature selection and execute feature selection results evaluation.

1. **Load execution definitions**: First of all, our framework loads a set of configurations that describes which algorithms should be run, if they should be sampled or not, what their parameters will be, and for which datasets they will execute.
2. **Load data into memory**: In this step, all available datasets are loaded, preprocessed, and stored in a read-only shared memory.
3. **Spawn worker processes**: Here, we spawn several worker processes equivalent to the number of available CPU cores. Every worker is born with access to the shared memory and is ready to receive selection and evaluation tasks.
4. **Create feature selection tasks**: From the task definitions loaded before, algorithms presented in Subsection 3.7 and datasets described in Subsection 5.1, the manager process combines all the defined parameters to generate a list of the feature selection tasks that should be run.
5. **Run feature selection algorithms on workers**: Iteratively, all the tasks are sent to workers as soon as they are available to process them. Results are saved.
6. **Run predictors**: After the results of feature selections are ready, several prediction tasks, described in Subsection 5.2.3, are deployed to workers. Results are then saved.
7. **Run stability evaluation**: In this step, we first select the feature selection results that suffered data perturbation. Subsequently, we group the filtered results by their parameters. Groups are then distributed to one worker each. For all groups of results, stability measures are taken (presented in Subsection 5.2.4). Then, stability results are saved both in complete and summarized fashions.
8. **Run reliability evaluation**: Here, reliability is evaluated similarly to stability, as described in Subsection 5.2.5. Reliability results are also saved in complete and summarized forms.
9. **Aggregate times**: Finally, all the execution times for the algorithms are averaged, summarized, and saved.

4.3. Components

We have developed a few core components and helpers to make experimentation fast and easy. First, the project contains a helper tool to generate and download all necessary datasets used in this work. In addition, the project includes some base feature selection models. Moreover, some components are responsible for loading all the configuration

```
[
    {
        "description": "GA on CuMiDa data with
            perturbations",
        "datasets": [
            "Liver_GSE22405",
            "Prostate_GSE6919_U95C"
        ],
        "algorithms": [
            {
                "name": "SVMGeneticAlgorithm",
                "params": [[50], [100], [200]],
                "runs": 0,
                "sample_runs": 10
            }
        ]
    },
    {
        "description": "SVM vs SVM-RFE on XOR
            Dataset",
        "datasets": [
            "XOR"
        ],
        "algorithms": [
            {
                "name": "LinearSVM",
                "params": [[50]],
                "runs": 5,
                "sample_runs": 0
            },
            {
                "name": "SVMRFE",
                "params": [[50]],
                "runs": 5,
                "sample_runs": 0
            }
        ]
    }
]
```

Figure 3: Example snippet of a configuration file for task creation on the proposed framework.

and data needed for algorithms to perform feature selection. Besides, modules to achieve scoring, stability, reliability, and execution time are provided. Lastly, a pipeline structure to execute feature selection tasks is presented.

4.4. Base models

Concerning standardization and easier use of the feature selection models, some base models were created. The model from which all other models derive is the **BaseSelector**. The base models are described below:

1. **BaseSelector**: The base model class from which all others derive. It contains the base functionalities that feature selectors need to be run on this framework. To add new algorithms to the framework, this class must be extended. It is used to implement all algorithms listed in Subsection 3.7.
2. **BaseEmbeddedFeatureSelector**: A class that wraps models that can yield weight results. It fits and extracts the model's respective weights to create feature ranking. Class used in the implementation of *Decision Tree*, *Lasso*, *LinearSVM* and *Random Forest* embedded feature selection (Subsection 3.7.7 and Subsection 3.7.9).
3. **GeneticAlgorithmFeatureSelector**: A wrapper feature selector that accepts a fitness function and uses it to perform feature selection with a Genetic Algorithm. This model is used in the implementation of *SVM-GA* (Subsection 3.7.6).
4. **KBestFeatureSelector**: A model class used to perform ranking filter feature selection approaches. It accepts a weighting function, which is applied to all the features-target-class pairs. This class was used to implement *KW Filter* and *MI Filter* feature selectors (Subsection 3.7.1 and Subsection 3.7.2).
5. **FeatureSelectorPipeline**: A wrapper class that chains models into a single feature selection procedure. They are used to implement hybrid feature selection approaches, such as *ReliefF-GA* (Subsection 3.7.8).
6. **RFE**: Wrapper feature selection model that wraps a single model and performs Recursive Feature Elimination with it. It was used to implement the *SVM-RFE* feature selector (Subsection 3.7.5).

4.5. Datasets and Dataloader

To gather the data needed for experimentation, scripts are disposed to generate the synthetic data and download the datasets used in this work (presented in Subsection 5.1). The class **DataLoader** provides the functionalities to load from memory and preprocess the data. Also, a class named **SharedDatasets** was created to aid the datasets' utilization in a parallel environment. This class aims to encode all the datasets into a shared memory region and hold their representation so that they can be accessed by any worker running different processes.

4.6. Tasks and Presets

Every feature selection is encoded within a **Task** model. There is the **TaskLoader** component to facilitate task creation. It can load configuration from JSON files, where it is possible to specify parameters to generate Tasks. These files were called **Presets**. The parameters a preset may contain are a list of datasets to be used, the feature selectors that will be run for each dataset, their respective parametrization, sampling strategies to use, and how many runs will be executed, with and without sampling. The snippet in Figure 3 displays an example of the preset file.

4.7. Feature Selection Evaluators

Lastly, the project contains a module called *evaluation*. This module provides all the classes needed to take the measurements we will soon describe in Subsection 5.2. Three main classes are used to assess the evaluation:

- **ResultsScorer**: The class responsible for running prediction tasks on the resulting selections and retrieving their respective scores.

- **ResultsStability**: The component that applies stability and reliability metrics over all feature selection results.
- **ExecutionTimesAggregator**: The component that takes the feature selections execution times, splits them by every combination of feature selector and data and finally takes the average.

4.8. Technologies and Implementation

The framework application was implemented in the language Python 3.6. All predictors, prediction scorers, dataset generation tools, mutual-information, and nearest-neighbors calculations in this work were used, as is, from Scikit-learn[4] library (Pedregosa et al., 2011). In addition, the Scipy[5] and Numpy[6] libraries (Virtanen et al., 2020; Harris et al., 2020) were employed for general data manipulation, the development of some of the algorithms, and stability metrics.

Moreover, the Pandas[7] library (Wes McKinney, 2010) was used to read comma-separated values data to aggregate evaluation results and general results analysis throughout the development of this work. Finally, all tasks were run on 2x *Intel Xeon E5-2650V4*, totaling 48 virtual cores with a base frequency of 2.2Ghz, 128 GB of RAM, and Ubuntu 18.04.5 LTS operating system.

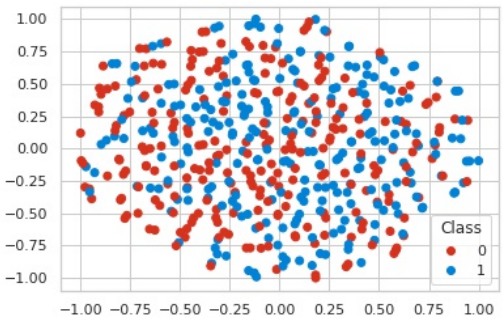
5. Methodology

Recapping what was discussed in Section 1, feature selection is one of the main techniques used to deal with the problems that come along with high dimensional data. This work aims to generate a comparative analysis of the main feature selection procedures in different domains and perspectives. This section presents the methodology and experiments conducted to achieve this goal. First, all the datasets upon which the algorithms performed feature selection are described. Then, our experimental setup is detailed, and all the measures taken are outlined.

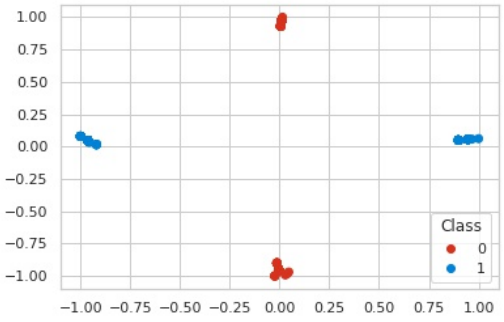
5.1. Datasets

Problems concerning a high number of features, a small number of samples, and noisy or redundant data are present in several domains. To address these situations, we selected datasets with unique properties that can be used to assess feature selection regarding these problems. We also synthesized some datasets to evaluate some specific characteristics of feature selectors, such as selection accuracy or the ability of selectors to deal with the correlation between features. All datasets presented here have two classes and have a balanced number of classes to shorten this work's scope. These constraints are arranged so that fewer variables are introduced in the analysis of the results.

In Subsection 5.1.1, the synthetic XOR dataset is described. Then, in Subsection 5.1.2, the creation of synthetic data to simulate redundancy is presented. Subsequently, in Subsection 5.1.3, we describe the CuMiDa database. Table 3 and 4 present more specific details about the data.



(a) All features



(b) Informative features

Figure 4: t-SNE of features in XOR dataset (a) before feature selection and (b) after feature selection. Each color corresponds to a class. As can be seen, the difference between the two figures can help the user decide if the selected features can represent the classes or not.

5.1.1. Synthetic XOR Binary Dataset

The XOR dataset is a binary composition of 500 samples and 50 features. Two of these features characterize the XOR logical operator[8] (Tan et al., 2009; Grisci et al., 2021) and, therefore, are informative to the target class. All the other 48 attributes are fully randomized binary values (noisy features). An example of an XOR dataset in which the attribute *class* is the result of the XOR operation between attributes *Informative 1* and *Informative 2*. Figure 4 is a 2-dimensional projection of all the features versus only the informative features. The main objective of utilizing such a dataset is to observe which evaluated algorithms can deal with a strong non-linear correlation between the two informative attributes.

[4] https://scikit-learn.org/
[5] https://www.scipy.org/
[6] https://numpy.org/
[7] https://pandas.pydata.org

[8] The XOR logical operator is a function that receives two binary values and returns 0/False if they are equal or 1/True if they are different.

Table 3
Datasets quantities

Dataset	Number of samples	Number of Features	Informative Features	Redundant Features	Number of Classes
XOR	500	50	2	0	2
Synth_A	100	5000	50	0	2
Synth_B	100	5000	50	50	2
Liver	48	22284	unknown	unknown	2
Prostate	115	12647	unknown	unknown	2

Table 4
Datasets' properties

Dataset	Source	Feature Type	Class Type	Missing Data	Balanced Classes
XOR	Synthetic	Binary	Binary	✗	✓
Synth_A	Synthetic	Numerical	Binary	✗	✓
Synth_B	Synthetic	Numerical	Binary	✗	✓
Liver	CuMiDa	Numerical	Categorical	✗	✓
Prostate	CuMiDa	Numerical	Categorical	✗	✗

5.1.2. Synthetic Data With Redundancy

To evaluate how some algorithms might deal with redundancy, we created two datasets utilizing the dataset creation tools from the Scikit-Learn library (Pedregosa et al., 2011). The tool first creates an n-dimensional hypercube, where n is equal to the number of informative features the dataset should have. Then, it creates Gaussian clusters[9] of data-points[10] for each vertex in the hyper-cube. Furthermore, an equal amount of clusters is assigned to each class. The informative features are the ones that are coordinates for the created points in the hypercube.

The first dataset, *Synth_A*, was created with 100 samples, binary target classes, and 5000 numerical features, from which 50 are informative (directly correlated to the target class). The second dataset, *Synth_B*, was also created with 100 samples, binary target classes, and 5000 numerical features, from which 50 are informative, but there are also 50 other redundant features. The redundant features are randomly generated linear combinations of the 50 informative ones. Both datasets are normalized between 0 and 1. Figure 5 shows a 2-dimensional projection of all features versus the projection of the 50 informative features in the *Synth_A* dataset. Figure 6 shows the same projection but for the *Synth_B* dataset.

5.1.3. CuMiDa

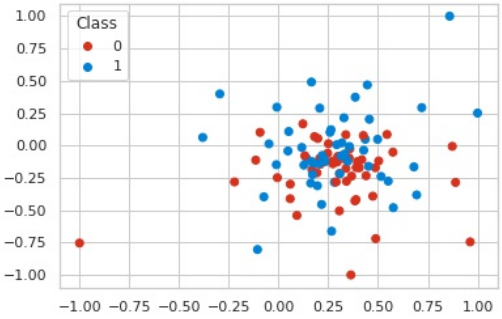
CuMiDa stands for Curated Microarray Database. It comprises extensively treated microarrays specifically curated for testing and benchmarking machine learning methods in cancer research, thus being ideally suited for our experiments (Feltes et al., 2019). All the data (78 datasets) was examined and handpicked from a sample of more than 30,000 databases. It was submitted to background correction, normalization, and filtered after some sample quality analysis (Feltes et al., 2019). Besides, CuMiDa also provides baseline accuracy from experimentation with some popular machine learning techniques. Because of their high dimensionality and relevance in medical applications, microarray datasets are often used in experiments in feature selection research (Grisci et al., 2018, 2019). However, between 2010 and 2020, 57% of the publications about feature selection that had experiments using gene expression data used at least one outdated dataset, 23% used only outdated data, and 32% did not adequately cited the source of the data (Grisci et al., 2023). The use of inadequate or outdated gene expression datasets is a common problem in feature selection publications, and adopting CuMiDa is a way to prevent this issue (Grisci et al., 2023).

In this work, we utilize two datasets from CuMiDa: *Liver-GSE22405* and *Prostate-GSE6919-U95C*. Both have numerical values for predictor features, binary categorical values for outcome classes, and both datasets are fully defined. There is no missing value in any variable. *Liver-GSE22405* dataset contains 48 samples with 22,284 features, while *Prostate-GSE6919-U95C* has 115 samples and 12,647 features, thus being good candidates to evaluate how algorithms deal with the *curse of dimensionality* and the *small sample problem*. Figure 7 presents the t-SNE 2-dimensional projection of all features from both datasets. For simplicity, we may refer to these datasets only by *Liver* and *Prostate* datasets.

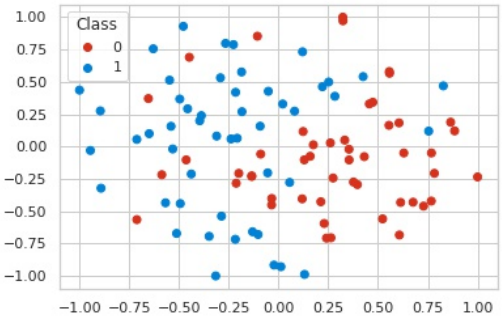
5.2. Experiments

Feature selection algorithms can be employed in virtually any machine learning task that relies on a dataset to perform classification or regression and can be used in a wide range of applications. However, every application has its own needs and criteria, e.g., medical applications

[9] Gaussian clusters are points of data sampled from a normal distribution, with mean value equal to the center of the cluster.
[10] Total number of data-points is equal to the number of samples.

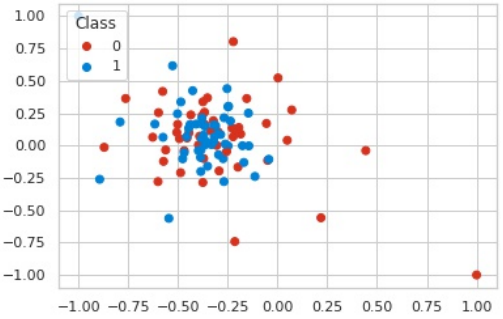


(a) All features

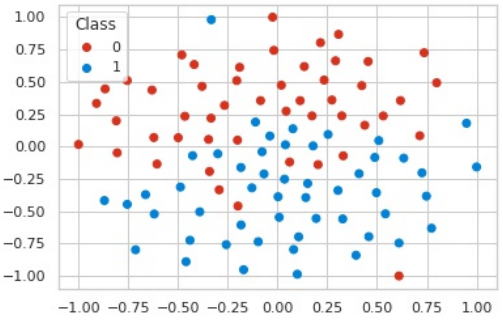


(b) Informative features

Figure 5: t-SNE of features in Synth_A dataset (a) before feature selection and (b) after feature selection. Details as in Figure 4.



(a) All features



(b) Informative features

Figure 6: t-SNE of features in Synth_B dataset (a) before feature selection and (b) after feature selection. Details as in Figure 4.

naturally prioritize high recall and specificity. Simultaneously, recommender systems usually hang to the side of fast execution times in the trade-off between time complexity and prediction scores, and, finally, biomedical applications seek high accuracy and stability.

We execute various setups to better address and understand where each feature selector best fits. Different datasets were employed, each with unique characteristics to assess distinct properties. Also, various sampling techniques to promote perturbations were applied to evaluate the robustness of feature selectors. Moreover, all the dataset combinations, sampling mode, feature selector, and input parameters are executed **31** times to generate statistically relevant results. All the executions of these combinations totaled **8,835** initial feature selection tasks. Table 5 displays the parameterization used for each executed algorithm. The value *auto* in the table means that the parameter's value was automatically defined by some heuristic. In all procedures executing Lasso, the regularization term λ was set to 0.001, which is still high enough for the model to penalize 99% of the features to weight zero in bigger datasets.

Each result of feature selection that is weighted or ranked is taken advantage of by the order's assumption to generate more efficient tests. Smaller subsets from the original selection are reevaluated whenever possible. This scheme cannot be used for algorithms in which the results are unordered subsets. In this case, several tasks with different parameters of the number of selected features were executed. Table 6 presents the size of the subsets evaluated for each dataset. This approach totaled **25,575** data points of resulting feature selections that were further submitted to necessary transformation and aggregation. The following subsections describe the evaluation criteria and measurements we apply to the obtained subsets of features.

5.2.1. Selection Accuracy

To evaluate how good a feature selector is at selecting relevant features, we analyze feature selection on two synthetic datasets: XOR and Synth_A (Subsection 5.1.1 and Subsection 5.1.2, respectively). The premise in evaluating these datasets is that one knows precisely which features are relevant. Then, for all the feature selection approaches

Table 5
Feature Selection Algorithms Parametrization per Dataset

Algorithms	Parameters	Datasets			
		XOR	Synth_A; Synth_B	Prostate	Liver
KW Filter; MI Filter	Selected Features	50	5000	12647	22284
mRMR	Selected Features	50	100	200	200
ReliefF	Selected Features	100	5000	12647	22284
	Number of Neighbours	100	100	100	100
Lasso	Selected Features	50	5000	12647	22284
	Alpha	0.001	0.001	0.001	0.001
Decision Tree	Selected Features	50	5000	12647	22284
	Criteria	Gini	Gini	Gini	Gini
Random Forest	Selected Features	50	5000	12647	22284
	Criteria	Gini	Gini	Gini	Gini
	Number of Estimators	100	100	100	100
SVM; SVM-RFE	Selected Features	50	5000	12647	22284
	Max Iterations	1000	1000	1000	1000
SVM-GA; ReliefF-GA	Selected Features	5; 10; 20	10; 20; 50; 100	10; 20; 50; 100; 200	10; 20; 50; 100; 200
	Population Size	auto	auto	auto	auto
	Number of Generations	100	100	100	100
	% Elite	0.05	0.05	0.05	0.05
	% Crossover	0.5	0.5	0.5	0.5
	% Mutation	0.001	0.001	0.001	0.001
	Max Fitness	1	1	1	1
	Fitness Function	5-fold SVM F-measure	5-fold SVM F-measure	5-fold SVM F-measure	5-fold SVM F-measure

Table 6
Evaluated feature subset sizes for each dataset.

Dataset	Evaluated subset sizes
XOR	5; 10; 20
Synth_A	5; 10; 20; 50; 100
Synth_B	5; 10; 20; 50; 100
Liver	5; 10; 20; 50; 100, 200
Prostate	5; 10; 20; 50; 100, 200

presented in this work, we calculate two measures: the percentage of informative features among the selected features and the percentage of the selected features that are, in fact, informative. The results for selection accuracy are later presented and discussed in Subsection 6.1.

Percentage of Informative Features Selected

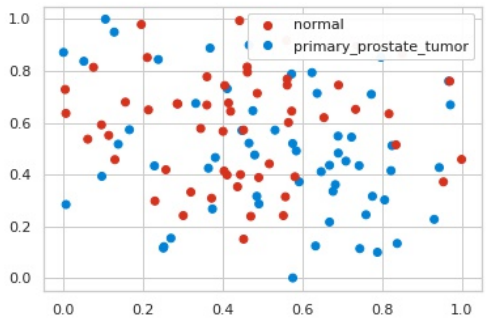
To better understand the proportion of the informative features being taken into account by the feature selectors, we introduce the *Percentage of Informative Features Selected* (PIFS) metric. The calculation for this metric is defined in Equation 24.

$$PIFS = \frac{|S_{selected} \cap S_{informative}|}{|S_{informative}|} \tag{24}$$

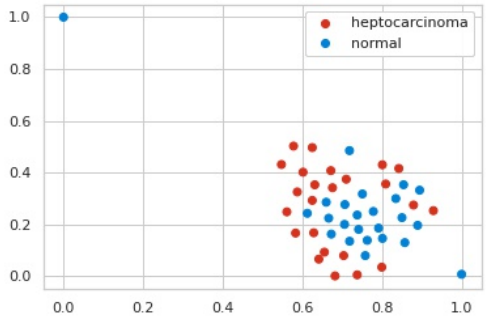
Where $S_{selected}$ is the set of selected features, and $S_{informative}$ is the set of informative features for a given dataset.

Percentage of Selected Features that are Informative

We also want to examine the amount of informativeness feature selectors obtain in the selected subsets. To



(a) Prostate-GSE6919-U95C



(b) Liver-GSE22405

Figure 7: t-SNE of features in CuMiDa datasets

quantify the informativeness within subsets, we introduce the *Percentage of Selected Features that are Informative* (PSFI) metric. This percentage is calculated by the formula in Equation 25.

$$PSFI = \frac{|S_{selected} \cap S_{informative}|}{|S_{selected}|} \tag{25}$$

Where $S_{selected}$ is the set of selected features, and $S_{informative}$ is the set of informative features for a given dataset.

5.2.2. Redundancy

One of the main goals of feature selection is to filter and reduce the volume of information retrieved from the original data while keeping only the vital information. As described in Subsection 3.7, some feature selection approaches try to assess this issue by minimizing redundancy within results. They usually try this by calculating the amount of redundant information between features or adding penalizing factors to their objective function. To evaluate how prominently feature selectors can deal with redundancy, we measure, for each selected subset, the amount of information every selected feature has to all the others (described in Subsection 3.6.3) and then take their average value. Results obtained for redundancy are detailed ahead in Subsection 6.2.

5.2.3. Prediction Scores

We employed several predictor models to evaluate a feature subset's prediction power, described in Subsection 3.3: Decision Trees, Random Forest, Naïve Bayes, and SVM. For each result and predictor, we execute 5-fold stratified cross-validation and retrieve their respective *accuracy* and *F-measure* as defined in Subsection 3.4. The results are further described in Subsection 6.3.

5.2.4. Stability

As stated in Subsection 3.8, high stability is an essential property for feature selectors to have. To evaluate stability, we utilize two approaches to promote perturbation at every execution:

1. *Bootstraps*: the data is re-sampled with reposition (Efron, 1992). On average, this strategy keeps 63.2% of the original samples, repeating some of the samples and, thus, adding a **significant amount of perturbation**.
2. *Sample Subsets*: 90% of the instances are sampled with reposition (Awada et al., 2012). This approach generates data with **small amounts of perturbation**.

After executing the algorithms with the sampled data, the stability metrics described in Subsection 3.8.5 are applied. Data is then grouped by their sampling type, algorithm, dataset, and the number of selected features. For each group's pair-wise combinations of results, the stability metrics described in Subsection 3.8.5 are taken and averaged, as shown in Equation 26. Because some of the datasets have an enormous amount of features and the majority of feature selectors would not rank nor weight more than 1% of the existing attributes, we normalize *Spearman's ρ* and *Pearson correlation* metrics to consider only the features that were ranked or weighted in the calculation. This normalization was done to obtain more relevant and comparable results, later presented in Subsection 6.4.

$$Avg_{metric}(S) = \frac{2}{|S| * (|S| - 1)} \sum_{i=1}^{|S|-1} \sum_{j=i+1}^{|S|} metric(S_i, S_j) \tag{26}$$

Where S is a subset of feature selection results.

5.2.5. Reliability

Boutsidis and Magdon-Ismail (2013) say that being deterministic is desirable because stochastic approaches can yield different results for every execution, thus decreasing expert confidence. To assess this, we also apply the same strategy presented in Subsection 5.2.4 to evaluate the algorithms' reliability concerning their results. However, the

metrics are applied to results not submitted to any perturbation. Results obtained are later presented at the end of Subsection 6.4.

5.2.6. Execution Time

Different algorithms have different time complexities and take different amounts of time to perform feature selection tasks. In this work, we executed every algorithm in a single-threaded process fashion and retrieved their averaged execution times. The results for the execution times are further displayed in Subsection 6.5.

6. Results and Discussion

This section presents the results of the experiments that were conducted. The accuracy of the algorithms in selecting relevant features is shown. Afterward, the results regarding redundancy among the selected feature subsets are presented. Subsequently, the results obtained by submitting feature selections to predictors are reported. Then, an analysis of the stability and reliability of the feature selection results is shown. Finally, the results of the feature selectors' execution time are described.

6.1. Selection Accuracy

This section describes the results of selection accuracy evaluation for both *Synth_A* and *XOR* datasets. The outcomes presented here are the average of **31** executions for each parameter setting combination with datasets without perturbation. Full results are available in Supplementary Material.

6.1.1. Results for XOR dataset

In Figure 8, the results obtained for the *XOR* dataset are shown. The percentage of informative features that the algorithms selected to specific quantities of selected features is presented in Figure 8a, and the rate of those selected features that are, in fact, informative is displayed in Figure 8b.

It is possible to see that both *ReliefF* and *Random Forest* selectors achieve perfect scoring, as they always selected the two informative features from the *XOR* dataset in our experiments. In contrast, filtering with *Kruskal Wallis* leads to the worst results. It never selected relevant features. These results are probably because both *ReliefF* and *Random Forest* are very sensitive to feature interactions[11], while *Kruskal Wallis* is univariate, and thus, it does not deal with correlation between features. *ReliefF-GA* and *SVM-GA* feature selectors also retrieved good average results. Due to their very stochastic nature, they could, most of the time, select 1 or 2 of the relevant features for *XOR*. Also, *SVM-RFE* and *Lasso* performed almost identically. They did not select any relevant features in subsets of sizes 5 and 10 and, on average, selected only one of the relevant features for subsets of size 20.

[11]The *Random Forest* model has high sensitivity when the correlation between features is non-linear, which is the case in the XOR dataset. Still, it might not yield satisfactory results on a perfectly linear setup.

6.1.2. Results for Synth_A dataset

Similarly to results for the *XOR* dataset, Figure 9a shows the percentage of informative features and Figure 9b, on the right, shows the percentage of which of those features are informative. However, contrary to the *XOR* results, the *Kruskal Wallis Filter* algorithm shows one of the best overall selection accuracies for the *Synth_A* dataset, being just behind *ReliefF* in terms of scores. Notably, for this dataset, *Mutual information Filter*, *mRMR*, *SVM-RFE*, *Decision Tree*, and *Lasso* had no feature selections with informative features for subsets of sizes 5 and 10.

Lastly, the overall selection accuracy implies that all tested algorithms, except for *ReliefF* and *Kruskal Wallis Filter*, on average, could select only from 0 to 2 relevant features. Therefore, *Synth_A* dataset must impose a hard challenge on feature selection tasks.

6.2. Selection Redundancy

This subsection presents the results obtained by evaluating how feature selectors deal with redundancy. All results were generated from feature selections of the *Synth_B* dataset (Subsection 5.1.2) without any form of perturbation. Table 7 shows the redundancy scores obtained by averaging the redundancy between each result's selected features for 31 executions of every algorithm. The overall redundancy does not appear to vary much between algorithms. However, it is possible to see that *mRMR* is the one that minimizes redundancy the most. This was expected because *mRMR* incorporates the redundancy in its minimization function.

6.3. Prediction Score

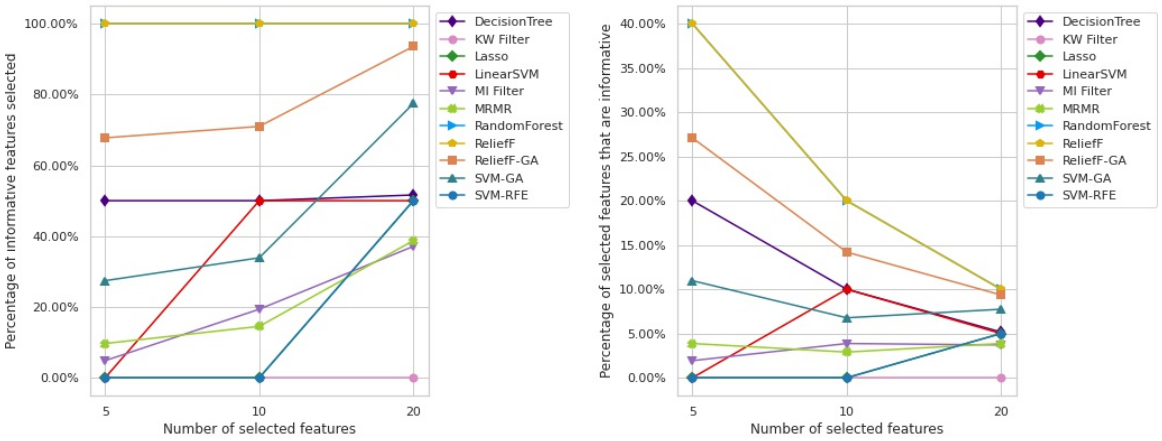
The average prediction scores obtained are presented from 31 executions for each dataset, feature selector, and subset size. It is shown that for every dataset, the prediction results by utilizing the Decision Tree, Naïve Bayes, SVM, and Random Forest predictors. As for the metrics, F-measure and accuracy are taken. However, accuracy is omitted for the sake of visibility. Because datasets are very balanced, the accuracy metric yields results similar to the F-measure.

6.3.1. Results for XOR dataset

Figure 10 shows the averaged F-measure for the four predictors used. In general, both *ReliefF* and *Random Forest* feature Selectors have the best results, achieving perfect scores when evaluated by the *Decision Trees* predictor with five features, and for *SVM* and *Random Forest*, they achieve perfect scores at subset sizes 5 and 10.

The Naïve Bayes classifier (Figure 10b) is not an attractive candidate to evaluate the XOR dataset because it only accounts for the probability of each feature to explain the data independently. Thus, it does not consider feature correlation. Besides, Naïve Bayes always tries to use all features, including the noisy ones, to explain the result being classified, therefore degrading results in this case.

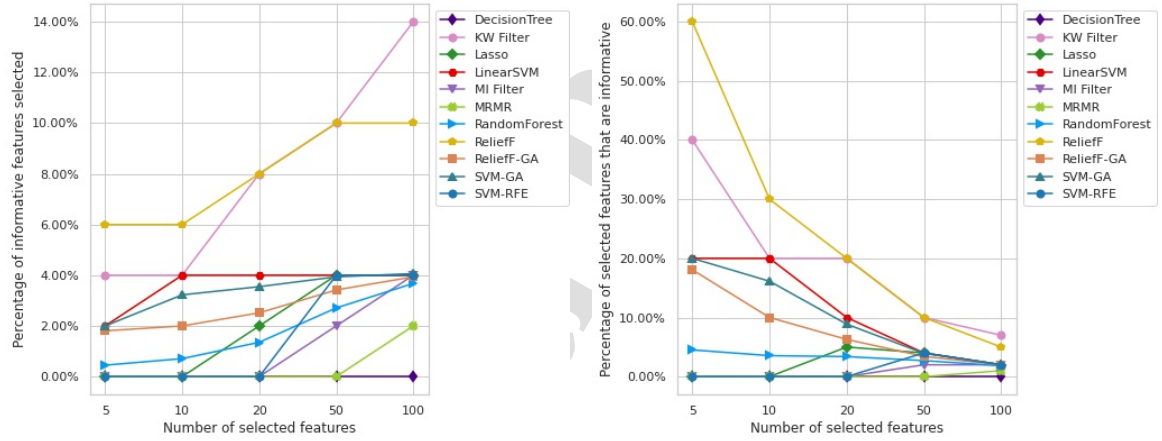
As stated in Subsection 6.1.1, both *ReliefF-GA* and *SVM-GA* feature selectors are often able to retrieve relevant features, thus having an above the average prediction score.



(a) Percentage of informative features that were selected.

(b) Percentage of selected features that are informative

Figure 8: Results of selection accuracy for XOR (average of 31 executions). Higher scores are better.



(a) Percentage of informative features that were selected

(b) Percentage of selected features that are informative

Figure 9: Results of selection accuracy for Synth_A (average of 31 executions). Higher scores are better.

Finally, none of the remaining feature selectors can handle the XOR dataset well, achieving accuracy values near 0.5.

6.3.2. Results for Synth_A dataset

As shown before, the *Synth_A* dataset provides a difficult challenge for feature selectors in selecting relevant features. Figure 11 shows the averaged F-measure for the four predictors used. When accounting for 20 or more selected features, *SVM-RFE* seems to have the best prediction scores for *Naïve Bayes*, *SVM*, and *Random Forest* predictors (Figure 11b, 11c, and 11d, respectively), while the *Decision Tree* feature selector provides the worst results. *KW Filter*, *Linear SVM*, *RelieF*, *SVM-GA*, and *Lasso* achieve decent overall results with these three predictors. While the first three seem to improve in prediction score in terms of the number of selected features, the latter two appear to reach their best results when 50 features are selected.

However, when looking at the results utilizing *Decision Tree* as the predictor (Figure 11b), the best scoring algorithm is *Decision Tree* as the feature selector. Besides, *SVM-RFE* results for *SVM* predictor yields some of the best results. These results show that both *Decision Tree* embedded and *SVM* wrapper feature selections might have over-fitted for this dataset.

Table 7
Average redundancy scores for Synth_B dataset (with standard deviation). The best scores obtained for each subset size by feature selector are in displayed in bold.

Feature Selector	Number of Selected Features				
	5	10	20	50	100
Lasso	0.0521 $\pm 2.45 \times 10^{-5}$	0.0313 $\pm 2.59 \times 10^{-5}$	0.0249 $\pm 8.56 \times 10^{-6}$	0.0255 $\pm 1.74 \times 10^{-6}$	0.0258 $\pm 9.27 \times 10^{-7}$
ReliefF	0.0468 ± 0.00	0.0356 ± 0.00	0.0367 $\pm 8.44 \times 10^{-6}$	0.0299 $\pm 2.48 \times 10^{-6}$	0.0276 $\pm 9.74 \times 10^{-7}$
KW Filter	0.0330 $\pm 3.58 \times 10^{-5}$	0.0313 $\pm 2.13 \times 10^{-5}$	0.0296 $\pm 6.84 \times 10^{-6}$	0.0286 $\pm 2.61 \times 10^{-6}$	0.0267 $\pm 1.11 \times 10^{-6}$
LinearSVM	0.0322 $\pm 1.83 \times 10^{-5}$	0.0326 $\pm 1.40 \times 10^{-5}$	0.0258 $\pm 5.85 \times 10^{-6}$	0.0265 $\pm 3.89 \times 10^{-6}$	0.0267 $\pm 1.96 \times 10^{-6}$
Random Forest	0.0295 $\pm 1.45 \times 10^{-2}$	0.0276 $\pm 6.80 \times 10^{-3}$	0.0260 $\pm 2.13 \times 10^{-3}$	0.0260 $\pm 1.17 \times 10^{-3}$	0.0259 $\pm 3.91 \times 10^{-4}$
SVM-GA	0.0290 $\pm 1.36 \times 10^{-2}$	0.0263 $\pm 5.86 \times 10^{-3}$	0.0265 $\pm 2.85 \times 10^{-3}$	0.0260 $\pm 1.04 \times 10^{-3}$	0.0257 $\pm 5.35 \times 10^{-4}$
MI Filter	0.0250 $\pm 3.47 \times 10^{-5}$	0.0267 $\pm 1.51 \times 10^{-3}$	0.0257 $\pm 9.90 \times 10^{-6}$	0.0273 $\pm 2.76 \times 10^{-6}$	0.0268 $\pm 1.50 \times 10^{-6}$
ReliefF-GA	0.0218 $\pm 1.27 \times 10^{-2}$	0.0250 $\pm 5.19 \times 10^{-3}$	0.0261 $\pm 3.21 \times 10^{-3}$	0.0261 $\pm 1.21 \times 10^{-3}$	0.0261 $\pm 6.75 \times 10^{-4}$
Decision Tree	0.0159 $\pm 1.65 \times 10^{-5}$	0.0260 $\pm 4.10 \times 10^{-3}$	0.0260 $\pm 1.40 \times 10^{-3}$	0.0251 $\pm 8.38 \times 10^{-4}$	**0.0250** $\pm \mathbf{1.98 \times 10^{-4}}$
SVM-RFE	0.0090 $\pm 1.76 \times 10^{-5}$	0.0238 $\pm 1.01 \times 10^{-5}$	0.0202 $\pm 5.40 \times 10^{-6}$	0.0247 $\pm 2.74 \times 10^{-6}$	0.0257 $\pm 1.36 \times 10^{-6}$
mRMR	**0.0052** $\pm \mathbf{3.51 \times 10^{-5}}$	**0.0173** $\pm \mathbf{1.07 \times 10^{-3}}$	**0.0198** $\pm \mathbf{7.98 \times 10^{-6}}$	**0.0237** $\pm \mathbf{2.40 \times 10^{-4}}$	0.0257 $\pm 1.05 \times 10^{-4}$

6.3.3. Results for Synth_B dataset

When looking at Figure 12, it is possible to see that the *Synth_B* dataset's feature selections yield very similar results to Synth_A's (Figure 11), except for the *SVM-GA* feature selector. For all predictors, the prediction power of the *SVM-GA* decreases as the number of selected features increases. It might be a consequence of the nature of our *SVM-GA* approach. It defines population size as the number of selected features, and the number of individuals is a function of population size and total number of attributes. The higher the number of selected features, the smaller the number of individuals in the population.

6.3.4. Results for Liver dataset

In Figure 13, the results obtained for the *Liver* dataset are presented. The overall results show that this dataset poses a more straightforward feature selection challenge for the evaluated selectors because all their respective selections can achieve average high prediction scores. The only noticeable result is that of the *Decision Tree* feature selector. Just as in Subsection 6.3.2, the *Decision Tree* yields the worst results when evaluated by our predictors, except when a *Decision Tree* predictor evaluates its selection. The conclusion here is that the *Decision Tree* over-fits as well for the *Liver* dataset. It is also worth noting that only *SVM-GA*, when selecting five features and being evaluated by the SVM predictor, consistently achieved an F-measure of 1.0.

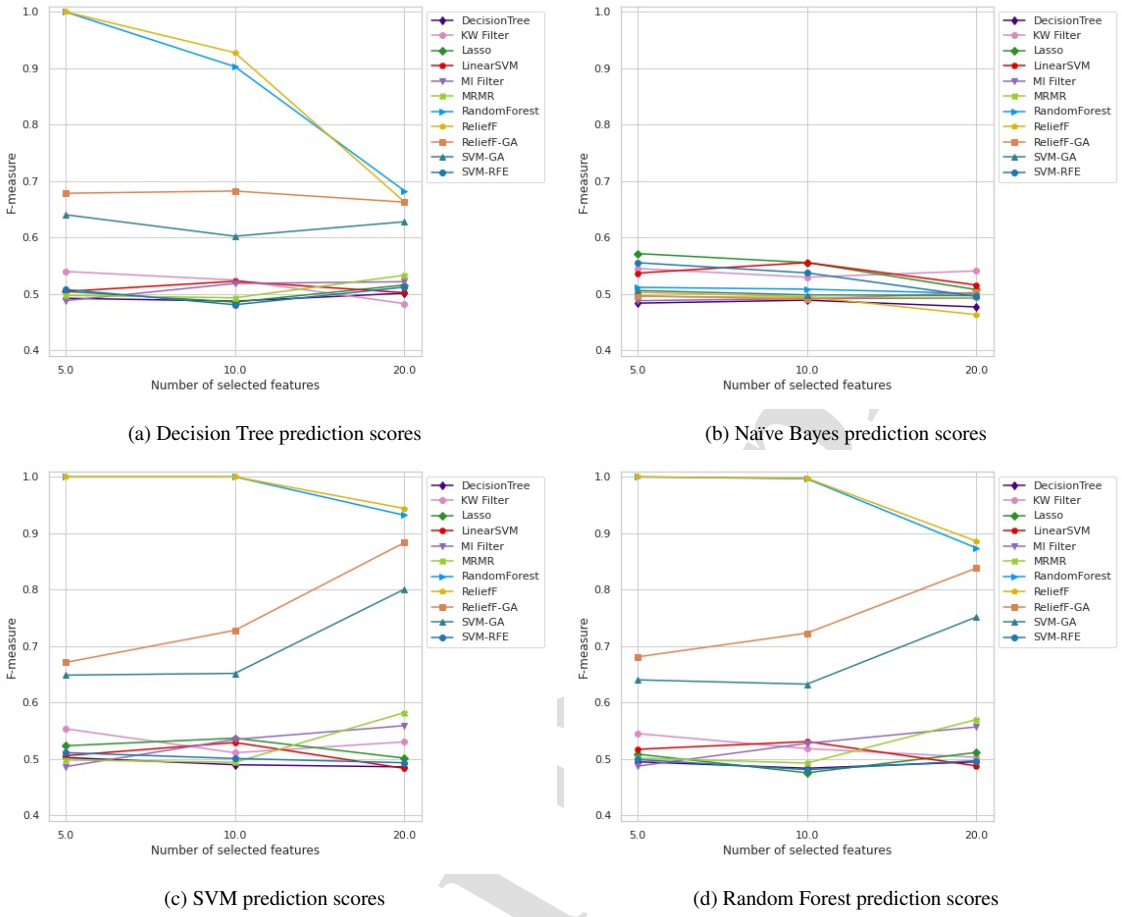
6.3.5. Results for Prostate dataset

Finally, the results for the *Prostate* dataset are shown. This dataset is the one that presents the lowest prediction scores for the feature selection approaches in this work. No predictor achieved a perfect classification score. Thus, the *Prostate* dataset provides the most challenging classification task among the five datasets.

Looking at Figure 14, just like for other datasets, the *Decision Tree* feature selector over-fits for prediction with a *Decision Tree* model. The same thing happens to *SVM-RFE* and embedded SVM selection. Both yield good results, but their best results are retrieved when evaluated by an SVM predictor. Also, *SVM-GA* gives reasonably good outcomes when the number of features selected is 5 or 10. However, its F-measure decreases the higher the number of selected features is.

6.4. Stability and Reliability

This subsection presents the results for *Stability* and *Reliability* analysis on the feature selectors. They were obtained by executing every algorithm for each subset size and dataset (without perturbation, small perturbation, and considerable perturbation) 31 times. Then, every stability metric was taken for these 31 executions. This work evaluated every stability metric presented in Subsection 3.8.5. However, since *Jaccard*, *Hamming*, *Percentage of Overlapping Features*, *Ochiai*, and *Dice* all evaluate subset results and retrieve values with very similar meaning and outcomes to *Kuncheva*



(a) Decision Tree prediction scores

(b) Naïve Bayes prediction scores

(c) SVM prediction scores

(d) Random Forest prediction scores

Figure 10: Prediction Scores for XOR (average of 31 executions)

Index[12], we are only going to present the latter in the results. For ranked results, *Canberra Distance* and *Spearman ρ* metrics results are presented. Lastly, the outcomes of Pearson's correlation coefficient are shown for weighted results. The subsequent subsections present a comparative analysis of the results from different perspectives on data and measures.

6.4.1. Kuncheva Index vs Spearman's ρ

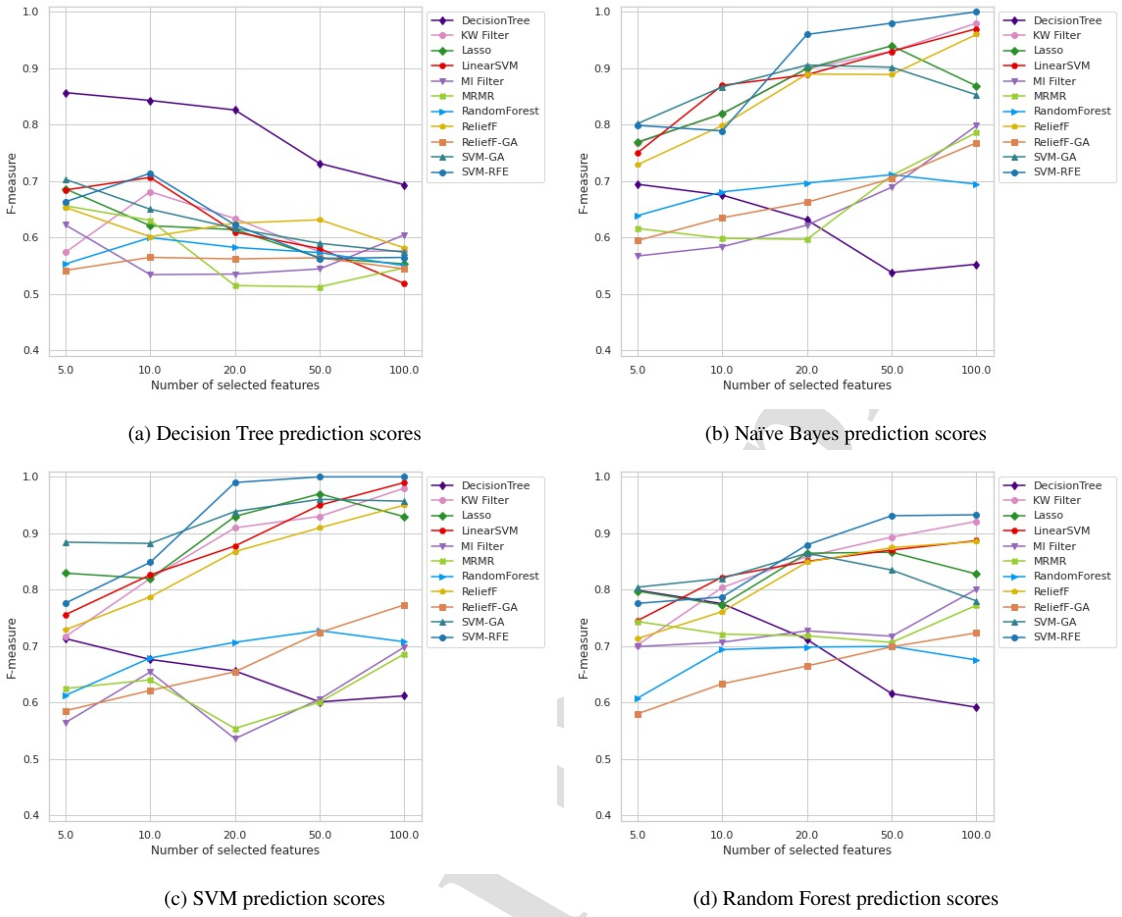
During results analysis, it was noticed that both *Kuncheva Index* and *Spearman's ρ* measure yielded very similar results to all kinds of sampling and datasets when evaluating subset results, even though one addresses stability to subsets and the other to rank correlation, respectively. To gauge the difference between them, we employed **MAE** for the normalized results of every experiment that resulted in subsets. Moreover, the mean absolute error is minimal: $\mathbf{4.849 \times 10^{-3}}$.

[12]Kuncheva Index is also a wise choice because it satisfies the property of correction for chance, can be addressed to all kinds of results presented here and, besides that, we only look to subsets of defined size, so the fact that it is not fully defined is not a problem.

Table 8 shows the error between some metrics evaluated. Figure 15 depicts stability results for both metrics in different datasets and sampling strategies. No significant difference can be perceived. Further on, due to this high resemblance between both metrics, we will only show *Kuncheva Index* results because it covers more algorithms than *Spearman's ρ*.

6.4.2. Stability on Significant amounts of Perturbation

Figure 16 shows the XOR dataset results under considerable amounts of perturbation (Subsection 5.2.4). It is interesting to notice that different metrics yield different notions regarding magnitude for this case. Nevertheless, for the Kuncheva Index, the difference in magnitude from the most stable to the least stable estimator is approximately 30% to 42% of the metric space. While looking at the Pearson correlation, this difference is even more significant, accounting for about 53% to 79% of the metric space. Also, even though Canberra cannot be directly compared to the



(a) Decision Tree prediction scores

(b) Naïve Bayes prediction scores

(c) SVM prediction scores

(d) Random Forest prediction scores

Figure 11: Prediction Scores for Synth_A dataset (average of 31 executions)

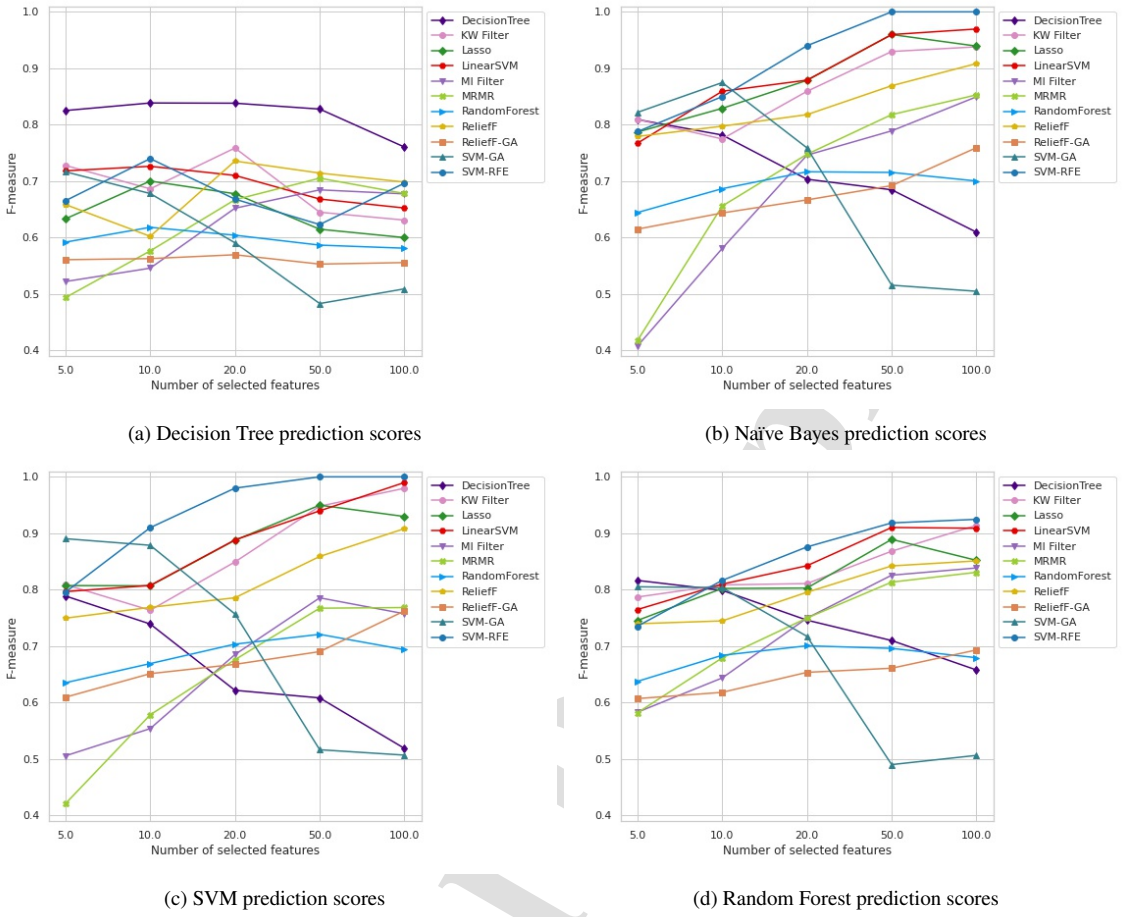
Table 8
Comparison of MAE between some of the employed stability metrics results. All metrics are normalized, and distances are complemented ($D_{comp} = 1 - D$) before the error calculation. The values in bold are the smallest errors obtained between the two estimators.

Metric	Jaccard	Kuncheva	Spearman	Canberra	Pearson
Jaccard	-	0.0641	0.0746	0.4501	0.0959
Kuncheva	0.0641	-	$\mathbf{4.849 \times 10^{-3}}$	0.4012	0.1337
Spearman	0.0746	$\mathbf{4.849 \times 10^{-3}}$	-	0.3968	0.1336
Canberra	0.4501	0.4012	0.3968	-	0.4478
Pearson	0.0959	0.1337	0.1336	0.4478	-

other two metrics, it is possible to visualize that it has tiny gaps in the distance between rankings.

Assessed by the Kuncheva Index, some algorithms seem to have positive stability growth regarding the number of features, while others decrease. Stability was expected to decrease because the XOR dataset only has two informative features, and, like so, selecting more variables should lead to more variance as they are random noise. Lastly, Pearson Correlation seems to be the most discriminative metric for this dataset. Both *ReliefF* and *Random Forest* achieve high positive correlation values. Yet again, both algorithms come across as the best option in dealing with the XOR dataset.

All metrics indicate that the *Decision Tree* embedded feature selection is one of the most stable approaches for bigger datasets under significant amounts of perturbation. *Decision Tree* stability is followed right after by *mRMR* and *Lasso*, which present good results in subsets and rank results (Figure 17a and 17b, respectively). However, when looking



(a) Decision Tree prediction scores

(b) Naïve Bayes prediction scores

(c) SVM prediction scores

(d) Random Forest prediction scores

Figure 12: Prediction Scores for Synth_B dataset (average of 31 executions)

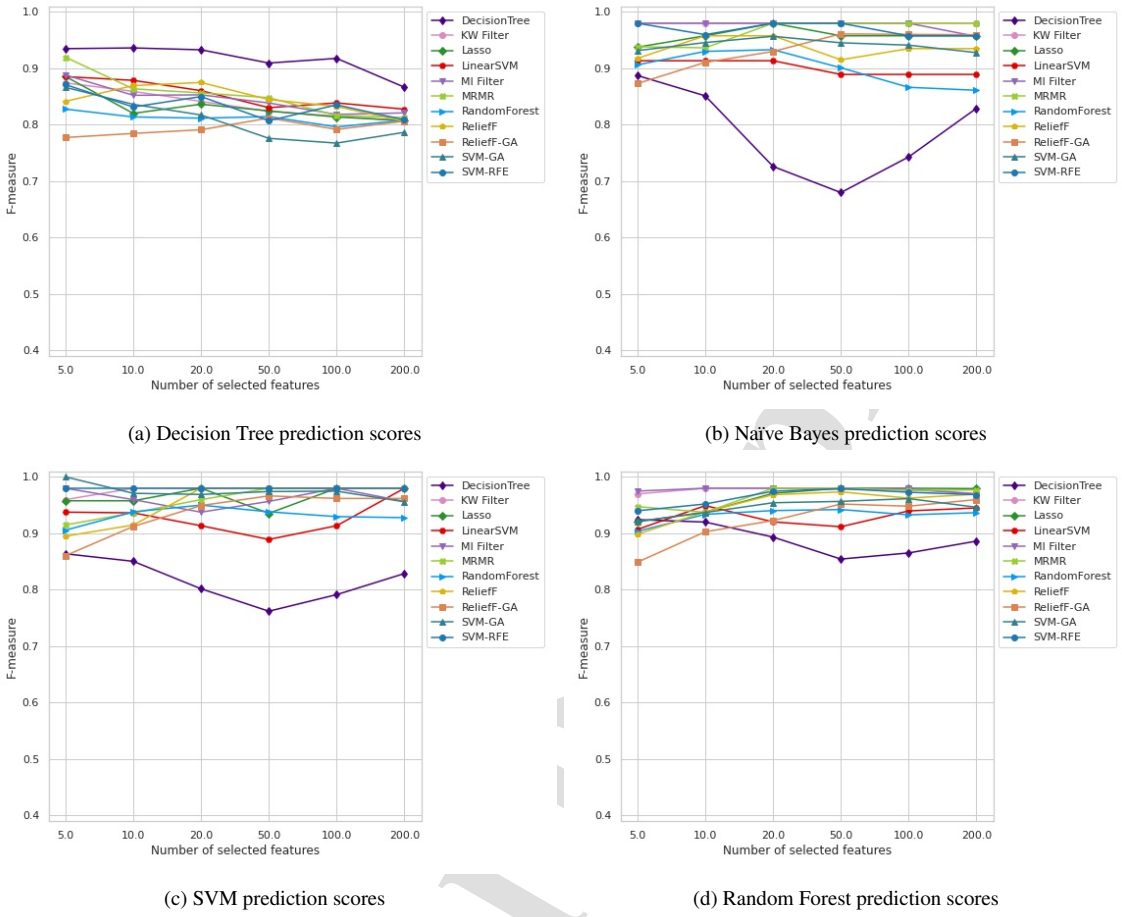
at Figure 17c, the *Pearson correlation coefficient* (PCC) for these algorithms states that the attributed weights are, in general, negatively linearly correlated, and just about as much as *KW Filter* and *Random Forest*. The overall results for the PCC metric indicate that the weights attributed to selections are inversely correlated. It must be due to the high variance in the input data. Finally, the results obtained for *Synth_A*, *Synth_B*, and *Liver* datasets are very similar to the results for the *Prostate* dataset. The most significant difference is that for the *Liver* dataset, *ReliefF* and embedded *SVM* feature selectors appear to be more stable than *mRMR*.

6.4.3. Stability on Small amounts of Perturbation

Another criterion for evaluating stability is how feature selectors respond to small amounts of perturbation. Like the results for significant perturbation, stability evaluation on larger datasets tends to yield similar results. Therefore, we will only show the results for *Synth_A* and *XOR* datasets.

For the *XOR* dataset, in Figure 18c, *Kuncheva Index* has denoted *ReliefF* as the most stable algorithm when the number of selected features is 5, but its stability falls when considering 10 or 20. In this context, the *Canberra Distance* metric yields results comparable to *Kuncheva Index*'s in terms of ordering. Furthermore, in the scenario of low perturbation, *Pearson correlation* shows results resembling when applied to *XOR* results with high variance in input data. Both *ReliefF* and *Random Forest* are also the most stable approaches. Lastly, *mRMR* and *Mutual Information Filter* are the less stable approaches.

Figure 19 shows the results of Kuncheva Index, Canberra Distance, and Pearson Correlation applied to feature selections from *Synth_A*. For *Kuncheva Index*, it is possible to notice that for a higher number of selected features, *Decision Tree* feature selection has the highest stability among the algorithms; even so, for a small number of selected features, it becomes one of the most unstable approaches. Right after *Decision Tree* are *Kruskal Wallis* filter, *ReliefF*, and *LinearSVM*, having returned fair overall results. And lastly, *Random Forest*, *ReliefF-GA*, and *SVM-GA* retrieve the worst stability indexes, being, sometimes, very close to the worst



(a) Decision Tree prediction scores

(b) Naïve Bayes prediction scores

(c) SVM prediction scores

(d) Random Forest prediction scores

Figure 13: Prediction Scores for Liver dataset (average of 31 executions)

possible value. Canberra distance seems to agree with the Kuncheva Index in most cases.

Last but not least, results change slightly when looking at the *Pearson Correlation*. When considering weights, *Decision Tree* scores drop significantly while *ReliefF* and *LinearSVM* feature selectors remain in similar positions. Consequently, both *ReliefF* and *LinearSVM* seem to be good choices for dealing with high-dimensional data in terms of stability.

6.4.4. Reliability

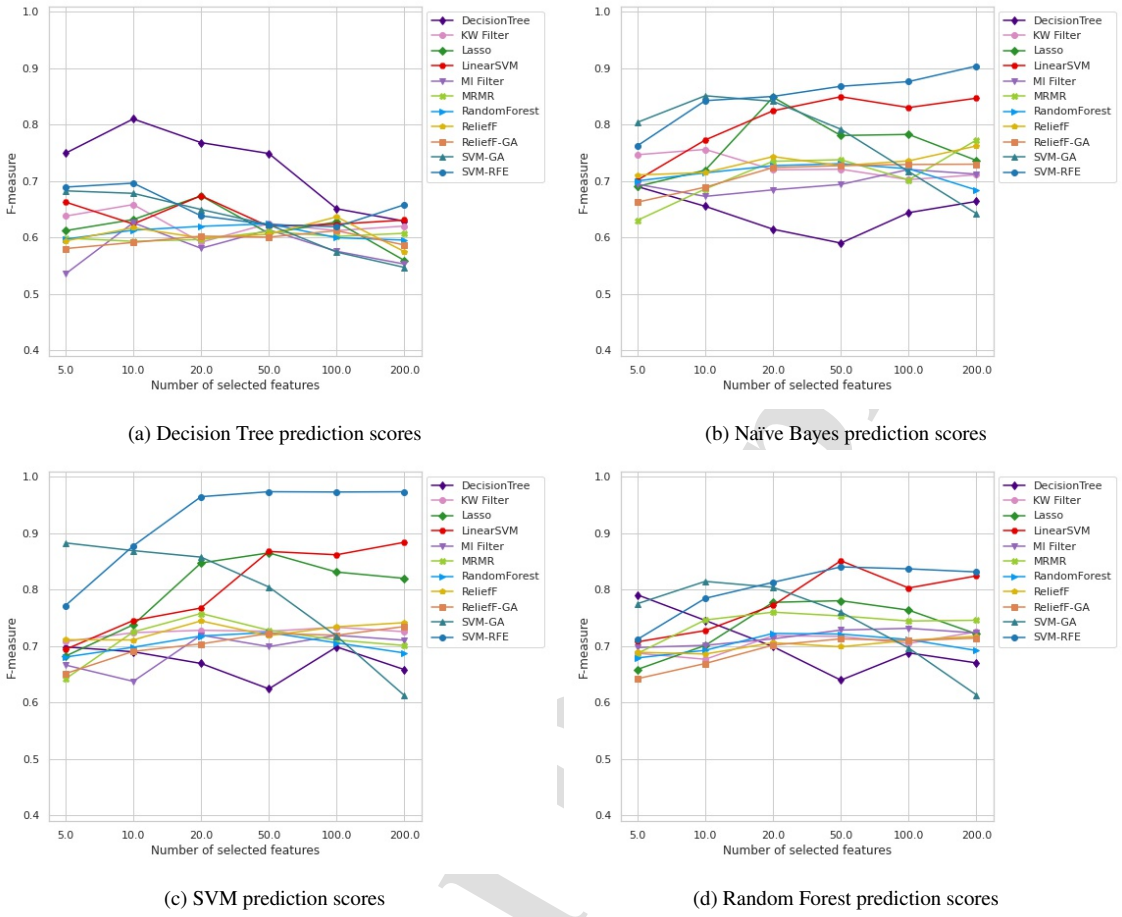
We now present the results obtained when evaluating the reliability of results on datasets without perturbation. As some of the feature selectors are deterministic or have convex objective function[13], they achieve perfect reliability. This group of feature selectors includes *Kruskal Wallis Filter*, *ReliefF*, *SVM-RFE*, *Lasso*, and embedded *SVM* selectors.

Figure 20 compares some results acquired through the Kuncheva Index and Pearson's correlation. For the larger datasets, *Synth_A* and *Liver*, the mutual information approaches, MI Filter and mRMR, have nearly perfect reliability. However, when looking at the *XOR* dataset, they are scored as the most unreliable approaches. As for the *Genetic Algorithm* based approaches, *SVM-GA* and *ReliefF-GA*, their very stochastic nature has a direct impact on their respective reliability, consistently achieving low scores. Finally, it is remarkable that the *Decision tree* yields better results than *Random Forest* overall (except for the XOR dataset on Pearson Correlation). As *Random Forest* is an ensemble approach, it was expected to have results with less variance.

6.5. Execution time

Table 9 presents the results obtained by the averaged execution time of all our algorithms for every dataset. All

[13] Having a convex objective function implies that, despite the estimator being started with random weights, it will eventually converge to the optimal solution if enough iterations are executed during the optimization process.



(a) Decision Tree prediction scores

(b) Naïve Bayes prediction scores

(c) SVM prediction scores

(d) Random Forest prediction scores

Figure 14: Prediction Scores for Prostate dataset (average of 31 executions)

results are measured considering the highest number of selected features for each dataset: 20 features for the *XOR* dataset, 100 features for *Synth_A* and *Synth_B* datasets, and 200 features for *Liver* and *Prostate* datasets.

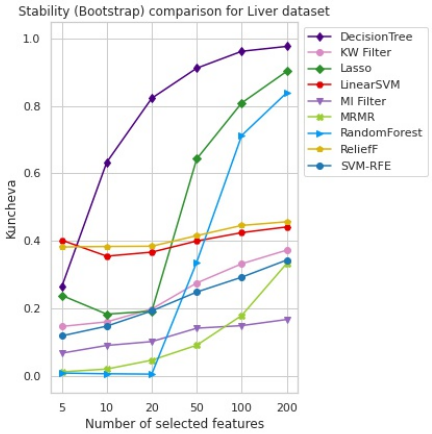
It is possible to note that even though *mRMR* is a filter method, it is the slowest of the tested approaches for more extensive datasets because it performs space searches. In contrast, the ensemble approach using *Random Forests* is the fastest-performing selector for high-dimensional data.

Figure 21 is a visual comparison of the execution times (logarithmically scaled) for all algorithms. It is noticeable that applying a hybrid approach with *ReliefF* filter before applying the *Genetic Algorithm* wrapper lowers processing time significantly. The difference in time is 2 to 10 times lower against the times of a simple *Genetic Algorithm* wrapper, as shown in Figure 22.

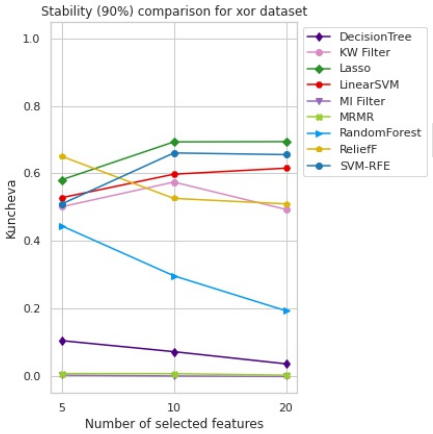
Finally, it is possible to notice that execution times of *Random Forest* and *ReliefF* are higher for the *XOR* dataset. Even though the data is way smaller in the number of features, the *XOR* dataset has more samples. Thus, it causes a higher execution time for these algorithms than their mean execution for other datasets.

6.6. Discussion

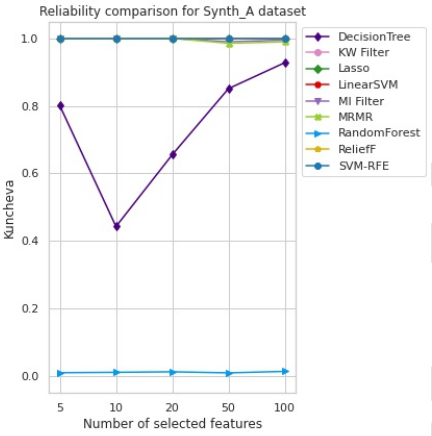
This section made it possible to bring up insightful information on where the algorithms better succeed or fail and compare which one is adequate for the addressed tasks. It is noticeable that *ReliefF* is fast and has yielded promising results in selection accuracy, prediction scores, stability, and reliability. It shows to be suited in a broad range of aspects, including when strong feature correlation is present. It is a compelling universal choice when dealing with small sample sizes, considering it is a filter approach with good generalization power. Its major drawback is that it weights all features without considering any redundancy between them. *Random Forest* is also a fast option that yields similar results to *ReliefF* in the presence of strong non-linear correlation, but it did only present good overall results for the *XOR* dataset. Nevertheless, it seems to be faster on datasets with fewer samples. That is not the *XOR* dataset's case.



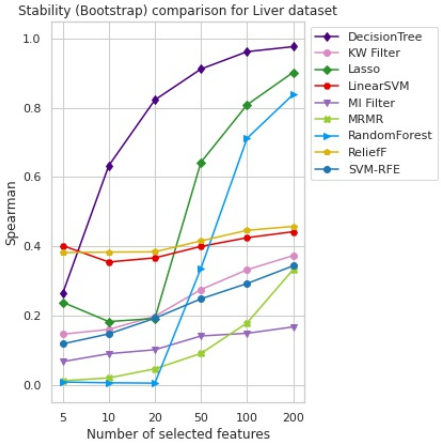
(a) Kuncheva on **Liver** dataset (significant amounts of perturbation).



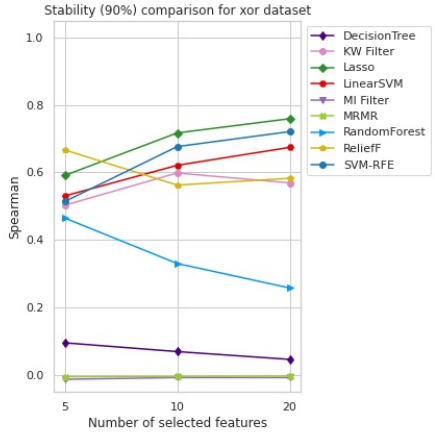
(b) Kuncheva on **XOR** dataset (small amounts of perturbation).



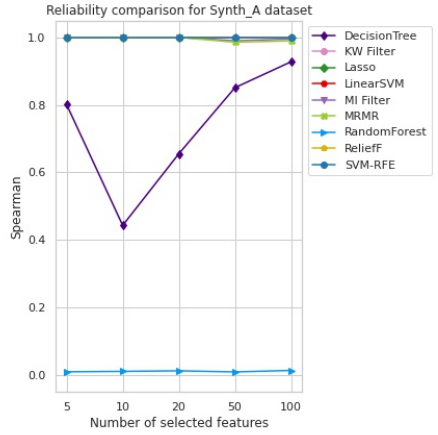
(c) Kuncheva on **Synth_A** dataset (no perturbation).



(d) Spearman ρ on **Liver** dataset (significant amounts of perturbation).

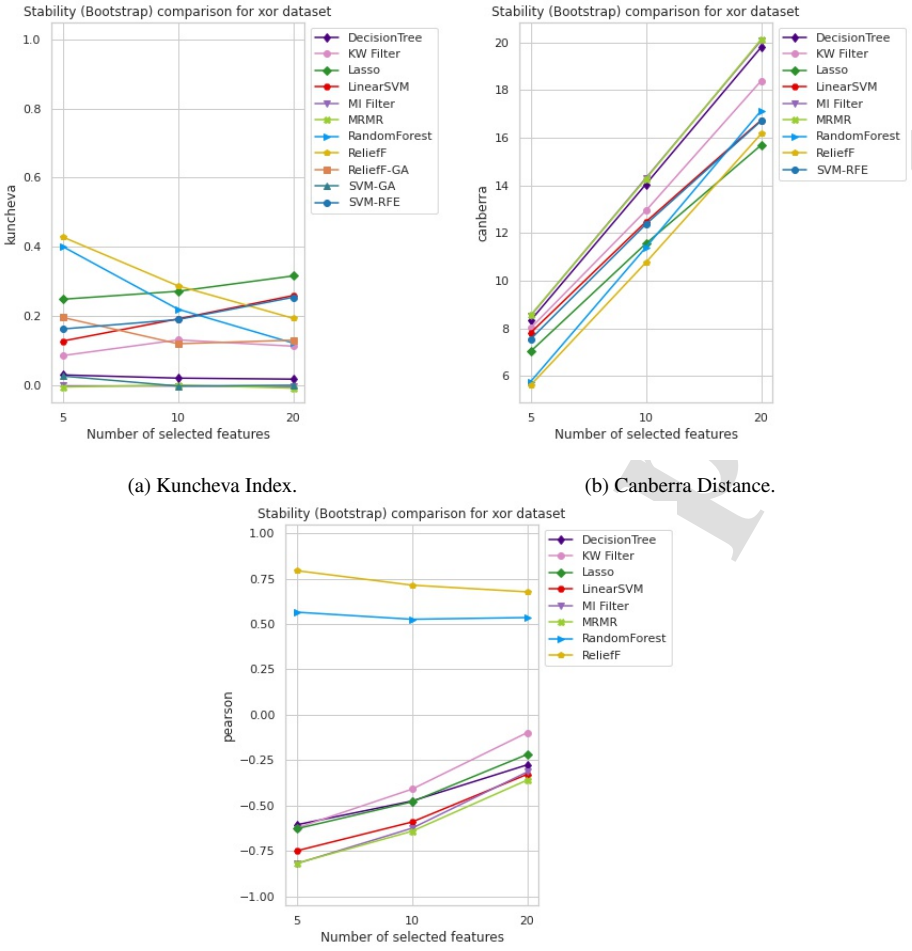


(e) Spearman ρ on **XOR** dataset (small amounts of perturbation).



(f) Spearman ρ on **Synth_A** dataset (no perturbation).

Figure 15: Visual comparison between Kuncheva Index and Spearman ρ metrics on feature selectors that both metrics can evaluate results.



(a) Kuncheva Index.

(b) Canberra Distance.

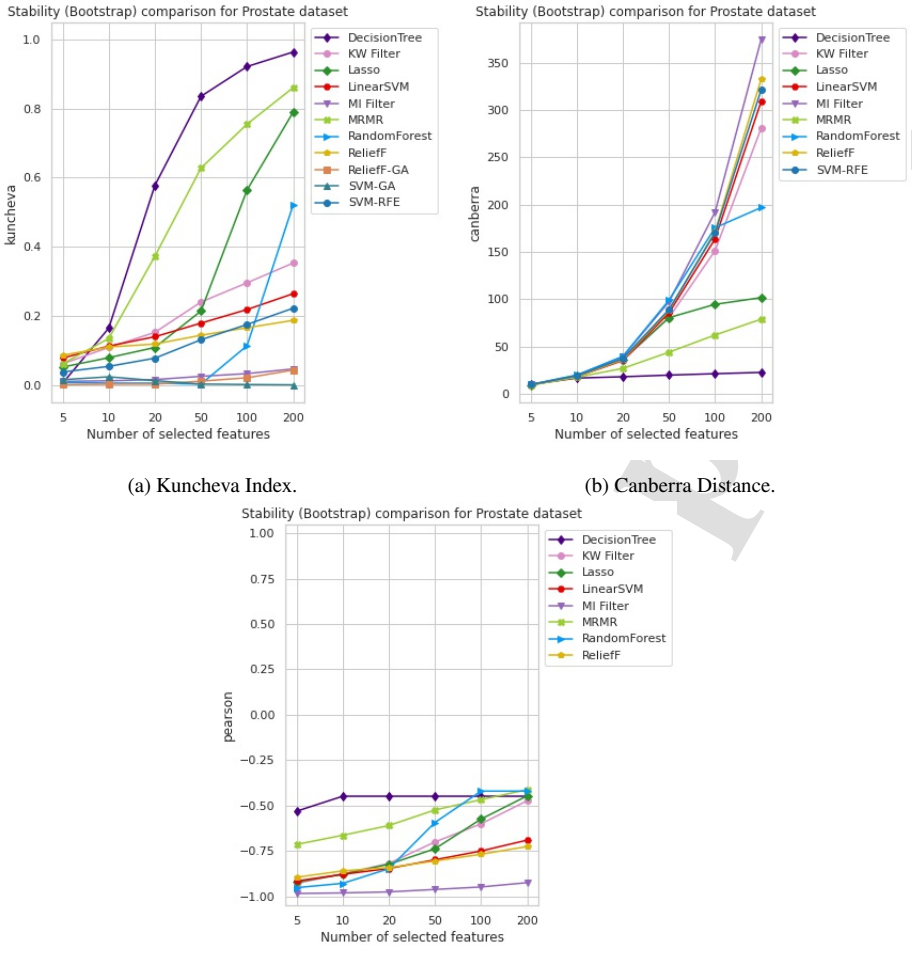
(c) Pearson correlation.

Figure 16: Metrics comparison for each feature selector algorithm over significant amounts of perturbation on XOR dataset (from 31 executions).

Regarding prediction power, it was also possible to perceive that *SVM-RFE* has yielded the best prediction scores for high dimensional datasets. Still, it falls considerably when its selections are used in prediction with the *XOR* dataset. Also, embedded *SVM* and *Lasso* selections yield satisfactory results, not as great, but with less computational time. Thus, they are both good candidates to be used on extensive data such as *CuMiDa* datasets. The *SVM-GA* was also an interesting approach to test. Its execution time is correlated with its parameterization, but it might be shorter depending on the stopping criteria. Even though it was generically parameterized, it yielded outstanding feature selections for prediction regarding small-size subsets. Besides, even though it is a wrapper approach, it did not seem to overfit as much as *SVM-RFE* and the other embedded procedures (Subsection 3.7.7).

Regarding data containing redundancy, as expected, *mRMR* obtained the best results. However, it did lack in terms of all the other measurements we considered. *SVM-RFE*, in addition to its prediction power, also seems to deal very decently with redundancy, with outcomes that approximate *mRMR*'s.

As an embedded feature selector, *Decision Tree* presents good stability results over prostate and liver datasets. However, as we utilized fixed subset sizes, non-weighted features might have been repeatedly selected between executions, and thus *Decision Tree* stability result may have been biased. Further experimentation is needed to find out. Besides, both *MI filter* and *mRMR*, even though simple filtering approaches, took longer to execute than expected, with *mRMR* being even slower than wrapper approaches. Further investigation is needed on whether mutual information is a slow approach



(a) Kuncheva Index.

(b) Canberra Distance.

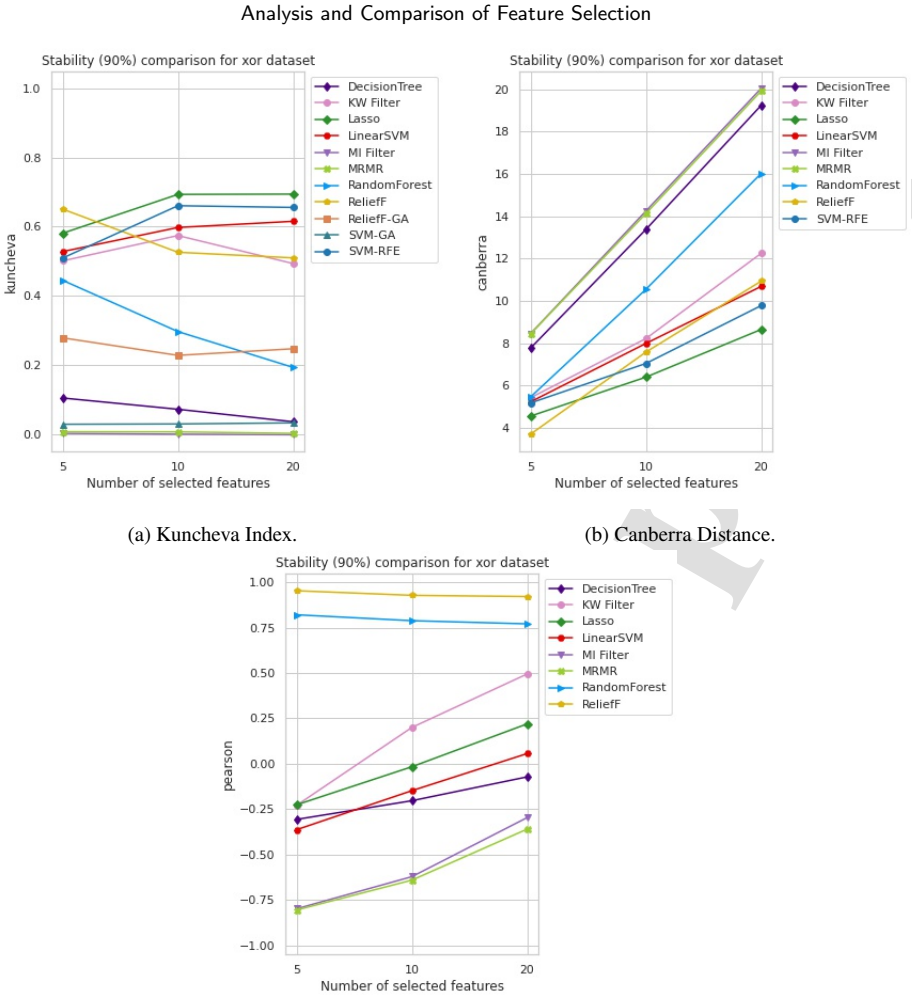
(c) Pearson correlation.

Figure 17: Metrics comparison for each feature selector algorithm over significant amounts of perturbation on the Prostate dataset (from 31 executions).

when dealing with numerical values or if the implementation (Pedregosa et al., 2011) is sub-optimal. Finally, the results were produced from feature selectors, which might not have had parameters perfectly optimized, and thus, the results might not be the optimal outcome. In most cases, the utilized parameters were the values that respective authors and libraries advised (Pedregosa et al., 2011).

There are still many aspects and situations that our framework cannot deal with and need further development. Among the topics that can be improved is that the application, in its current state, can not deal with incomplete data. Also, it does not yet support regression problems. Aside from that, all the presented algorithms are implemented to run considering a single-threaded process without parallelism. However, a significant portion of the feature selectors can take advantage of threading and parallelization. For example, *ReliefF* should be able to calculate the weights for every instance in parallel, whereas *Kruskal Wallis Filter*, *Mutual Information Filter*, and *mRMR* should be able to parallelize the calculation of statistical correlation between variables. *Genetic Algorithms* could distribute their crossover, mutation, and fitness evaluation operations. Lastly, any ensemble approach should be able to parallelize the training and prediction of its estimator components. For example, the *Random Forest* ensemble could parallelize the creation of its decision trees and its prediction process. On top of that, as they perform many vector operations, several of these algorithms could also benefit from GPUs.

Furthermore, different kinds of analysis can be done on the obtained data. For example, one could analyze the relevance and consistency of a single feature on the results among different feature selection methodologies. Beyond that, one could bring these results into the perspective of domain experts to assess their quality. Besides, it should



(a) Kuncheva Index.

(b) Canberra Distance.

(c) Pearson correlation.

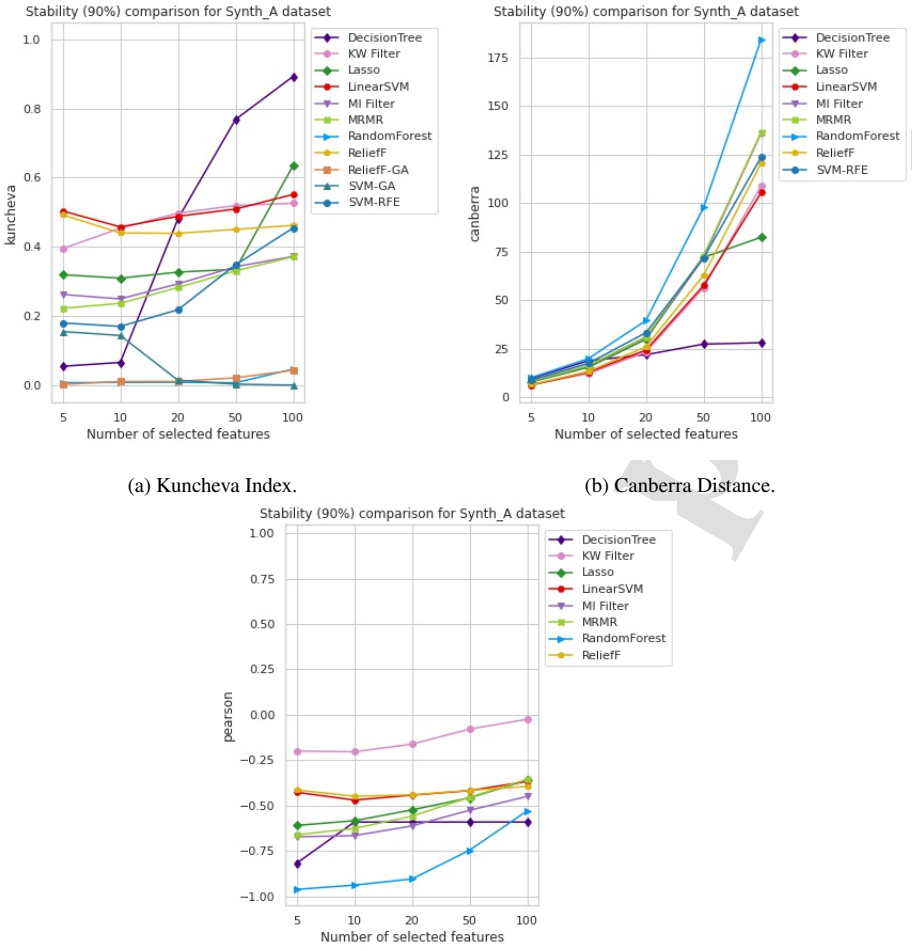
Figure 18: Metrics comparison for each feature selector algorithm over small amounts of perturbation on XOR dataset (from 31 executions).

be possible to analyze the correlation between the measurements discussed in this work so that newer and more general feature selection quality estimators can be investigated.

7. Conclusion

This work presented the fundamentals of feature selection and the problems that it tries to solve. We introduced some of the most widespread approaches and feature selection methods and delved into their properties and workings. Next, we reviewed some metrics that measure prediction accuracy and stability. After that, several ways in which feature selections can be evaluated were discussed. Subsequently, we presented how we set up and conducted the experiments. Lastly, we showed and discussed the results we obtained. Many interesting insights about all the algorithms, results, and even evaluation metrics could be brought up. It was possible to see that every algorithm works very differently in many regards, and results show that one algorithm might be more useful in some specific domain or situation than others. Also, the usefulness of an algorithm can be evaluated by different metrics, which, even if accounting for the same objective, might present different results. For example, we have shown that two metrics of stability, applied to the same data, can disagree on the most stable result.

As mentioned in the introduction of this work, one of the primary motivations behind research in feature selection is dealing with the ever-increasing amount of data in almost every field that concerns areas such as machine learning, statistics, business, and medicine. The volume of gathered data does not appear to stop growing so soon, and as long as it keeps increasing, so should research in feature selection. Thus, it is of great utility that we further extend the proposed framework, not only to keep up with the state-of-the-art



(a) Kuncheva Index.

(b) Canberra Distance.

(c) Pearson correlation.

Figure 19: Metrics comparison for each feature selector algorithm over small amounts of perturbation on the Prostate dataset (from 31 executions).

approaches but also to attend to more problems and domains. The results presented here give a general, but also very informative, view of how the presented algorithms perform in different aspects and may pose a general baseline for comparison in the field of feature selection. Moreover, we not only supply this baseline. We also created a framework that supports generating all the results shown throughout this work, making experimentation fully reproducible. Also, this framework can be, to a greater extent, used to generate more databases and extended to new algorithms and metrics for feature selection benchmarks.

Finally, the resulting data in this work account for a large amount of generated and collected multi-dimensional information, and even though we tried to present some of the most relevant details and insights, it was not possible to cover all and every outcome of the executed procedures in a single report. On account of that, the complete and already summarized data is available as supplemental material, and the framework used was made publicly available.

It is essential to acknowledge the limitations of this study. One notable limitation is the reliance on binary classification datasets, which may not fully capture the complexities of real-world scenarios. The decision to only use binary classification datasets was made to keep the scope of the study and comparisons feasible, as adding more datasets and algorithms would make the number of research variables too high. Future research should endeavor to incorporate multiclass datasets to provide a more nuanced evaluation of the performance of feature selection methods. Additionally, while our framework facilitates the comparison and evaluation of feature selection algorithms, it may be subject to certain constraints and biases inherent in the datasets

(a) Kuncheva Index for XOR dataset.

(b) Kuncheva Index for Synth_A dataset.

(c) Kuncheva Index for Liver dataset.

(d) Pearson correlation for XOR dataset.

(e) Pearson correlation for Synth_A dataset.

(f) Pearson correlation for Liver dataset.

Figure 20: Comparison between the reliability of feature selectors using Kuncheva Index (above) and Pearson's correlation coefficient (below) for XOR, Synth_B, and Liver datasets (31 feature selections).

Table 9
Algorithms mean execution times per dataset in seconds when selecting 20 features for XOR dataset, 100 features for Synth_A and Synth_B datasets, and 200 features for Liver and Prostate datasets (average of 93 executions for every dataset and feature selector pair). The highlighted values are both the fastest and the slowest execution times.

Algorithm Type	Algorithm	Dataset				
		XOR	Synth_A	Synth_B	Liver	Prostate
Filter	KW Filter	0.077	5.918	5.744	25.677	15.994
	MI Filter	0.494	21.056	23.133	85.502	61.492
	ReliefF	0.572	0.207	0.245	0.327	0.564
	mRMR	16.238	**1990.809**	**2047.387**	**14952.389**	**11208.552**
Wrapper	SVM-RFE	2.818	426.917	437.269	2300.282	3047.588
	GeneticAlgorithm	**38.367**	1844.029	1842.428	479.918	4804.901
Embedded	DecisionTree	**0.006**	0.284	0.262	0.273	0.823
	Lasso	0.026	0.662	0.633	2.388	2.155
	LinearSVM	0.140	0.148	0.144	0.138	0.357
Hybrid	ReliefF-GA	22.003	361.554	361.038	42.063	362.393
Ensemble	RandomForest	0.583	**0.079**	**0.073**	**0.068**	**0.115**

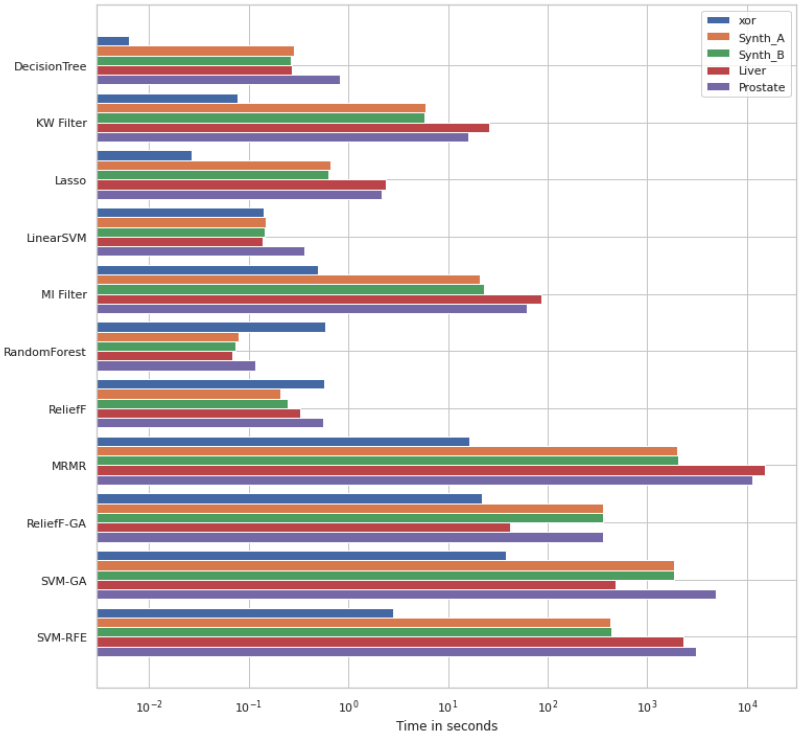


Figure 21: Algorithms execution time comparison on selecting 20 features for XOR dataset, 100 features for Synth_A and Synth_B datasets, and 200 features for Liver and Prostate datasets (average of 93 executions for each dataset and algorithm).

and evaluation metrics employed. Researchers should exercise caution when interpreting the results and consider the context-specific factors influencing algorithm performance.

Moving forward, several avenues for future research warrant exploration. Firstly, there is a need to investigate the potential of deep feature selection methods (Section 2) in improving the performance of traditional techniques. Leveraging deep learning architectures to extract informative features from high-dimensional datasets could enhance predictive accuracy and robustness. Furthermore, it is crucial to

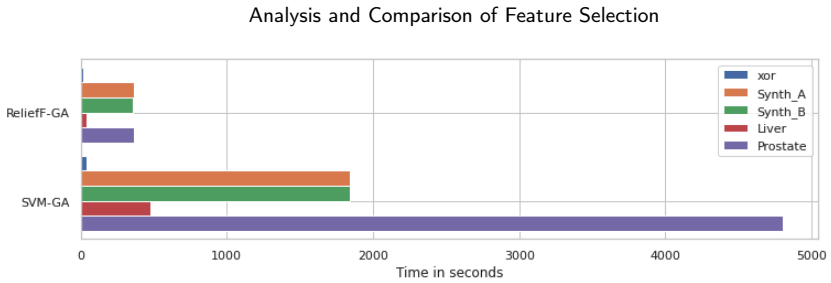


Figure 22: GA and ReliefF-GA execution time comparison on selecting 20 features for XOR dataset, 100 features for Synth_A and Synth_B datasets, and 200 features for Liver and Prostate datasets (average of 93 executions for each dataset and algorithm).

extend the scope of this research to encompass multiclass datasets. While our analysis primarily focused on binary classification tasks, future studies should evaluate the effectiveness of feature selection methods in handling multiclass scenarios. This extension would provide a more comprehensive understanding of algorithm performance across diverse problem domains.

As described in Subsection 5.2, in our experimental setup, we considered the parameterization of each feature selection algorithm to ensure fair and comprehensive evaluation. The parameters were chosen based on established best practices, empirical evidence from previous studies, and domain knowledge, following the standard recommendations from the original publications or the implementation guidelines from software libraries. However, we acknowledge the importance of analyzing the impact of parameter choices on algorithm performance and determining if an optimal configuration for these parameters exists.

To address this concern, our framework easily allows the setup of different values (Subsection 4.6) to conduct further analysis to investigate the sensitivity of our results to variations in parameter values. In future work, it is possible to explore the effects of parameter tuning on the performance metrics of feature selection algorithms across different datasets and scenarios, as the ideal set of parameter values is expected to change according to the data. This analysis will systematically vary the parameters within a predefined range and evaluate their impact on selection accuracy, stability, and computational efficiency. Additionally, it is possible to employ techniques such as grid search or cross-validation to search for optimal parameter configurations systematically. However, given the large number of comparisons already present in this study, such analysis is considered out of scope and would be better presented as a separate research.

In summary, this study has made significant contributions to the field of feature selection by offering comprehensive insights into algorithm performance, introducing a reproducible experimentation framework, and identifying avenues for future research. By addressing the limitations and extending the scope of our research, we can further advance the understanding and application of feature selection methods in managing high-dimensional datasets across diverse domains. However, we highlight that the framework, datasets, and methodology proposed in this paper are freely available and suited for further experiments, making it possible to readily adapt the experiments we present to new feature selection algorithms and types of data, including multiclass datasets, and the quick comparison between new and old feature selection models.

Data and code availability

The necessary source code, hyperparameters, and datasets used in the experiments and results can be accessed in GitHub: https://github.com/sbcblab/GenExpFS. The microarray data used in this work is available in the CuMiDa repository: http://sbcb.inf.ufrgs.br/cumida.

CRediT authorship contribution statement

Matheus Cezimbra Barbieri: Conceptualization, Methodology, Software, Validation, Formal analysis, Data Curation, Writing - Original Draft, Visualization. **Bruno Iochins Grisci:** Conceptualization, Methodology, Validation, Writing - Review Editing, Supervision. **Márcio Dorn:** Conceptualization, Resources, Writing - Review Editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by grants from the Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) [19/2551-0001906-8], Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) [314082 / 2021-2; 440279/2022-4; 408154/2022-5], the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) [STICAMSUD 88881.522073 / 2020-01; DAAD/-CAPES PROBRAL 88881.198766/2018-01] - Brazil, and the Emerging Leaders in the Americas Program Scholarship with the support of the Government of Canada. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil

(CAPES) - Finance Code 001. This study was funded by INCT Ciências Forenses with funding from the National Council for Scientific and Technological Development (CNPq).

References

Ang, J.C., Mirzal, A., Haron, H., Hamed, H.N.A., 2015. Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. IEEE/ACM transactions on computational biology and bioinformatics 13, 971–989.

Avila, E., Kahmann, A., Alho, C., Dorn, M., 2020. Hemogram data as a tool for decision-making in covid-19 management: applications to resource scarcity scenarios. PeerJ 8, e9482.

Awada, W., Khoshgoftaar, T.M., Dittman, D., Wald, R., Napolitano, A., 2012. A review of the stability of feature selection techniques for bioinformatics data, in: 2012 IEEE 13th International Conference on Information Reuse & Integration (IRI), IEEE. pp. 356–363.

Bolón-Canedo, V., Sánchez-Marono, N., Alonso-Betanzos, A., Benítez, J.M., Herrera, F., 2014. A review of microarray datasets and applied feature selection methods. Information Sciences 282, 111–135.

Boutsidis, C., Magdon-Ismail, M., 2013. Deterministic feature selection for k-means clustering. IEEE Transactions on Information Theory 59, 6099–6110.

Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

Cilia, N.D., De Stefano, C., Fontanella, F., Raimondo, S., Scotto di Freca, A., 2019. An experimental comparison of feature-selection and classification methods for microarray datasets. Information 10, 109.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Machine learning 20, 273–297.

Diaz-Gomez, P.A., Hougen, D.F., 2007. Initial population for genetic algorithms: A metric approach., in: Gem, pp. 43–49.

Dorn, M., Grisci, B.I., Narloch, P.H., Feltes, B.C., Avila, E., Kahmann, A., Alho, C.S., 2021. Comparison of machine learning techniques to handle imbalanced covid-19 cbc datasets. PeerJ Computer Science 7, e670.

Efron, B., 1992. Bootstrap methods: another look at the jackknife, in: Breakthroughs in statistics. Springer, pp. 569–593.

Feltes, B.C., Chandelier, E.B., Grisci, B.I., Dorn, M., 2019. Cumida: An extensively curated microarray database for benchmarking and testing of machine learning approaches in cancer research. Journal of Computational Biology 26, 376–386.

Feltes, B.C., Poloni, J.D.F., Dorn, M., 2021. Benchmarking and Testing Machine Learning Approaches with BARRA:CuRDa, a Curated RNA-Seq Database for Cancer Research. Journal of Computational Biology 28, 931–944. doi:10.1089/cmb.2020.0463.

Formica, V., et al., 2020. Complete blood count might help to identify subjects with high probability of testing positive to sars-cov-2. Clinical Medicine 20, e114–9.

Frohlich, H., Chapelle, O., Scholkopf, B., 2003. Feature selection for support vector machines by means of genetic algorithm, in: Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence, IEEE. pp. 142–148.

Gao, W., Hao, P., Wu, Y., Zhang, P., 2023. A unified low-order information-theoretic feature selection framework for multi-label learning. Pattern Recognition 134, 109111.

Gao, W., Li, Y., Hu, L., 2021. Multilabel feature selection with constrained latent structure shared term. IEEE Transactions on Neural Networks and Learning Systems .

Gill, M., Anderson, R., Hu, H., Bennamoun, M., Petereit, J., Valliyodan, B., Nguyen, H.T., Batley, J., Bayer, P.E., Edwards, D., 2022. Machine learning models outperform deep learning models, provide interpretation and facilitate feature selection for soybean trait prediction. BMC plant biology 22, 1–8.

Grisci, B., Kuhn, G., Colombelli, F., Matter, V., Lima, L., Heinen, K., Pegoraro, M., Borges, M., Rigo, S., Barbosa, J., et al., 2022. Perspectives on risk prioritization of data center vulnerabilities using rank aggregation and multi-objective optimization. arXiv preprint arXiv:2202.07466 .

Grisci, B.I., Feltes, B.C., Dorn, M., 2018. Microarray classification and gene selection with fs-neat, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, Rio de Janeiro, Brazil. pp. 1–8.

Grisci, B.I., Feltes, B.C., Dorn, M., 2019. Neuroevolution as a tool for microarray gene expression pattern identification in cancer research. Journal of biomedical informatics 89, 122–133.

Grisci, B.I., Feltes, B.C., de Faria Poloni, J., Narloch, P.H., Dorn, M., 2023. The use of gene expression datasets in feature selection research: 20 years of inherent bias? Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery , e1523.

Grisci, B.I., Krause, M.J., Dorn, M., 2021. Relevance aggregation for neural networks interpretability and knowledge discovery on tabular data. Information Sciences 559, 111–129.

Guyon, I., Weston, J., Barnhill, S., Vapnik, V., 2002. Gene selection for cancer classification using support vector machines. Machine learning 46, 389–422.

Han, Q., Hu, L., Gao, W., 2024. Feature relevance and redundancy coefficients for multi-view multi-label feature selection. Information Sciences 652, 119747.

Harris, C.R., et al., 2020. Array programming with NumPy. Nature 585, 357–362. doi:10.1038/s41586-020-2649-2.

He, Z., Yu, W., 2010. Stable feature selection for biomarker discovery. Computational biology and chemistry 34, 215–225.

Jurman, G., Riccadonna, S., Visintainer, R., Furlanello, C., 2009. Canberra distance on ranked lists, in: Proceedings of advances in ranking NIPS 09 workshop, Citeseer. pp. 22–27.

Kalousis, A., Prados, J., Hilario, M., 2007. Stability of feature selection algorithms: a study on high-dimensional spaces. Knowledge and information systems 12, 95–116.

Khaire, U.M., Dhanalakshmi, R., 2019. Stability of feature selection algorithm: A review. Journal of King Saud University-Computer and Information Sciences .

Kuncheva, L.I., 2007. A stability index for feature selection., in: Artificial intelligence and applications, pp. 421–427.

Lazar, C., et al., 2012. A survey on filter techniques for feature selection in gene expression microarray analysis. IEEE/ACM Transactions on Computational Biology and Bioinformatics 9, 1106–1119.

Lin, S., 2010. Rank aggregation methods. Wiley Interdisciplinary Reviews: Computational Statistics 2, 555–570.

Loh, W.Y., 2011. Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1, 14–23.

Maaten, L.v.d., Hinton, G., 2008. Visualizing data using t-sne. Journal of machine learning research 9, 2579–2605.

Wes McKinney, 2010. Data Structures for Statistical Computing in Python, in: Stéfan van der Walt, Jarrod Millman (Eds.), Proceedings of the 9th Python in Science Conference, pp. 56 – 61. doi:10.25080/Majora-92bf1922-00a.

Miao, J., Niu, L., 2016. A survey on feature selection. Procedia Computer Science 91, 919–926.

Mohana Chelvan, P., Perumal, K., 2016. A survey on feature selection stability measures. International Journal of Computer and Information Technology 5, 98–103.

Molnar, C., 2020. Interpretable machine learning. Lulu. com.

Njoku, U.F., Abelló, A., Bilalli, B., Bontempi, G., 2022. Impact of filter feature selection on classification: an empirical study, in: Proceedings of the 24th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP), CEUR. pp. 71–80.

Nogueira, S., Sechidis, K., Brown, G., 2017. On the stability of feature selection algorithms. The Journal of Machine Learning Research 18, 6345–6398.

Pedregosa, F., et al., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

Peng, H., Long, F., Ding, C., 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Transactions on pattern analysis and machine intelligence 27, 1226–1238.

Pes, B., 2019. Ensemble feature selection for high-dimensional data: a stability analysis across multiple domains. Neural Computing and Applications , 1–23.

Powers, D.M., 2020. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. arXiv preprint arXiv:2010.16061 .

Reis, M.S., Estrela, G., Ferreira, C.E., Barrera, J., 2017. featsel: A framework for benchmarking of feature selection algorithms and cost functions. SoftwareX 6, 193–197.

Rish, I., et al., 2001. An empirical study of the naive bayes classifier, in: IJCAI 2001 workshop on empirical methods in artificial intelligence, pp. 41–46.

Robnik-Šikonja, M., Kononenko, I., 2003. Theoretical and empirical analysis of relieff and rrelieff. Machine learning 53, 23–69.

Saberi-Movahed, F., Rostami, M., Berahmand, K., Karami, S., Tiwari, P., Oussalah, M., Band, S.S., 2022. Dual regularized unsupervised feature selection based on matrix factorization and minimum redundancy with application in gene selection. Knowledge-Based Systems 256, 109884.

Sakar, C.O., Polat, S.O., Katircioglu, M., Kastro, Y., 2019. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and lstm recurrent neural networks. Neural Computing and Applications 31, 6893–6908.

Salman, R., Alzaatreh, A., Sulieman, H., 2022. The stability of different aggregation techniques in ensemble feature selection. Journal of Big Data 9, 1–23.

Sheikhpour, R., Berahmand, K., Forouzandeh, S., 2023. Hessian-based semi-supervised feature selection using generalized uncorrelated constraint. Knowledge-Based Systems 269, 110521.

Shreem, S.S., Abdullah, S., Nazri, M.Z.A., Alzaqebah, M., 2012. Hybridizing relieff, mrmr filters and ga wrapper approaches for gene selection. J. Theor. Appl. Inf. Technol 46, 1034–1039.

Tadist, K., Najah, S., Nikolov, N.S., Mrabti, F., Zahi, A., 2019. Feature selection methods and genomic big data: a systematic review. Journal of Big Data 6, 1–24.

Tan, M., Hartley, M., Bister, M., Deklerck, R., 2009. Automated feature selection in neuroevolution. Evolutionary Intelligence 1, 271–292.

Vergara, J.R., Estévez, P.A., 2014. A review of feature selection methods based on mutual information. Neural computing and applications 24, 175–186.

Virtanen, P., Gommers, R., et al., 2020. Scipy 1.0: Fundamental algorithms for scientific computing in python. Nature Methods .

Watts, T., Xue, B., Zhang, M., 2019. Blocky net: A new neuroevolution method, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE. pp. 586–593.

Whiteson, S., Stone, P., Stanley, K.O., Miikkulainen, R., Kohl, N., 2005. Automatic feature selection in neuroevolution, in: Proceedings of the 7th annual conference on Genetic and evolutionary computation, pp. 1225–1232.

Willmott, C.J., Matsuura, K., 2005. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. Climate research 30, 79–82.

Yan, L., et al., 2020. An interpretable mortality prediction model for covid-19 patients. Nature Machine Intelligence 2, 283–288.

Orcid Information

Matheus Barbieri – 0000-0002-5389-706

Bruno Iochins Grisci - 0000-0003-4083-5881

Márcio Dorn - 0000-0001-8534-3480

Credit Author Statement

Matheus Cezimbra Barbieri: Conceptualization, Methodology, Software, Validation, Formal analysis, Data Curation, Writing - Original Draft, Visualization.

Bruno Iochins Grisci: Conceptualization, Methodology, Validation, Writing - Review Editing, Supervision.

Márcio Dorn: Conceptualization, Resources, Writing - Review Editing, Supervision, Project administration, Funding acquisition

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: