UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Hybrid filter-wrapper approaches for feature selection

Bachelor's Degree in Computer Engineering

Computation specialization

David Gili Fernández de Romarategui

Director: Lluís Antonio Belanche Muñoz

GEP Tutor: Marcos Eguiguren Huerta

Defense: 28th October 2021

Acknowledgements

First and foremost, I would like to thank the director of this thesis Lluís Antonio Belanche Muñoz, without whom this project could not have been successfully completed. Despite difficult circumstances and amidst a global pandemic, his guidance, support and flexibility have been of utmost importance for the development of this thesis.

I would like to thank my friends and especially my university peers. Although this marks the end of this period, having them aside has made the journey much more enjoyable.

Finally, I can not express in words how thankful I am to my brother and mother. This has been a nerve-wracking year for me, and I am sure none of this would be possible without their unconditional support.

Abstract

Over the last couple of decades, more business sectors than ever have embraced digital technologies, storing all the information they generate in databases. Moreover, with the rise of machine learning and data science, it has become economically profitable to use this data to solve real-world problems. However, as datasets grow larger, it has become increasingly difficult to determine exactly which variables are valuable to solve a given problem.

This project studies the problem of feature selection, which tries to select a subset of relevant variables for a specific prediction task from the complete set of attributes. In particular, we have mostly focused on hybrid filter-wrapper algorithms, a relatively new branch of study, that has seen great success in high-dimensional datasets because they offer a good trade-off between speed and accuracy.

The project starts by explaining several important filter and wrapper methods and moves on to illustrate how several authors have combined them to form new hybrid algorithms. Moreover, we also introduce a new algorithm called BWRR, which uses the popular ReliefF filter to guide a backward wrapper search. The key novelty we propose is to recompute the ReliefF rankings at several points to better guide the search. In addition, we also introduce several variations of this algorithm.

We have also performed extensive experimentation to test this algorithm. In the first phase, we experimented with synthetic datasets to see which factors affected the performance. After that, we compared the new algorithm against the state-of-the-art in real-world datasets.

Keywords: feature selection, hybrid filter-wrapper, Relief based algorithm, high-dimensional datasets.

Resum

Durant les darreres dècades, molts sectors empresarials han adoptat les tecnologies digitals, emmagatzemant tota la informació que generen en bases de dades. A més, amb l'auge de l'aprenentatge automàtic i la ciència de les dades, s'ha tornat econòmicament rendible utilitzar aquestes dades per resoldre problemes del món real. No obstant això, a mesura que els conjunts de dades creixen en mida, cada vegada és més difícil determinar exactament quines variables són valuoses per resoldre un problema específic.

Aquest projecte estudia el problema de la selecció de variables, que intenta seleccionar el subconjunt de variables rellevants per a una determinada tasca predictiva. En particular, ens centrarem en els algoritmes híbrids que combinen mètodes filtre i embolcall. Aquesta és una àrea d'estudi relativament nova, que ha obtingut bons resultats en conjunts de dades amb grans dimensions perquè ofereixen un bon compromís entre velocitat i precisió.

El projecte començarà explicant diversos mètodes filtre i embolcall i seguidament ensenyarà com diversos autors els han combinat per obtenir nous algoritmes híbrids. També introduirem un nou algoritme al qual anomenarem BWRR, que utilitza el popular filtre ReliefF per guiar una cerca cap enrere. La principal novetat que proposem és recomputar ReliefF en certs punts per guiar millor la cerca. Addicionalment, introduirem diverses variacions de l'algoritme.

També hem realitzat una extensa experimentació per a provar el nou algoritme. Primerament, hem treballat amb conjunts de dades sintètiques per esbrinar quins factors afectaven el rendiment. Seguidament, l'hem comparat amb l'estat de l'art en diversos conjunts de dades reals.

Paraules clau: selecció de variables, algoritmes híbrids filtre-embolcall, algoritmes basats en Relief, conjunts de variables de grans dimensions.

Resumen

Durante las últimas décadas, muchos sectores empresariales han adoptado las tecnologías digitales, almacenando toda la información que generan en bases de datos. Además, con el auge del aprendizaje automático y la ciencia de datos, se ha vuelto económicamente rentable utilizar estos datos para resolver problemas del mundo real. Sin embargo, a medida que los conjuntos de datos aumentan de tamaño, cada vez se vuelve más difícil determinar exactamente qué variables son valiosas para resolver un problema específico.

Este proyecto estudia el problema de la selección de variables, que intenta seleccionar el subconjunto de variables relevantes dada una determinada tarea predictiva. En particular, nos centraremos en los algoritmos híbridos que combinan los métodos filtro y envoltura. Esta es una área de investigación relativamente reciente, que ha obtenido buenos resultados en conjuntos de datos de grandes dimensiones porque ofrecen un buen compromiso entre velocidad y precisión.

El proyecto empezara explicando los métodos filtro y envoltura y, seguidamente, enseñara como diversos autores los han combinado para formar nuevos algoritmos híbridos. También introduciremos un nuevo algoritmo al que llamamos BWRR, que utiliza el popular filtro ReliefF para guiar una búsqueda hacia atrás. La principal novedad es que se recomputa ReliefF en ciertos puntos para así guiar mejor la búsqueda. Adicionalmente, introduciremos diversas variaciones del algoritmo.

También hemos realizado una extensa experimentación para probar el nuevo algoritmo. Primeramente, trabajaremos con conjuntos de datos sintéticos para descubrir que factores afectan el rendimiento. Seguidamente, lo compararemos con el estado del arte en diversos conjuntos de datos reales.

Palabras clave: selección de variables, algoritmos híbridos filtro-envoltura, algoritmos basados en Relief, conjuntos de variables de grandes dimensiones.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

For many years, datasets were small enough that feature selection could be seen as an optional preprocessing step. However, since the recent Big Data explosion, it has become a must in order to deal with high-dimensional datasets. As stated by Yvan Saeys, Iñaki Inza and Pedro Larrañaga [1], "During the last decade, the motivation for applying feature selection (FS) techniques in bioinformatics has shifted from being an illustrative example to becoming a real prerequisite for model building".

The large majority of research done in the field of feature selection has targeted either filter or wrapper methods individually. However, the investigation into hybrid filter-wrapper algorithms did not become a prominent area of research until this last decade. Therefore, we think that there is still potential to explore new ways of combining both methods.

Of all possible branches of feature selection, we decided to focus our research on hybrid filter-wrapper algorithms because they offer a good trade-off between computational speed and performance. They serve as a middle ground between the filter and wrapper approaches, being faster than wrappers while selecting better subsets than filters. These characteristics have made hybrid algorithms outperform traditional feature selection approaches in problems with large feature spaces. For example, as shown by Nada Almugren and Hala Alshamlan [2], several hybrid algorithms have become state of the art in different prominent cancer classification problems.

A big focus of this project will be put on the ReliefF filter, this is because it will be our filter of choice used in our hybrid algorithm. Relief-based algorithms (RBA) are a family of algorithms that perform well in many different feature selection problems. However, after studying the published literature, we found that there is a lack of hybrid algorithms that combine RBAs with sequential wrappers.

1.2 Goals

- **Study relevant feature selection literature:** Study filter, wrapper and hybrid filter-wrapper feature selection approaches.
- **Implement a new hybrid algorithm:** Design a hybrid feature selection algorithm using an algorithmic combination of both filter and wrapper techniques.
- **Analyse the algorithm theoretically:** Perform a computational analysis explaining the time and space complexities.

- **Design a synthetic benchmark:** This will allow us to better control key variables such as the number of instances, noise or the number of irrelevant and relevant variables. Moreover, we will be able to determine how each factor is affecting the performance.
- **Synthetic experimentation:** Compare the proposed algorithm with the pure filter and wrapper approaches using synthetic datasets. For the project to be a success the new algorithm should be faster than wrappers and have better final subsets than filters.
- **Real-world experimentation:** Compare the proposed algorithm with several selected hybrid state-of-the-art algorithms using real-world datasets. These real-world datasets will mostly include high-dimensional datasets such as microarray gene data.
- **Results analysis:** Analyse the results obtained in the experimentation and discuss whether or not the proposed algorithm improves on the state-of-the-art algorithms in any aspect.

1.3 Document structure

We will briefly explain how the document is organized:

- **Section 2:** Introduces the reader to the feature selection problem and explains the different families of solutions.
- **Section 3:** Explains more in depth the filter paradigm, with a special focus on Relief based algorithms.
- **Section 4:** Briefly describes sequential wrapper methods.
- **Section 5:** Illustrates different state-of-the-art hybrid algorithms.
- **Section 6:** We propose a new hybrid algorithm, as well as several different variations of it.
- **Section 7:** Detailed description of the experimentation setup. It explains how the synthetic datasets were made, which factors were considered important and which evaluation measure is used.
- **Section 8:** Discussion of the results obtained in the experimentation phase.
- **Section 9:** Overall project conclusions.
- **Section 10:** Section dedicated to project management. It includes tasks, temporal planning, an estimated budget and the sustainability report.
- **Section 11:** Technical competences of the bachelor's thesis.
- **Section 12:** Complete list of tables and figures generated in the experimentation phase.

2 Concepts

This section introduces the reader to the problem of feature selection and explains the different families of solutions and the current state-of-the-art.

2.1 Feature selection

In machine learning, feature selection (FS) is a process that tries to find a minimum size subset of features without losing valuable information about the target class. It does so, in general, by trying to identify and remove redundant or irrelevant features. Some of the benefits of applying feature selection are the following:

- Removing irrelevant o nearly irrelevant features allows for a better and clearer understanding of the data. It also helps point out underlying data relations that may not be obvious at the start.
- Reduces drastically the computational cost of some learning algorithms.
- Improves overall accuracy results. Not always having more information results in a better overall score. In fact, the removal of irrelevant features has proven to improve the performance of several classifiers.
- Improves generalization and reduces overfitting.
- Avoid the curse of dimensionality. This term refers to problems that appear when the dimensionality increases and the data becomes sparse. This sparsity can be a problem for any algorithm that uses statistical significance. In these conditions, to obtain reliable results the amount of data needs to increase exponentially with the dimensionality.

The feature selection problem can also be formally expressed. Having a complete set F with class label C, Lei Yu and Huan Liu defined the problem as trying to find the minimum size subset S, $S \subseteq F$, such that $P(C \mid S)$ is as close as possible to $P(C \mid F)$ [3]. Although it may seem simple at first, it is in fact NP-hard.

Many algorithms, specially filters, do not perform direct feature selection. Instead, they choose to solve a similar problem called **feature weighting**. This is considered a more general problem that tries to determine weight values that approximate the degree of relevance of each individual feature.

The main difference between both approaches is that feature selection outputs a subset of features, while feature weighting outputs an array of weights. Despite this many feature weighting algorithms are used in feature selection problems because given an array of weights $w : [w_0, w_1, \ldots w_f]$ and a cutoff p, we can obtain a feature subset S by just selecting features that have $w_i >= p$. Alternatively, one can also obtain a feature subset by sorting the weight vector decreasingly and selecting the k best features.

2.2 Relevance and redundancy

To determine the quality of the final subset it important to define the concept of relevance. John, Kohavi, and Pfleger distinguished three disjoint categories: strongly relevant, weakly relevant, and irrelevant [4]. Let F be a full set of features, f_i a feature, C the class attribute and $S_i = F - \{f_i\}$. These categories are defined as follows:

- **Strongly relevant:** A feature f_i is strongly relevant iff

$$P(C \mid f_i, S_i) \neq P(C \mid S_i)$$

- **Weakly relevant:** A feature f_i is weakly relevant iff

$$P(C \mid f_i, S_i) = P(C \mid S_i) \text{ and}$$

$$\exists\, S'_i \subseteq S \text{ such that } P(C \mid f_i, S'_i) \neq P(C \mid S'_i)$$

- **Irrelevant:** A feature f_i is irrelevant iff

$$\forall\, S'_i \subseteq S, \quad P(C \mid f_i, S'_i) = P(C \mid S'_i)$$

If we want to obtain the optimal subset of features, we should select all strongly relevant features, some weakly relevant and none of the irrelevant.

The concept of redundancy is harder to define. In its most simple form, two features are considered redundant if all their values are completely correlated. However, it is harder to know whether a feature f_i is redundant with a set of features S. Lei Yu and Huan Liu formalized the concept of redundancy using Markov blankets [3].

Markov blanket: Given a feature $f_i \in F$, a class C, a subset $S \subset F, f_i \notin S$ is a Markov blanket for f_i iff:

$$P(F - S - \{f_i\},\ C \mid f_i,\ S) = P(F - S - \{f_i\},\ C \mid S)$$

In simpler terms, f_i and S are a Markov blanket if, given S, f_i is conditionally independent of $F - S - \{f_i\}$ and C. This means that S, contains all information that f_i provides about C and all the other features in F.

Redundancy: Given a selected set of features F, a feature f_i is redundant from F iff it is weakly relevant and has a Markov blanket within F.

Moreover, it can be proven theoretically that if we find a Markov blanket of feature f_i in F, we can safely remove f_i.

2.3 Families of solutions

The traditional way to classify feature selection algorithms is to divide them into 3 subgroups: filter, wrapper and embedded. Additionally, we can also find hybrid algorithms that algorithmically combine several approaches.

Filter

Filter methods are those that use intrinsic statistical measures of the data to evaluate features. The main characteristic of filter approaches is that they do not require the use of a specific learning algorithm and therefore are considered classifier agnostic. Most filter methods do not perform feature selection, instead, they most often do feature weighting.

Filter algorithms can also be further subdivided by which type of measure they use to assess feature relevance [5]. Being the most important ones: information, distance, consistency, similarity and statistical measures. Another way to classify filter methods is to consider whether or not they analyze features separately or not. Univariate algorithms will rank features individually, ignoring possible feature dependencies, whereas multivariate methods rank multiple features trying to taking into account feature interactions. The univariate methods are generally much faster, but they can not identify redundant features or consider feature interactions.

Some of the advantages of filter feature selection, as stated by Sánchez-Maroño N., Alonso-Betanzos A. and Tombilla-Sanromán M, are that they are less computationally expensive, independent of the induction algorithm and normally result in a better generalization [6]. However, the final subset of features often ends up being large and the final prediction score tends to be worse.

Wrapper

Wrapper methods use a classifier to assess the quality of feature subsets and try to find the subset that maximizes the classifier accuracy with the least amount of features. There are 2^n possible subsets, so a brute force wrapper solution involves checking which of the possibilities yields the best scores. However, this brute force approach is too computationally taxing for most datasets. Therefore, it is standard to use metaheuristics algorithms that try to find a good enough subset, although they do not guaranty an optimal solution.

The major advantage of wrapper methods is that it has been empirically proven that because subsets are evaluated using a real modeling algorithm, a better accuracy and a smaller final subset size can be obtained. On the contrary, their major flaw is that they are computationally expensive and thus see limited use in high-dimensional datasets. Another thing to keep in mind is that wrapper methods are prone to larger generalization errors.

Embedded

Embedded algorithms aim to combine the feature selection process with the modeling algorithm's execution. They do so normally by optimizing two-part objective functions where a penalty for large feature sets is set in place. Some embedded algorithms include Lasso, Elastic Net and Random Forests.

Embedded algorithms have better computational complexity than wrapper methods while still utilizing a classifier. They are also regarded as being more complex than filter methods.

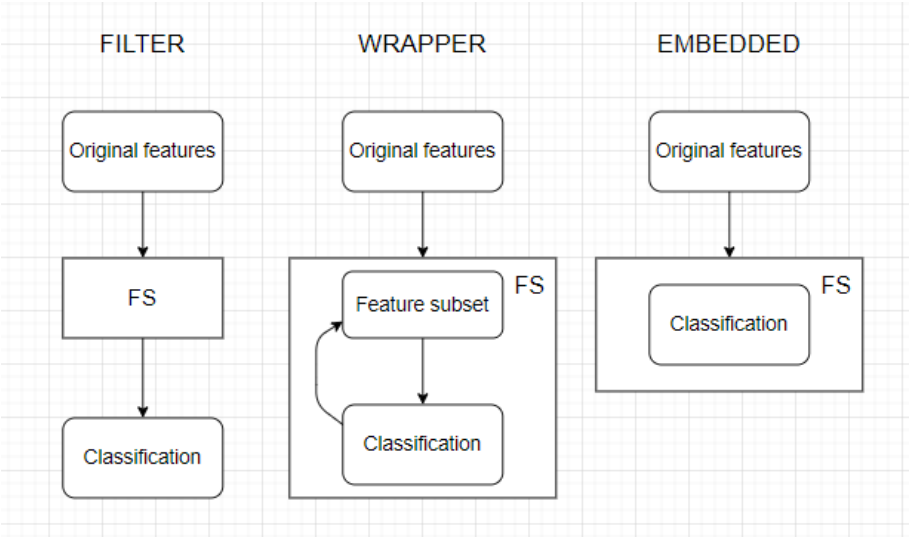

Figure 1: Representation of different types of FS.

Hybrid algorithms

Hybrid algorithms try to algorithmically combine filter, wrapper and embedded approaches to obtain better feature selection algorithms. In this project, we will only discuss filter-wrapper hybrid algorithms, which are those that are a combination of filter and wrapper approaches. Normally this combination is done in a way that the new algorithms will combine the best characteristics of both approaches. For example, some hybrid algorithms employ the two-stage approach where a filter algorithm is executed on the original feature space to remove some features. The much more computationally expensive wrapper can now be executed on the reduced set to better select the final features. This approach is fast because it uses the fast filter as preliminary screening, while still maintaining good final scores thanks to the refinement wrapper phase.

Although there is not a formal sub-classification for hybrid algorithms we will distinguish between two types:

- Two-stage algorithms: They first execute the filter to perform an initial pruning. Thereafter, the smaller feature space is then fed to the wrapper feature selection algorithm that gives the final subset.

- Filter guided wrappers: They use the ranking produced by the filter to guide the wrapper search.

Hybrid algorithms have been a way to incorporate wrappers into high-dimensional datasets. In large feature spaces, the pure wrapper approach becomes computationally intractable because the number of classifier evaluations at best scales quadratically with the number of features.

2.4 Previous work

The problem of feature selection has been an open area of research for many years.

Regarding the filter approach, some of the most notable algorithms include: distance-based such as Relief, ReliefF; information-based such as Information Gain (IG) and statistical-based like Fisher score (FS). [5]

Research on wrappers has mostly focused on analyzing how different metaheuristics behave when applied to feature selection. Some of them are Simulated Annealing (SA), Tabu Search (TS), Particle Swarm Optimization (PSO). There are also some evolutionary-based algorithms such as Genetic Algorithm (GA) and other nature-inspired algorithms such as Ant Colony Optimization (ACO). [7]

However, it has not been until the last decade that hybrid filter-wrapper approaches started to gain traction. A proven way to combine both involves performing the filter method as a preprocessing step to do an initial reduction of the feature space. After that, a wrapper method is executed to obtain the final subset. This has proven to avoid high computational costs while still maintaining good final scores. [2].

An example of this methodology was proposed by Lin Sun et al. They used a hybrid combination of the distance-based filter ReliefF and an Ant Colony Optimization (ACO) wrapper approach [8]. The proposed algorithm was named RFACO-GS and proved to be competitive in Tumor Classification problems.

Several researchers have also focused on algorithms that use the ranking provided by the filter to better guide the wrapper search. This approach is based on the concept of incremental ranked usefulness and the first algorithm of its kind was presented by Ruiz, Riquelme and Aguilar-Ruiz [9]. This is an active area of research and several algorithms have been proposed based on this approach [10][11].

Another recent line of research involves combining multiple filter or wrapper algorithms to perform feature selection. Yukyee Leung and Yeungsam Hung introduced the Multiple-filter-Multiple-Wrapper (MFMW) model [12]. This new approach combined the results of multiple filters, using the union of ranking lists, and after performing voting with multiple classifiers in the wrapper phase. With this approach, they managed to improve the accuracy and robustness of the classification.

3 Filter algorithms

This section explains the filter algorithms that are later used in the project. It first describes information-based filters which are used the most in hybrid algorithms. Later on, we focus on the ReliefF family which are used in our proposed algorithms.

3.1 Information-based

Information-based algorithms are a particular branch of filters that integrate information theory in their evaluation method [13]. The base measure of information theory is the entropy, which is a measurement of the uncertainty of a random variable. For discrete variable x with mass probability $p(x)$ is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2(p(x_i)) \tag{1}$$

Similarly the joint entropy of two variables x and y can be expressed as:

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{n} p(x_i, y_j) \log_2(p(x_i, y_j)) \tag{2}$$

It is also important to define the conditional entropy between two variables:

$$H(X \mid Y) = H(Y) - H(X,Y) \tag{3}$$

3.1.1 Mutual information

The mutual information (MI) measures how much information of a random variable is contained by another variable [14]. It is formally defined as:

$$I(X;Y) = H(X) – H(X \mid Y) \tag{4}$$

In feature selection this measurement is normally used to quantify how much information about the target each feature gives us. After that, the features are ranked and the k best selected.

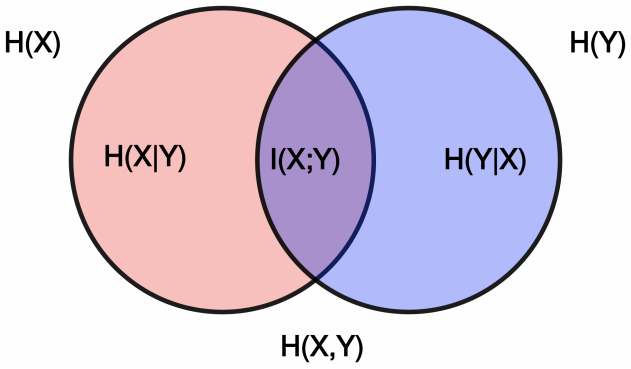

Figure 2: Venn diagram of information measures for two variables

3.1.2 Symmetrical Uncertainty

The symmetrical uncertainty (SU) is a nonlinear information-based measure [15]. It can be interpreted as a mutual information normalized to the interval $[0, 1]$. It is defined as follows:

$$SU(X, Y) = \frac{2 \cdot I(X; Y)}{H(Y) + H(X)} \tag{5}$$

This normalization compensates for mutual information's bias towards features having a large number of different values.

3.1.3 Joint Mutual Information

The metrics previously introduced all rank features according to how much information about the target they contain. However, if we were to select the k first features, we may find that many of the selected features are redundant with others.

Joint mutual information (JMI) was one of the first filter schemes that tried to balance the relevance-redundancy trade-off [16]. The JMI score of a variable X_k and target Y considering an already selected subset S is defined by:

$$\begin{aligned} JMI(X_k) &= \sum_{X_j \in S} I(X_k X_j; Y) = \sum_{X_j \in S} [I(X_k; Y \mid X_j)] \\ &= I(X_k; Y) - \frac{1}{|S|} \sum_{X_j \in S} [I(X_k; X_j) - I(X_k; X_j | Y)] \end{aligned} \tag{6}$$

Where $I(X_k X_j; Y)$ is the mutual information between the target and a joined random variable $X_k X_j$ and where $I(X_k; Y \mid X_j)$ is the conditional mutual information (CMI) which is the mutual information between X_k and Y conditioned to the knowledge of X_j.

The JMI score of variable X_j can be seen as the sum of MI obtained when pairing it with each feature already selected. The idea behind this approach is to accept features that are complementary to the selected subset.

A feature selection framework where the JMI metric is employed is the Joint Mutual Information Maximisation (JMIM) [17], which is a filter algorithm that uses JMI combined with a forward greedy approach to select features. In the first iteration the feature with the highest MI score is added to the selected subset S. After that, at iteration the features are selected according to this criterion:

$$JMIM = arg\,max_{X_i \in F - S}(min_{X_j \in S}(I(X_i, X_j; C))) \tag{7}$$

3.2 Relief-based algorithms

Relief-based algorithms (RBA) are instance-based algorithms. They try to estimate the relevance of attributes according to how well their values distinguish between the instances that are near each other. In general, the weights are calculated by a negative update in which the attributes that are different for near instances of the same class (hits) are penalized, and a positive update that rewards attributes that can separate near instances from different classes (misses). All algorithms of this family share these common traits:

- **Selecting the near neighbours:** All RBAs find near instances from the randomly selected instance R_i. Some algorithms just select the first k near instances, while others consider all instances that lay in a determined radius.
- **Able to detect feature interactions:** All RBAs have shown the capacity to detect 2-way interactions between features.
- **Computational cost:** All non-iterative RBAs support an asymptotic complexity of $O(n^2 \cdot a)$. This comes mostly from the computation of the distance matrix.
- **Anytime algorithms:** They estimate the feature weights by averaging the results obtained over multiple iterations. However, one can obtain valid results by stopping it at any time.

3.2.1 Relief

Relief was proposed by Kira in 1992 [18]. It introduced the idea of trying to estimate the quality of attributes depending on how well their values distinguish between instances that are near. The original Relief could only be used in binary classification problems and it had problems in noisy datasets.

Algorithm 1 Relief

Input: N instances; A features; T number of iterations;
Output: weighted array of features

1: $W = [0.0] * A$
2: **for** $i = 1$ to T **do**
3: randomly select instance R_i
4: find nearest hit H and miss M
5: **for** $a = 1$ to A **do**
6: $W[a] = W[a] - \text{diff}(a, R_i, H)/T + \text{diff}(a, R_i, M)/T$
7: **end for**
8: **end for**

The core algorithm worked as follows: first, an instance is randomly selected (line 3). Then the nearest hit (H) which is the closest instance with the same target and the

nearest miss (M) which is the closest instance of different class would be selected. (line 4). Finally, the estimates would be updated (line 6). This process is repeated T times, in each iteration a new random instance is selected as a pivot.

The diff(A, I_1, I_2) function is used to calculate the distance between two instances I_1, I_2. It is computed differently depending on the type of feature. For nominal features, it is defined as:

$$\text{diff}(A, I_1, I_2) = \begin{cases} 0 & value(A, I_1) \neq value(A, I_2) \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

If the attribute is numerical, a normalized Euclidean distance is used:

$$\text{diff}(A, I_1, I_2) = \frac{|value(A, I_1) - value(A, I_2)|}{max(A) - min(A)} \tag{9}$$

It is worth noting that depending on the choice of T, Relief has some randomness and different executions could yield different results. This randomness comes from the fact that if $T < n$, not all instances will be chosen as pivot R_i (line 3). It is possible to obtain stable results just by making $T = M$. This variation is often referred to as **Relieved** and eliminates the randomness by selecting all instances as pivot. This makes the asymptotic complexity raise from $O(T \cdot n \cdot a)$ to $O(n^2 \cdot a)$. The $T = n$ assumption is often made in literature when analysing RBA complexity. Therefore, we will follow the same convention in this document.

3.2.2 ReliefF

Kononenko analysed the base Relief algorithm and proposed several new versions of the algorithm that introduced several improvements [19]. They also proposed a general version named ReliefF, which combined the best characteristics of the rest.

ReliefF improved its predecessor in several different ways:

- **More robust to noise:** Taking the k nearest hits and misses and averaging the results instead of just taking the nearest, makes ReliefF much more robust to noise.
- **Extension to multiclass problems:** The key to extending Relief to multiclass problems is selecting the k nearest misses from each class (line 6) and slightly adapting the metric (line 10). To compute the new metric, each near miss positive update is weighted by the prior probability of its class.
- **Extension to missing data:** The diff function was extended to cases where one or both instances contained missing values (Eq 10):

$$\text{diff}(A, I_1, I_2) = \begin{cases} 1 - P(values(A, I_2 \,|\, class(I_1)) & I_1 \text{ unknown} \\ 1 - \sum_V^{\#values(A)} P(V \,|\, class(I_1)) * P(V \,|\, class(I_2)) & I_1, I_2 \text{ unknown} \end{cases} \tag{10}$$

Algorithm 2 ReliefF

Input: N instances; A features; K selected neighbours; T number of instances;
Output: weighted array of features

1: $W = [0.0] * A$
2: **for** $i = 1$ to T **do**
3: randomly select instance R_i
4: find K nearest hits H_j
5: **for all** class $C \neq class(R_i)$ **do**
6: select K nearest misses from class C, $M_j(C)$
7: **end for**
8: **for** $a = 1$ to A **do**
9: $W[a] = W[a] - \sum_{j=1}^{k} \text{diff}(a, R_i, H_j)/T * k$
10: $+ \sum_{C \neq class(R_i} \left[\frac{P(C)}{1-P(class(R_i))}\right] \sum_{j=1}^{k} \text{diff}(a, R_i, M_j(C))/T * k$
11: **end for**
12: **end for**

Ever since ReliefF's conception, it has been the most used RBA algorithm, making the original algorithm obsolete. This is mostly because ReliefF is a much more general algorithm that can be used in more problems than the original, namely with datasets with noise, missing values or multi-class targets.

3.2.3 RReliefF

One of the major limitations of Relief and ReliefF was their inability to deal with regression problems. This was in part because prior algorithms used the selection of near hits (with same class) and misses (with different class). However, the concept of near misses and hits can not be used when the target is continuous.

Robnik-Ŝikonja, Kononenko propesed Regression ReliefF (RReliefF) [20]. The key difference is that instead of determining if two instances are a hit or miss, RReliefF determines if two instances' target values are different using probabilities.

$$W[A] = \frac{P_{diffC|diffA} P_{diffA}}{P_{diffC}} * \frac{(1 - P_{diffC|diffA}) P_{diffA}}{1 - P_{diffC}} \tag{11}$$

In each iteration, the algorithm re-estimates the probabilities P_{diffC} (line 8), P_{diffA} (line 6) and $P_{diffC|diffA}$ (line 9). After finishing the main loop, the weights are calculated according to eq. (11). Another novelty introduced by RReliefF is the function $d(i, j)$ which exponentially decreases the influence of an instance the further it is.

Algorithm 3 RReliefF

Input: N instances; A features; K selected neighbours; T number of iterations;
Output: weighted array of features

set all $N_{dC}, N_{dA}[A], N_{dC\&dA}[A], W[A]$ to 0;
for $i = 1$ to T **do**
  randomly select instance R_i
  find K nearest instances I_j
  for $j = 1$ to K **do**
    N_{dC} += diff$(\tau(\cdot), R_i, I_j) * d(i, j)$;
    for $a = 1$ to A **do**
      $N_{dA}[a]$ += diff$(a, R_i, I_j) * d(i, j)$;
      $N_{dC\&dA}[a]$ += diff$(\tau(\cdot), R_i, I_j) *$ diff$(a, R_i, I_j) * d(i, j)$
    end for
  end for
end for
for $a = 1$ to A **do**
  $W[a] = (N_{dC\&dA}[a]/N_{dC}) - ((N_{dA}[a] - N_{dC\&dA}[a])/(T - N_{dC}))$
end for

3.2.4 Iterative RBAs

The estimates produced by ReliefF are able to detect feature interactions because the nearest neighbours are computed using the entire vector of features. However, the ReliefF estimates are known to deteriorate in presence of many irrelevant attributes because near instances may be in fact far apart after removing these noisy features.

Several approaches have been tried to improve ReliefF estimates in highly irrelevant feature spaces. The most successful solutions are iterative RBAs which involve repetitive calls to the baseline ReliefF algorithm.

3.2.4.1 Tuned ReliefF (TuRF)

Moore, Jason H proposed Tuned ReliefF (TuRF) [21], an iterative method that was able to improve ReliefF estimates in presence of irrelevant attributes. In each iteration, the ReliefF estimates are computed and the lowest scoring features are eliminated. In the next iteration, the ReliefF weights are re-estimated with the reduced subset. After each iteration, the resulting weights are more accurate because nearest neighbour calculations include less noisy and irrelevant features.

In the original paper, TuRF is only combined with the ReliefF algorithm. However. it is worth noting that TuRF is just a recursive feature elimination framework that can be wrapped around other Relief-based algorithm.

Algorithm 4 TurF

Input: N learning instances described by A features; T iterations;
Output: weighted array of the remaining features

1: **for** $i = 1$ to T **do**
2: compute ReliefF
3: sort values decreasingly
4: remove worst T/A attributes
5: **end for**
6: return last ReliefF ranking for each remaining attribute

3.2.5 Analysis of Relief mesures

3.2.5.1 Interpretation of Relief metric

ReliefF by iteratively doing positive and negative updates is, in fact, calculating an approximation of the following probability:

$$\begin{aligned} W[A] = & P(\text{diff. value of A} \mid \text{nearest inst. from diff. class}) \\ & - P(\text{diff. value of A} \mid \text{nearest inst. from same class}) \end{aligned} \tag{12}$$

When using RReliefF, because the class values continuous, hits and misses can no longer be defined. Instead, the algorithm tries to estimate eq. (11).

3.2.5.2 Why is Relief able to detect interactions?

Perhaps the defining characteristic of Relief-based algorithms is that they can detect interactions between features. This ability is in fact due to the nearest condition in the hits and misses selection. It has been observed experimentally, that when the number of selected nearest neighbours approaches the number of instances, the algorithm becomes unable to detect interactions. Kononenko also analysed the Relief measure without the nearest condition, which by convention is named **myopic Relief**, and concluded that it is closely related with the Gini index, a myopic impurity function that is unable to detect feature interactions [22]. If we eliminate the near condition in eq. (12), the probability being estimated becomes:

$$\begin{aligned} W[A] = & P(\text{diff. value of A} \mid \text{different class}) \\ & - P(\text{diff. value of A} \mid \text{same class}) \end{aligned} \tag{13}$$

Kononenko showed that by applying several transformations, it is possible to express the myopic Relief's probability estimate in terms of the Gini gain.

$$W'[A] = \frac{P_{equal} * \mathit{Gini\ Gain}}{P_{samecl} * (1 - P_{samecl})} \tag{14}$$

This direct relation between the myopic Relief weights and the Gini Index allows us to compare both estimates. The Gini Index is an impurity function that assumes conditional independence between the attributes. This assumption makes it unable to detect interactions between attributes.

This fact makes it clear that the near condition must be the key that allows Relief to take into account feature interactions. An explanation to this is given by Kononenko [22]. By only considering near instances Relief is estimating the average over local estimates in smaller parts of the instance space. This allows Relief to detect feature dependencies that are inappreciable at a global scale and can only be detected by taking into account locality.

This effect can be easily visualized by computing the separability and usability of ReliefF's rankings while changing the number of neighbours. Separability refers to the difference between the worst-ranked relevant feature and the highest-ranked irrelevant feature ($J_{R_{worst}} - J_{I_{best}}$). Usability is defined by the difference between the highest ranked important and irrelevant features ($J_{R_{best}} - J_{I_{best}}$).

In fig 3, we can see how increasing the number of neighbours does in fact decrease the quality of estimates. This explains why it is often recommended to set the number of neighbours to 5.

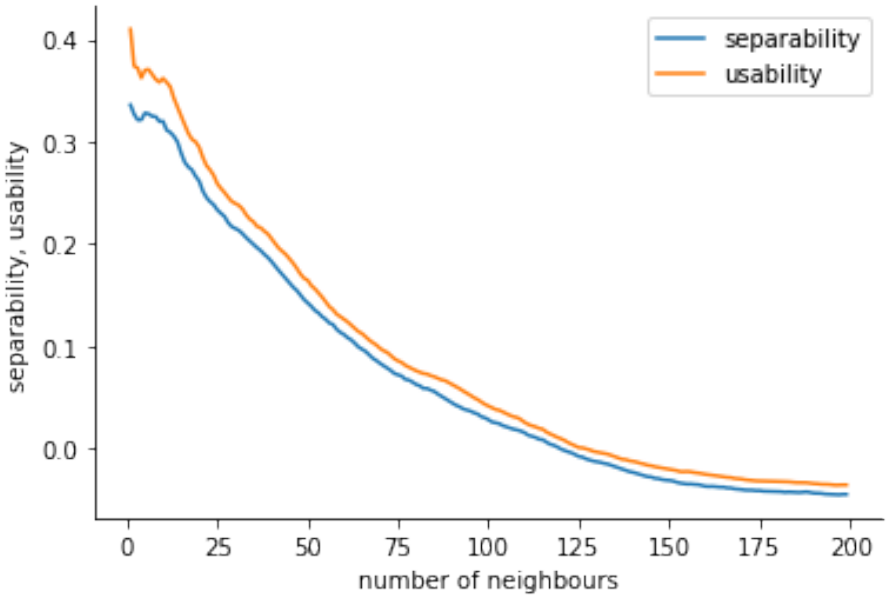

Figure 3: Separability and usability of ReliefF rankings depending on number of neighbours.

3.2.5.3 Do irrelevant features affect the Relief metric

It is also worth considering whether or not irrelevant and noisy features degrade the Relief weights. As mentioned before the elimination of the near condition makes

Relief unable to detect interactions. A similar thing happens when there is a large amount of noisy and irrelevant features. In such feature spaces, the selection of the nearest hits and misses becomes increasingly random. This is because, some instances may seem close when taking into account all features, but in reality, if we calculate the distances with only the relevant features, they may be far apart.

To better visualize this effect we performed a small test where the separability and usability of ReliefF's rankings were calculated while increasing the number of irrelevant features. In fig 4, we can clearly see that ReliefF estimates degrade as the number of irrelevant features increases.

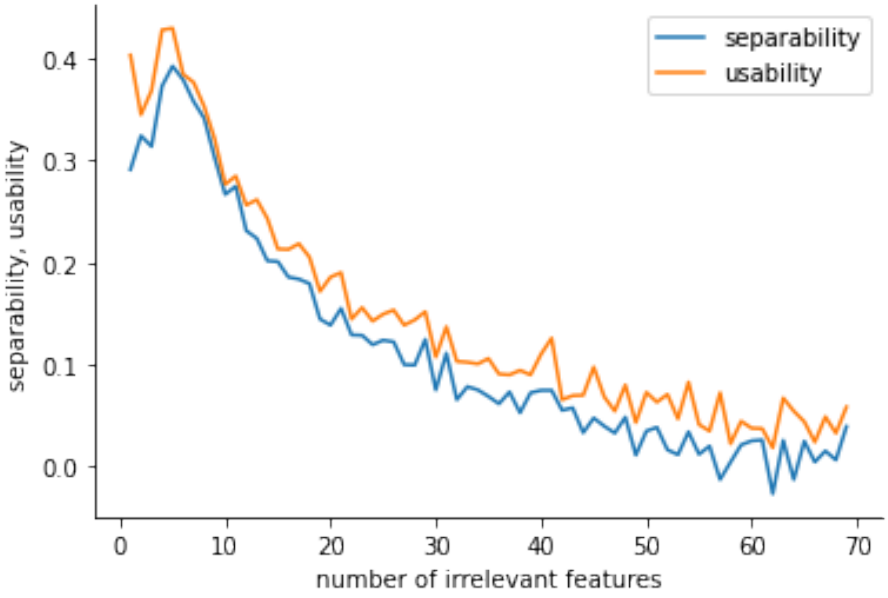

Figure 4: Separability and usability of ReliefF rankings depending on number of irrelevant features.

3.2.5.4 Selecting a cut-off

For each attribute Relief estimates a relevance score $1 \geq \delta \geq -1$, the greater this value is the more relevant the algorithm considers it. Nevertheless it is not easy to select a threshold that can separate relevant from irrelevant features. This is a common problem for filter algorithms.

Kira, studied possible thresholds τ to cut-off the features [18]. He also proposed bounds for this threshold, $0 < \tau \leq \frac{1}{\sqrt{\alpha m}}$ where m is the number of iterations and α the probability of accepting into the final subset an irrelevant feature.

In practice, we normally do not know the probability α for real world datasets. That is why the upper bound is generally impossible to know. When dealing with real world datasets, it is more common to select the final subset using an ad-hoc cutoff based on the computational power available or previous knowledge of the data.

4 Wrapper algorithms

In this section, we will briefly explain the wrapper approach. Additionally, we will focus on sequential wrappers because they will be used later in the project.

Wrapper methods essentially use a learning algorithm as a black box to score different feature subsets. This score will be obtained using holdout or cross-validation. Different wrapper algorithms are normally defined by their strategy to generate the different feature subsets that will be evaluated.

If we have a complete set of n features, optimal feature selection algorithms often require checking each potential feature subset, resulting in $O(2^n)$ number of evaluations. This is often too great of a limitation and sequential or meta-heuristic methods are more often used because they require fewer wrapper evaluations. Among all the different types, we will only go into detail about the sequential wrappers because they will be used later in this project.

4.1 Sequential wrappers

Sequential algorithms are greedy algorithms that select or remove the feature that maximizes the classification score at each iteration. Depending on the search strategy we differentiate between forward and backward algorithms.

In sequential forward selection (SFS) we start with the empty set ($S = \{\emptyset\}$). Each iteration, the feature that maximizes the score when added to the already selected subset will be permanently selected. ($x_{sel} = arg\,max_{x \notin S}\, J(S \cup x)$).

On the other hand in a sequential backward selection (SBS) we will start with the complete set ($S = \{X_1 \dots X_N\}$). Each iteration we will remove the feature that when doing so maximizes the score. ($x_{sel} = arg\,max_{x \in S}\, J(S \setminus x)$).

The main benefit of the backward approach is that as it starts with the complete set, the learning algorithm can consider feature interactions. However, this also means that on average it will be trained with more features, adding extra computational time. One advantage of the forward direction is that it is much easier to implement early stopping, this is because at iteration I in SFS the best I features have been selected while in SBS the I worst have been removed.

In the first iteration, we will need to run n evaluations, in the next $n-1$ and so on. It is easy to check that this results in $O(n^2)$ number of wrapper evaluations:

$$Eval = n + (n-1) + \cdots + 1 = \sum_{k=1}^{n} k = \frac{n(n+1)}{2} = O(n^2)$$

Algorithm 5 SFS

Input: N learning instances described by A features; K features to select;
Output: final feature subset

1: $S = \{\emptyset\}$
2: **for** $i = 1$ to K **do**
3: **for all** $X_j \notin S$ **do**
4: $J_j = J(S \cup X_j)$
5: **end for**
6: $J_{sel} = arg\, max(J_j)$
7: $S = S \cup X_{J_{sel}}$
8: **end for**

The main problem of sequential wrapper algorithms is that feature that are selected into the subset can not be removed later on. This often implies that the algorithm will often get stuck in a local optimum. To solve this problem we can use the floating variants of the sequential algorithms. This solution combines forward and backward approaches.

In sequential floating forward selection (SFFS) after each forward iteration, backward steps will be used to delete features from the selected subset as long as the objective function increases. Similarly in sequential floating backward selection (SFBS) after each backward step, a sequence of forward steps is executed.

The floating solutions often improve the quality of the selected feature set. The main downfall is the increase in the number of wrapper evaluations to $O(n^3)$.

5 Hybrid algorithms

In this section we explain some relevant hybrid algorithms that will be later used in the experimentation. We will distinguish between the two stage approach and the wrapper guided by filter approach.

5.1 Two stage approach

The most explored way to combine filter and wrapper approaches is to sequentially execute them one after the other in two different stages. In the first stage, the much faster filter algorithm is executed with the complete set. After that, a pruning of the worst weighted features leaves a smaller feature set. In the second stage, the wrapper algorithm is executed in the reduced search space. This combination is specially useful in high-dimensional datasets where executing a complete wrapper search is computationally unfeasible.

The biggest flaw of this approach is that if our filter is not ranking appropriately, some relevant features might be discarded before reaching the wrapper stage.

Another thing to consider is where to cutoff the filter ranking. It is common to just select a number of features that will be computationally tractable in the wrapper stage.

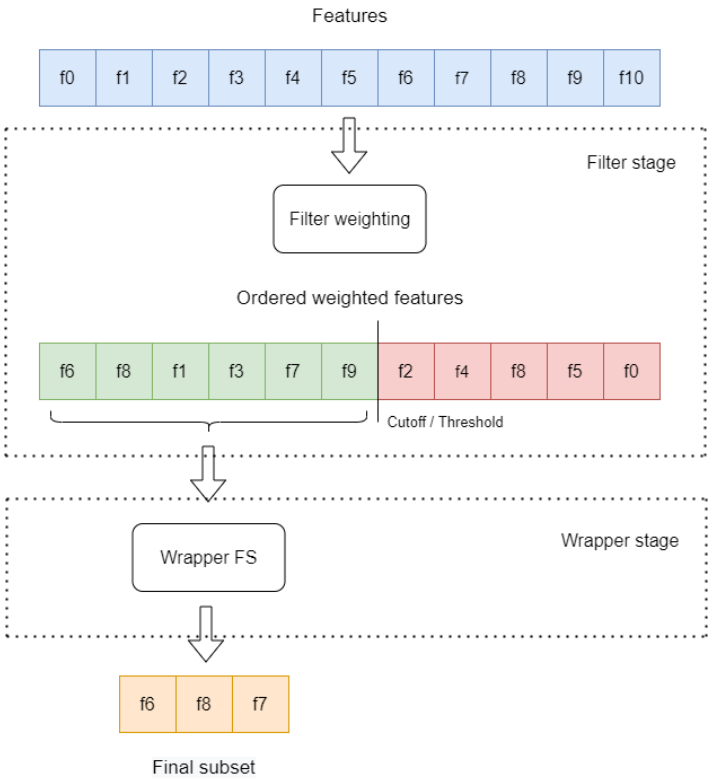

Figure 5: Representation of two stage hybrid FS.

Examples of the two stage approach include a PMI-based filter followed by a heuristic Firefly Algorithm search [23]. Another combination was proposed by Ding et al, who used Information Gain followed by a SFFS wrapper [24].

5.1.1 iSFSM

iSFSM is a more complex two-stage hybrid algorithm proposed by Hui-Huang Hsu et al [25]. In the first stage, candidate features are selected from the original feature set using the F-Score and Information Gain filters. These are both computationally efficient filters that excel at detecting redundant and irrelevant features respectively.

The main novelty of iSFSM is the way it selects the candidate set. After executing both filters, the top k best features from each ranking will be kept. We will name them X_{fs} and X_{ig}. After that each feature each separated into one of these sets:

- $X_{fs} \cap X_{ig}$: Features that are in the k best for both filters. This set contains the most promising features.
- $X_{fs} \oplus X_{ig}$: Features that appear in the k best for only one filter.
- $\neg X_{fs} \cap \neg X_{ig}$: Features that are not in the best k for both filters. These features will be discarded.

The filter step reduces the search region from the whole feature set to the union set of the two filters. Moreover, instead of starting the search from the empty set, it preselects all features in the intersection set. These reductions in the potential search space, result in much faster execution times when compared with the traditional wrappers. After, we move on to wrapper stage to fine tune the result. In this step, SBS is executed on $X_{fs} \cap X_{ig}$ to remove features that are not necessary. After that, it runs a SFS that adds features from $X_{fs} \oplus X_{ig}$ if they improve the result. This step is repeated iteratively until the test accuracy is not improved anymore.

5.1.2 MFMW

The main downfall of two-stage algorithms comes from the fact that some relevant features never reach the wrapper stage. Moreover, depending on the wrapper used different final subsets will be selected. To deal with this problem Yukyee Leung and Yeungsam Hung introduced the Multiple-Filter-Multiple-Wrapper (MFMW) approach [12].

Different filters use vastly different measures to assess feature relevance. This means that if we use a single filter model, it is possible for a relevant feature to be deemed irrelevant because our filter measurement does not capture its value properly. To minimize this, in the first stage, multiple filters are used to ensure no relevant feature

is not selected. For each filter, the first k features are selected, then the union of feature sets moves on to the wrapper stage.

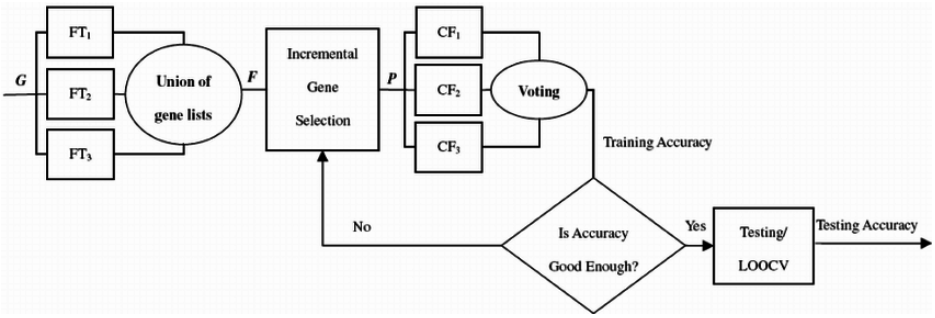

Figure 6: Multiple-Filter-Multiple-Wrapper diagram. [12]

Similarly, in the wrapper stage, multiple classifiers are used. It is not uncommon for different classifiers to predict different class labels for a given sample. In those cases, there is a need to implement a voting scheme to resolve conflict. In the original paper, majority voting was used.

5.2 Wrapper guided by filter approach

Wrapper search methods often require a quadratic number of evaluations, this is often too computationally expensive for larger datasets. A way to circumvent this problem is to use the rankings of a filter algorithm to guide the wrapper search. In such approaches, the much faster filter is executed first, after that, features are traversed by the wrapper algorithm from the best ranked feature to the worst. In each iteration, we use the concept of incremental usefulness to determine if the new feature is useful.

Definition of incremental usefulness: Given a dataset with X features and a learning algorithm L. A new feature x_i is said to be incrementally useful if scores obtain by L on the set $X \cup X_i$ are significantly better than those obtained on X.

Hybrid algorithms that follow this approach have the following common traits:

- **O(n) number of evaluations:** By traversing the features using the ranking, we only pass through each feature once, thus reducing the complexity from quadratic to linear.
- **Significant improvement testing:** To evaluate each subset k-fold cross-validation is performed. To determine if a subset is better than another several approaches can be followed. We could simply check if the mean of folds is better or we could perform some form of statistical testing.
- **Use of a filter that ranks features:** To guide the wrapper search we need a filter that ranks features.

5.2.1 BIRS

Ruiz et al. introduced the best incremental ranked subset (BIRS) [9]. This paper is the first to introduce the concept of a hybrid algorithm where the filter ranking is used to guide the wrapper search.

The algorithm starts by ranking all features using the symmetrical uncertainty (SU) filter (line 3). After that, the weights are ordered decreasingly. Then, the best score is set to zero and the selected set S initialized with the empty set. After, the algorithm loops through all features following the filter ranking. Each iteration, a feature is temporally added to the selected subset (line 9) and the classification score of the newly generated subset is obtained (line 10. If the new subset is significantly better than the current, the feature is permanently added and the best score is updated. This process is repeated for each feature until it reaches the last ranked one.

Another crucial aspect to consider is how to determine whether or not the subset with the newly added feature significantly improves the last subset (line 11). To determine this, a five-fold cross-validation is performed to estimate the classification accuracy for a given set of features. After that, a Student's paired two-tailed t-test is used to determine if the improvements are statistically significant. If the resultant p-value is under 0.1 the feature is accepted.

Algorithm 6 BIRS

Input: N instances; A features; class C; J_f filter scorer; J_w wrapper scorer
Output: final feature subset

1: $W = [0.0] * A$
2: **for** $i = 1$ to A **do**
3: $\quad W[i] = J_f(A_i, C)$
4: **end for**
5: order W decreasingly
6: $bestScore = 0$
7: $S = \emptyset$
8: **for** $i = 1$ to A **do**
9: $\quad S_{aux} = S \cup \{A_{W_i}\}$
10: $\quad score = J_w(C, S_{aux})$
11: $\quad$ **if** $score \triangleright bestScore$ **then**
12: $\quad\quad bestScore = score$
13: $\quad\quad S = S_{aux}$
14: $\quad$ **end if**
15: **end for**

* We will only include BIRS pseudocode because it constitutes the baseline hybrid wrapper guided by filter algorithm. We will also examine other similar algorithms, for them, we will just discuss the key changes they introduced.

5.2.2 IWSS

Bermejo et al. [10] experimentally analysed different relevance criteria for Incremental-wrapper FSS, including the paired t-test proposed in the BIRS algorithm. The results obtained suggested that the statistical measure used in BIRS was prone to overfitting. They also suggested an alternative relevance criteria named MinFoldBetter, which considers a subset better if its mean accuracy over a 5-fold cross-validation is better and improves in at least k out of the 5 folds. This k is a parameter of the algorithm and is normally set to 2 or 3.

With this new relevance criteria, they later proposed the Incremental Wrapper Subset Selection (IWSS), which follows the wrapper guided by filter search pattern but with the new improved relevance criteria.

Bermejo et al. also proposed some further improvements to the baseline IWSS algorithm. To reduce the number of wrapper evaluations they used early stopping. The termination condition is dictated by a new threshold parameter $\tau \in (0, 1]$. Initially, they suggested exploring only the first $N * \tau$ features before stopping. However, they later proposed to change to a dynamic stopping point. In this second approach, which was named IWSS_s, each time a new feature is accepted into the selected subset, the stopping point is moved further away.

Another novelty introduced in the paper is IWSS with replacement (IWSS_r). This approach, in similar fashion to the sequential floating algorithms allows to remove features that had already been added to the selected subset. To do so, at each iteration, the algorithm not only is allowed to accept a feature, but also to swap it by any other feature in the selected subset. The aim of this approach is to obtain more compact subsets. However, the introduced operation raises the number of wrapper evaluations to quadratic.

6 Proposed algorithms

In this section, we will propose several algorithms that are inspired by iterative RBAs and hybrid algorithms that use the filter ranking to guide the wrapper.

The main weakness of RBAs is that their estimates worsen in feature spaces with many irrelevant features. Iterative based RBAs, like TuRF, partially solve this problem. However, they still use a still not reliable ReliefF ranking to remove features in early iterations.

Most hybrid algorithms we find in the literature start from the empty set and perform a forward search. A weakness of this approach is that in early iterations, most features are found relevant, being the most obvious case the first iteration, where the best ranked feature is automatically accepted. Another downfall of the forward approach is that the learning algorithm is trained with small subsets, thus feature interaction is not being taken into account when testing if a feature is relevant.

Our proposed new algorithm, will try to solve both problems discussed above.

6.1 BWRR

We propose Backward Wrapper guided by ReliefF with Rerank (BWRR), a novel wrapper guided by filter hybrid algorithm. These are its defining characteristics:

- **Backward search:** Instead of the traditional forward search, we will perform a backward search starting from the complete set of features.
- **RBA filter:** We will use ReliefF instead of the traditionally used information-based filters to guide the search. RBAs have proven to be complete algorithms that perform well even in high-dimensions, however, we have seen a lack of use in combination with sequential wrappers.
- **Re-ranking:** Each time the learning algorithms deletes a feature from the selected set we will perform a re-rank. By deleting features that the learning algorithm considers irrelevant and recomputing ReliefF in a less noisy feature space, its estimates should improve, better guiding the wrapper search in the next iteration.

BWRR starts by computing the RBA estimates of the complete feature space and ordering the attributes increasingly. (line 1). After that, the wrapper score of the complete dataset is computed (line 4). Following that we start the iterative process. In each iteration, the feature with the worst filter score is temporally deleted from the subset (line 12) and the new subset is evaluated (line 13). If the new score has decreased significantly, the feature we eliminated was in fact useful, otherwise, we can permanently delete it from the subset. In this case, we recompute ReliefF with the reduced feature space to better guide the wrapper search in the next iteration (line 19).

Algorithm 7 BWRR_{v1}

Input: N instances; A features; J_w wrapper subset evaluator
Output: final feature subset

$W = RBA()$
order W increasingly
$S = \{A_1, A_2, \ldots, A_N\}$
$bestScore = J_w(S)$
for $i = 1$ to A **do**
  $S_{aux} = S \setminus \{A_{w_i}\}$
  $score = J_w(S_{aux})$
  if not $score \triangleleft bestScore$ **then**
    $bestScore = score$
    $S = S_{aux}$
    $W = RBA()$
  end if
end for

6.1.1 Theoretical base

The algorithm is based on the concept of incremental ranked usefulness, in which a feature is added to the selected subset if $L(X \cup x_i) \triangleright L(X)$. This same concept can be slightly modified to better suit a backward search approach. In this case a feature x_i is considered relevant when $L(X \setminus x_i) \triangleleft L(X)$.

We can distinguish several situations depending on the relevance of the deleted feature at each iteration:

- x_i is relevant: When removing a relevant x_i from X. The overall score of the new subset $X \setminus x_i$ should be lower than the one of set X. This is because x_i is relevant and therefore the subset X should contain more information about the class than $X \setminus x_i$.
- x_i irrelevant: If the removed feature x_i is completely irrelevant, then set $X \setminus x_i$ has the same meaningful information as X. Moreover there are several learning algorithms, like k nearest neighbours, that degrade its results in presence of irrelevant features. This means that when removing an irrelevant feature, it is not strange for the score to actually improve.
- x_i is redundant: Similarly, if we remove a redundant feature x_i, as x_i contains information already present in $X \setminus x_i$, the overall score of the new subset should be more or less the same as that of X.

Benefits

Our proposed elimination mechanism is a much safer option than regular iterative RBAs because it is a learning algorithm that has the final say in whether a feature is useful to solve the problem.

Another important advantage is that by following a backward search, the learning algorithm will be able to take into account all possible feature interactions. A known problem of the forward approach is that as we start with the empty subset, the interactions between features are mostly not taken into account. This is specially true in early iterations where the selected subset still has a small number of features.

Disadvantages:

The main disadvantage of following a backward sequential approach is the increased computation time. Although in both hybrid search directions the number of subset evaluations is the same, in practice, because the backward approach starts with the full set, the subsets to be evaluated in each iteration tend to be larger than those of the forward approach. This limitation makes the execution times a bit worse, specially if the wrapper evaluation is performed with a learning algorithm that does not scale well with the number of features.

Another thing that impacts the performance negatively is the impossibility to adopt effective early stopping mechanisms. This is because it starts by looking at the worst ranked features first, most of these attributes will be eliminated and the relevant features will usually not be explored until the lasts iterations.

6.1.2 Evaluation measure

It is also important to define how we will determine if a subset is significantly worse than another. In this version a five-fold cross-validation will be performed to estimate the classification accuracy for a given set of features. However, finding that the mean of the five-fold cv has lowered does not automatically mean that the feature is relevant. To obtain reliable results, we need to perform some form significance analysis to determine if the result is significantly worse. To do so we have considered several options:

- Using an approach similar to BIRS in which a Student's paired two-tailed t-test is used.
- Using an approach similar to IWSS in which the means of 5 fold-cross validation are used with the added condition of needing k subsets being better. In our case this last condition is changed to k subsets being worse. Following the work done in IWSS, a k of 3 will be used.
- Using the means of the 5 fold-cross validation paired with the condition that the new subset has to be $k\%$ worse to be considered significantly worse. In this case a k of 1% will be used.

After performing several tests and following the the studies conducted by Ruiz, R., Riquelme, J., and Aguilar-Ruiz, J. [10], we conclude that the second works best for our algorithm. A subset will be considered significantly worse if the mean of 5-fold cv is lower and at least 3 folds have lower results.

6.2 BWRR$_{v2}$

We also present a second version, which we will name BWRR$_{v2}$, that has a slight modification that will help speed up computational times in datasets with a large number of irrelevant attributes. The main computational burden in such datasets comes from the fact that BWRR recomputes ReliefF each time it finds a feature to be irrelevant. Although as shown in section 6.5, the time complexity is dominated by the wrapper, performing up to a ReliefF evaluations of cost $O(n^2 \cdot a)$ is still quite penalising.

Moreover, in large features spaces, removing an irrelevant attribute will not have a large effect on the distribution of the nearest neighbours. This means that after deleting an irrelevant feature in one iteration, the ReliefF estimates will most likely remain quite close to the ones we already had. Therefore, we can afford to spend several iterations without recomputing without the wrapper search being significantly affected.

To ease this computational burden and extend the algorithm to datasets with a large number of irrelevant features we decided to change the re-ranking condition. In the new BWRR$_{v2}$ algorithm, we will only recalculate ReliefF estimates when the wrapper search finds a relevant feature. We only recompute in this case because finding a relevant feature might be an indication that the ReliefF ranking has degraded.

Algorithm 8 BWRR$_{v2}$

Input: N instances; A features; J_w wrapper subset evaluator
Output: final feature subset

1: $W = RBA()$
2: order W increasingly
3: $S = \{A_1, A_2, \ldots, A_N\}$
4: $bestScore = J_w(S)$
5: **for** $i = 1$ to A **do**
6: $\quad S_{aux} = S \setminus \{A_{w_i}\}$
7: $\quad score = J_w(S_{aux})$
8: $\quad$ **if** $score \triangleleft bestScore$ **then**
9: $\quad\quad W = RBA()$
10: $\quad$ **else**
11: $\quad\quad bestScore = score$
12: $\quad\quad S = S_{aux}$
13: $\quad$ **end if**
14: **end for**

The main benefit of this new version is that it will reduce the execution times without severely impacting the quality of the resulting set.

6.3 BWRR_{v3}

We also present two variations that will cater towards high-dimensional datasets. In such problems, the biggest performance bottleneck comes from the fact that we are performing a backward search. As we start with a complete set of attributes, in the first wrapper iterations the learning algorithm is being trained with a very large number of features. This process takes time because the computational time of training most learning algorithms scales with the number of attributes.

The key to improving efficiency can be found when analysing in which iterations the regular BWRR_{v1} is most likely to find relevant attributes. We tested with several synthetic problems and we found that it mostly finds the relevant features in the last iterations. This is because as we recompute ReliefF, the new estimates keep being improved, which better guides the search and leaves the relevant attributes the last.

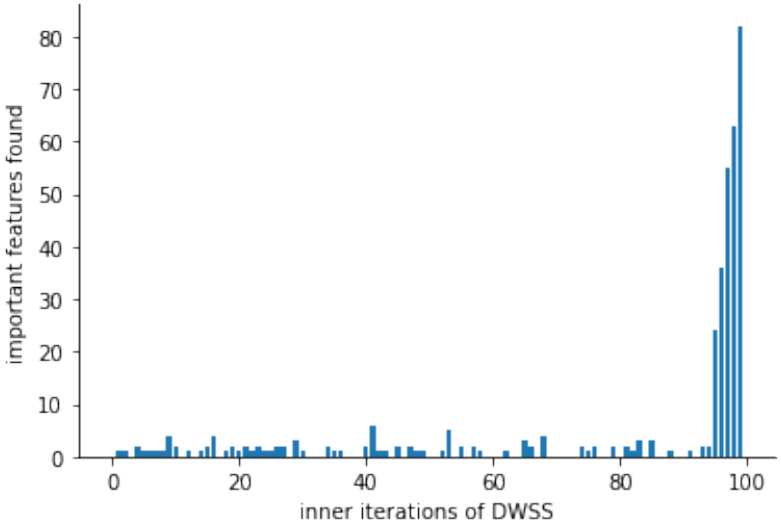

Figure 7: Iterations where the algorithm found relevant features out of 100 different runs. [Compiled by the author]

As the algorithm is not likely to find relevant features in the first iterations, it is reasonable to do an early pruning of features instead of running time consuming iterations that are not likely to find relevant features.

We propose BWRR_{v3}, an algorithm with two stages. In the first stage, a fast filter like the mutual information will be performed, after which the k best features will be selected. In the second stage, we will perform BWRR_{v2} with the pruned subset. The main benefit of removing some features in the early stage is that now in the first iterations of the BWRR_{v2} wrapper, the learning algorithms will be trained with fewer features, thus performing much faster. This new algorithm combines the two types of hybrid algorithm. It is a two stage approach in which in the second stage instead of running a regular wrapper we run a hybrid wrapper guided by filter algorithm.

6.4 BWRR_{v4}

Our last variation, which we will name BWRR_{v4}, will also be focused on high-dimensional datasets. The main novelty introduced in this version is that we will be treating features in blocks of size b. Each iteration, the worse b features will be temporally deleted, thereafter, we will test the accuracy of the new subset. Next, we will perform significance testing which can lead to two cases:

- **The new subset has not significantly lower results:** In this case, we will just remove all b features permanently.
- **The new subset has significantly lower results:** In this case, we can assume that at least one of the b features had to be relevant for the problem. To detect which features are relevant, we will return to the previously used approach of deleting features one by one. By doing so, we will be able to detect which of the b features are valuable and keep them, and which of them were irrelevant and permanently delete them.

Algorithm 9 BWRR_{v4}

Input: N instances; A features; J_w wrapper subset evaluator
Output: final feature subset

$W = RBA()$
order W increasingly
$S = \{A_1, A_2, \ldots, A_N\}$
$bestScore = J_w(S)$
for $i = 1$ to A **do**
  $B = \{$worst b ranked features$\}$
  $S_{aux} = S - B$
  $score = J_w(S_{aux})$
  if $score \triangleleft bestScore$ **then**
    # Some feature was relevant
    for fb in B **do**
      $S_{inner} = S - \backslash\{fb\}$
      $scoreInner = J_w(S_{inner})$
      if not $score \triangleleft bestScore$ **then**
        $bestScore = scoreInner$
        $S = S_{inner}$
      end if
    end for
    $W = RBA()$
  else
    # All features were irrelevant
    $bestScore = score$
    $S = S_{aux}$
  end if
end for

This approach reduces computational time because as seen in fig. 7, early iterations are very likely to not contain relevant features. This means that we can assess if a group of iterations contains only irrelevant features by performing one evaluation instead of b.

6.5 Complexity

In this section we will discuss the complexity of the proposed algorithm, both time and space complexities will be analysed. Before starting, it is important to understand that as all hybrid methods it is composed by a filter and wrapper. We will refer to the complexity of the filter as F and the complexity of the wrapper as W. As nomenclature we will use n as the number rows and a as the number of features.

In our case, the filter is ReliefF which is a RBA with complexity $O(n^2 \cdot a)$. However, we will not define a fixed wrapper learning algorithm. This is common practice in both wrapper and hybrid approaches where the learning algorithm is seen as a black box and its choice is left to the user.

$\mathbf{BWRR_{v1}}$

The main advantage of the proposed hybrid approach is that it performs a linear number of wrapper evaluations with respect to the number of attributes ($O(a \cdot W)$). The number of filter evaluations will depend on the number of selected features. In the worst case, at each iteration, the deleted feature will not be useful and therefore we will perform up to a re-evaluations ($O(a \cdot (n^2 \cdot a))$). After each re-evaluation it is also necessary to order the resulting attribute weights, this can be done in $O(a \log a)$. In total the time complexity related to filter evaluations is $O(a \cdot ((n^2 \cdot a) + a \log a)) = O(a \cdot (n^2 \cdot a))$.

Taking all of this into account we reach the final time complexity of the algorithm:

$$O(a \cdot (W + F)) = O(a \cdot (W + n^2 \cdot a)) = O(a \cdot max(W,\ n^2 \cdot a))$$

Although this is the worst case complexity, for most databases the number of filter reranks will smaller. In fact, as we only rerank when we find an irrelevant feature, the actual number of filter evaluations should be close to the number of irrelevant attributes I.

$$O(a \cdot W + I \cdot F) = O(max(a \cdot W,\ I \cdot n^2 \cdot a)$$

It is also worth noting that for most complex complex learning algorithms $W >> n^2 \cdot a$. This means that in most cases the real time complexity of the algorithm will be $O(aW)$.

The space complexity is mostly defined by the computation of nearest neighbours inside ReliefF. This computation has a spacial cost of $O(n \cdot a)$.

BWRR$_{v2}$

The only meaningful difference that BWRR$_{v2}$ introduces is performing filter reranks only when we find a relevant feature. In this case, although the worst case complexity remains the same, the number of filter evaluations should be closer to the number of relevant attributes R.

$$O(a \cdot W + R \cdot F) = O(max(a \cdot W,\ R \cdot n^2 \cdot a)$$

BWRR$_{v3}$

BWRR$_{v3}$ is a two stage algorithm, therefore to analyse the complexity we will analyse stage 1 (S_1) and stage 2 (S_2) separately.

In the first stage, we will execute the mutual information filter. If we have a continuous variable, the mass probability can not be defined, and therefore the mutual information weights are computed using several estimators [26].

- **3KL-Estimation:** The computational complexity is $O(n \cdot logn)$.
- **KSG-Estimation:** The computational complexity is $O(n \cdot logn)$.

Only the first k ranked features will move on to the wrapper space. After that, we just execute BWRR$_{v2}$, leaving us with a complexity S_2 of:

$$O(k \cdot (W + F)) = O(k \cdot (W + n^2 \cdot a)) = O(k \cdot max(W,\ n^2 \cdot a))$$

The final complexity of the algorithm is just $O(S_1 + S_2)$

BWRR$_{v4}$

In BWRR$_{v4}$ we test subsets by removing groups of b features at a time. If the new subset decreases the scores significantly we return to removing one feature at a time to determine which features were in fact useful.

In the worst case, where after removing a group of b features the new subset decreases, the number of wrapper evaluations can increase slightly. The new complexity moves to:

$$O((a + a/b) \cdot W + a \cdot F) = O(max((a + a/b) \cdot W,\ n^2 \cdot a^2)$$

However, we argue that on average the number of wrapper evaluations should decrease because we can assess if a group of iterations contains only irrelevant features by performing one evaluation instead of b.

7 Experimental set-up

In this section we will explain the experimentation set-up. We will describe the selected datasets, the evaluation measures and the relevant factors.

7.1 Experimental phases

The experimentation part will be divided into 2 stages:

Phase 1: Testing on synthetic datasets

First of all to get a grasp of how the proposed algorithms are performing we will test them in synthetic datasets. In these, we will be able to control key factors that will help us understand how the algorithm is working. Each dataset will be used to analyse the performance of the algorithm at solving different types of problems.

This phase will be used to test the different variations of the proposed algorithm. However, we will not be doing direct comparisons with other state-of-the-art hybrid algorithms. Instead, we will be comparing the performance of our algorithm against a pure filter and a pure wrapper approach.

Phase 2: Testing on real world data

In the second phase we will be testing the algorithm on real-world datasets. This step is necessary because it is always important to see how the algorithm fares against other state-of-the-art algorithms in a real-world environment.

This phase can also be subdivided depending on the type of dataset we will use. We will be working on regular datasets and high dimensional datasets. In the experimentation phase, we will mostly focus on high-dimensional datasets because it is where hybrid solutions are most effective.

7.2 Algorithms

In the experimentation part, we will use the package ReBATE [27], which implements many RBA algorithms. We will also be using some base utilities from the scikit package [28]. I have implemented the majority of algorithms in Python based on the pseudo-codes presented in the original papers. They have also been adapted to scikit, allowing users to execute regular scikit feature selection pipelines in combination with the implemented algorithms.

The majority of algorithms are parameter-free. We will fix ReliefF number of estimates to 5 as recommended by Kononenko and we will use the Relieved version.

The only exception will be the iSFSM algorithm in which for computational reasons I will fix the SubsetSize parameter to 100.

The two initial versions of our algorithm are parameter-free. For $BWRR_{v3}$ we will be setting the cuttof parameter to 500, this means that at most 500 features will move to the second stage. For $BWRR_{v4}$ we will also fix the blockSize to 50.

7.3 Real-world experimentation setup

7.3.1 Datasets

To perform experimentation we will need some data to execute the feature selection methods. We will select our datasets regardless if it is a regression or classification tasks as the proposed algorithm can deal with both. However, we will impose the following characteristics to our datasets:

- **Number of features:** There need to be enough features to perform feature selection. We will be using medium sized datasets between $10 - 100$ features and larger datasets with 500+ features.
- **No missing data:** Although RBAs can work with missing data the learning algorithms we will use can't. To avoid having to do data imputation and adding extra uncertainty to our dataset we will only select complete datasets.

We obtained datasets from UC Irvine Machine Learning Repository [29] and Arizona State University feature selection datasets [30].

Regular datasets

For each dataset, we will show the instances, features, classes, types and topics. Moreover, we will also include the source where we found it.

Dataset	Instances	Features	Classes	Type	Topic	Source
breast-cancer	569	32	2	Continuous, binary	Biological	1
ionosphere	351	34	2	Continuous, binary	Physical	1
sonar	208	60	2	Continuous, binary	Physical	1
parkinsons	197	23	2	Continuous, binary	Biological	1
wine	178	13	3	Continuous, multi-class	Physical	1

Table 1: Medium-size datasets datasets.

High-dimensional

In the last phase, we will need high-dimensional datasets to compare the hybrid feature selection algorithms. There are several areas of research that produce datasets with such large feature spaces. The main candidates are text datasets, image datasets and microarray datasets. We finally decided to use the latter because they are the most used in the relevant hybrid FS literature.

We will not go deeper into explaining each dataset separately. In summary, all of them are microarray datasets in which each feature represents a determined gene, each instance is then a set of genomics information of a patient. Feature selection is specially useful in this type of datasets because it tries to find which genes are responsible for the development of certain illnesses.

On a more technical note, these problems are hard because the data is sparse, being the relevant subset orders of magnitude smaller than the irrelevant. Moreover, the number of instances is also much smaller than the number of features. This can be a problem because it is easy to overfit.

Dataset	Instances	Features	Classes	Type	Topic	Source
colon	62	2000	2	Discrete, binary	Biological	2
GLIOMA	50	4434	4	Continuous, multi-class	Biological	2
leukemia	72	7070	2	Discrete, binary	Biological	2
lung_discrete	73	325	7	Discrete, multi-class	Biological	2
Prostate_GE	102	5966	2	Continuous, Binary	Biological	2
TOX_171	171	5748	4	Continuous, multi-class	Biological	2

Table 2: High-dimensional datasets.

Additionally we will run the algorithms on NIPS 2003 Feature Selection Challenge datasets [31]. These are synthetic datasets that were specially designed and are normally used as feature selection benchmarks.

7.3.2 Evaluation measure

In real-world datasets, we do not know for a fact which features are relevant and irrelevant. Therefore, to evaluate the selected subset we will use the accuracy if it is

a classification problem or the mean squared error otherwise.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{15}$$

It is worth noting that several other measurements like AUC, precision or f1-score could be used. We opted for the traditional accuracy metric because is by far the most used in the literature.

We will also consider important the size of the selected subset. In cases where two FS algorithms end up with similar scores, the one that resulted in the smaller set is normally preferred.

7.4 Synthetic experimentation setup

7.4.1 Datasets

Dataset 1 - Parity

The first synthetic problem that we will analyse is the classic parity problem. All the variables of this dataset, including the target, will be binary $\{0, 1\}$. The target value is a function of whether or not the number of variables that contain a 1 is even. More formally we can describe it has:

$$target(X) = \begin{cases} 0 & if \quad \sum_{x_i \in X} x_i \text{ even} \\ 1 & if \quad \sum_{x_i \in X} x_i \text{ odd} \end{cases} \tag{16}$$

Dataset 2 - Majority

Majority is a synthetic dataset that contains only binary variables $\{0, 1\}$. If there is a majority of ones among the relevant variables, the class will be a 1. Otherwise, it will be a 0. More formally we can describe it has:

$$target(X) = \begin{cases} 0 & if \quad \text{majority of zeros} \\ 1 & if \quad \text{majority of ones} \end{cases} \tag{17}$$

In general this classification task is a bit easier than the Parity problem.

Dataset 3 - Linear regression

In this particular setting, we want to analyse how the algorithm works in a linear regression problem. The dataset is comprised of features with continuous values

between 0 and 1. $[0, 1]$. The target value is assigned by assigning a coefficient ω_i to each variable and adding all their values.

$$target(X) = \sum_{x_i \in X} \omega_i \cdot x_i$$

All left to do is define the coefficients ω_i of each feature. Note that the larger $abs(\omega_i)$ is the more the feature will impact the target value. Therefore those features with higher $abs(\omega)$ will be considered more relevant.

All that it is left to do is to decide how we will assign the values of these coefficients. There are several linear solutions available, we will choose $\omega = [1, -1, 2, -2 \dots]$. Expressing the target in a more formally way we get the following formula:

$$target(X) = \sum_{i=1}^{|X|} \left\lceil \frac{i}{2} \right\rceil \cdot (-1)^{i+1} \cdot x_i \tag{18}$$

7.4.2 Dataset factors

One of the advantages of synthetic datasets is that we can modify some key parameters to see how the algorithms perform in several controlled situations.

- **Number of relevant (r):** This is the number of variables that are relevant for the target. For each dataset we will try with values: $[4, 12, 32]$
- **Number of irrelevant:** (i) This is the number of variables that were not used to compute the target. For each dataset, we will try with values: $[0.5, 2.0, 4.0] * r$. In case the resulting number is is decimal we will take the ceiling of it.
- **Number of redundant:** (r') Variables that are in the same equivalence class as one of the relevant variables. Therefore, they do not give us extra information. For each dataset, we will try with values: $[0.0, 0.25, 0.5] * r$.
- **Number of instances:** We will consider several different values for the number of instances of the training dataset: $[500, 1000, 5000]$.
- **Noise:** We will only study the effects of adding noise in the two regression datasets. We will be adding $[0\%, 2.5\%, 5.0\%]$ of noise.
- **Number of features:** The number of features is not really a parameter of the dataset, just the sum of r, i and r'.

For all the relevant factors we decided to only choose 3 levels because is the maximum we can afford with our current computation capabilities. To design the experiments we chose a full factorial design. For the parity and majority datasets, we will have a 3^4 full factorial and for the regression problems a 3^5 design. For each combination, we will be executing 4 feature selection algorithms. In total we will need $4 * (2 * 3^4 + 3^5)$ different executions.

7.4.3 Dataset generation

To generate each of the datasets we will implement a function of the form $f(instances, r, i, r', noise) -> dataset$. This function will take as input the instances, noise and number of relevant, irrelevant and redundant features and output a dataset ready to be used as a benchmark. This function will also contain other subroutines that will be needed to prepare the final dataset:

- **Value generator:** This will be a function that takes either a set of values or a maximum and minimum in case of continuous values.
- **Target calculator:** To compute the target we will use eq 16 for Parity, eq 17 for Majority and eq 18 for Linear. After that, we will add noise to it if needed.
- **Redundant feature generator:** To add redundant features we first need to know which are the relevant ones. After that, we will generate redundant features out of the relevant ones.
- **Label generator:** To be able to compute the *score* it is a must to be able to identify with features are relevant, irrelevant and redundant. Moreover there is also the need to know how to relate redundant variables to their correspondent relevant variables. This will be needed to establish equivalence classes. To correctly identify the features we will use the following nomenclature.
  - ith relevant feature: We will name it rel-i
  - ith irrelevant feature: We will name it irr-i
  - ith redundant feature: We will name it red-i_j being j a sort of foreign key that links red-i with rel-j.
- **Feature shuffler:** For extra safety, we will shuffle the columns of the dataset.

Combining this together we can establish a baseline for the implementation of dataset generators. Following all these steps we implement this pseudocode:

Algorithm 10 Synthetic dataset generator

Input: r relevant, i irrelevant, r' redundant, I instances, n noise
Output: final synthetic dataset D

```
for i = 1 to I do
    for k to r + i do
        D[i][k] = Generate random value.
    end for
end for
Compute y by using the relevant variables.
Add noise to y
Generate r' redundant features from the relevant variables.
D = Feature shuffle(D)
return D
```

Note that of all these steps the only ones that are dataset specific are the value generation (line 3) and the target calculations (line 8). This means that to generate each dataset we will only need to provide a value generator that will depend on the possible set of values for each problem and a way to calculate the target. All the other steps are the same for all datasets and thus can be left unchanged.

7.4.4 Evaluation measure

The biggest advantage of analysing feature selection algorithms in synthetic datasets is that we know which features are relevant X_R, irrelevant X_I and redundant $X_{R'}$. This means that from the total set of features X, we can partition it into $X = X_R \cup X_I \cup X_{R'}$. This extra knowledge allows us to better measure the quality of a resulting subset S. In synthetic datasets we will use as a metric the *score* mentioned by L. C. Molina, L. Belanche and A. Nebot [32]. The informal ideal behind the construction of this metric is to penalize the following cases:

1. There are relevant features lacking in A (the solution is incomplete).
2. There are more than enough relevant features in A (the solution is redundant).
3. There are some irrelevant features in A (the solution is incorrect).

Moreover using $\alpha = [\alpha_R, \alpha_I, \alpha_{R'}]$; $\alpha_T \geq 0$, $\sum \alpha_T = 1$; we can choose how much each situation is penalized.

To formally define the score it separates features into equivalence classes, where in each class all elements are redundant to each other. Then it defines the equivalence relation between features $x_i \sim x_j \iff x_i$ and x_j represent the same information. $A^{\sim}$ is then defined as the quotient set defined by $\sim$.

$$I = 1 - \frac{|A_I|}{|X_I|} \tag{19}$$

$$R = \frac{|A_R^{\sim}|}{|X_R|}, \quad \text{where } A_R^{\sim} = (A_R \cup X_R)^{\sim} \tag{20}$$

The redundancy term is more complex. We introduce the following notation: $A_R^{\sim} \uplus X_R^{\sim} = \{[x_i] \in X_R^{\sim} | \exists [x_j] \in A_R^{\sim} : [x_j] \supseteq [x_i]\}$. And also a function that receives as input a quotient set $F(Q) = \sum_{|x| \in Q} (|x| - 1)$

$$R' = \begin{cases} 0 & F(A_R^{\sim} \uplus X_R^{\sim}) \\ \frac{1}{|X_{R'}|} \left(1 - \frac{F(A_R^{\sim})}{F(A_R^{\sim} \uplus X_R^{\sim})}\right) & \text{otherwise} \end{cases} \tag{21}$$

Finally the final *score* of a selected subset A can be defined by the relevance, irrelevance and redundancy terms multiplied by their respective weights.

$$S(A) = \alpha_R R + \alpha_I I + \alpha_{R'} R' \tag{22}$$

The last thing to consider are the values of the weights α. This is important because they determine the importance of the relevance, irrelevance and redundancy terms. The simplest way to set the values would be to give each term the same relevance. However, this is often not very practical because in practice we tend to value some more than others. For example:

- It is considered better to add an irrelevant variable than to miss a relevant one.

$$\frac{\alpha_R}{|X_R|} > \frac{\alpha_I}{|X_I|}$$

- We prefer selecting a redundant variable over an irrelevant one.

$$\frac{\alpha_I}{|X_I|} > \frac{\alpha_{R'}}{|X_{R'}|}$$

It is then advisable to choose values of α that reflect what conditions we value more in the final subset. The original paper analyses some of these conditions and reaches an equation that the weights should follow:

$$\beta_R \alpha_R = \alpha_I \qquad\qquad \beta_I \alpha_I = \beta'_R$$

Finally, they also propose values for β_R, β_I which we will be adopting in this project: $\beta_R = \epsilon/2$, $\beta_I = 2\epsilon/3$. We are now left with only one parameter ϵ, during the experimentation we will be using $\epsilon = 1$ which means that α_R will be counting twice as much as α_I

8 Experiments discussion

8.1 Synthetic experiment

As discussed in the previous section in the synthetic experimentation we will be interested in determining how our algorithm behaves when we vary different relevant factors. To do so, we have run a 3 level full factorial design with all synthetic datasets.

To study how each factor affects the score, accuracy and time outputs we decided to perform ANOVA. Analysis of variance (ANOVA) is a statistical test used to determine whether two or more population means are the same. It is based on the law of total variance, which tries to partition the variance in a particular variable into different sources of variation. It is particularly powerful in factorial designs because for example with 3 factors x, y, z it is capable of modeling main effects (x, y, z) and interactions (xy, xz, yz, xyz). We will now present the most relevant findings of the ANOVA analysis. Nevertheless, the complete tables can be seen in section 12.2.

Accuracy

- The number of instances and the irrelevant features are significant factors with regards to the prediction score in the Majority and Linear problems.
- The number of relevant features seems to be the most relevant factor, being a key factor in all synthetic datasets studied for all algorithms tested. This is specially true for the Parity problem, whose difficulty seems to be only gated by the number of relevant factors.
- The number of redundant variables and the amount of noise do not appear to be statistically significant in any of the problems studied.
- All factors seem to be affecting the different algorithms in a similar manner. We have not yet found a factor that is only significant for a subset of algorithms.

Score metric

- With regards to score, the number of relevant features again is the most significant factor, followed by the number of instances which is important in 2 of the datasets studied.
- The number of redundant variables does not appear to be a relevant factor in any of the problems studied.
- An interesting fact is that now irrelevance is only considered a significant factor for the BWRR_{v1} and BWRR_{v2} algorithms. This means, that the number of irrelevant features must affect the quality of subsets in some way.

- The biggest difference is that noise seems to be quite significant to predict the score metric. This indicates that although the noise is not decreasing the performance in a significant way, the quality of subsets estimated with the score metric is in fact getting worse.

We will also present the interaction plots that better show the effects that each factor have on the algorithms. The complete set of interaction plots can be seen in section 12.1.

Accuracy

Figure 8 and figure 9 show how both the number of relevant and irrelevant features affect the prediction results. In general, our algorithms $BWRR_{v1}$ and $BWRR_{v2}$ perform slightly worse than the regular wrapper, but better than the pure filter. In the parity dataset both SBS, $BWRR_{v1}$ and $BWRR_{v2}$ obtain similar results, outperforming the baseline score and ReliefF filter. In the other problems, the pure wrapper approach seems to be the most effective, $BWRR_{v1}$ and $BWRR_{v2}$ provide a middle ground and ReliefF performs slightly worse than our hybrid algorithms.

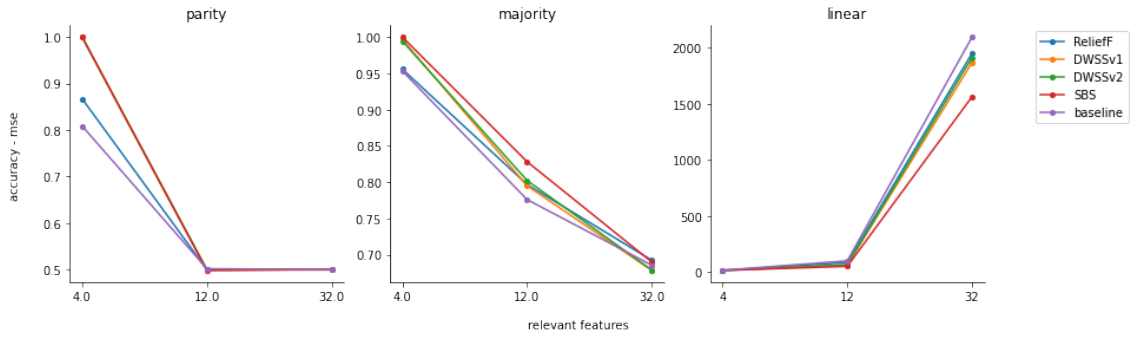

Figure 8: Relevance and accuracy interaction plot

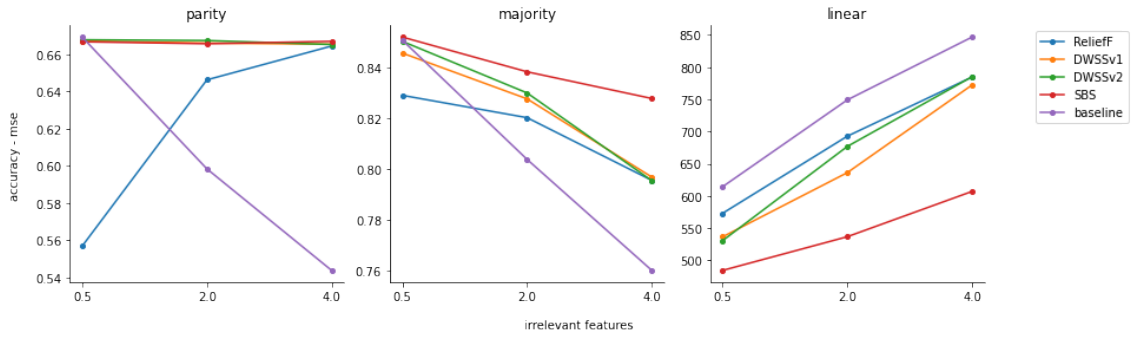

Figure 9: Irrelevance and accuracy interaction plot

Score

Figure 11 and figure 12 show how both the number of instances, relevant and irrelevant features affect the score metric. It is quite clear that SBS performs the best, followed by $BWRR_{v1}$ and $BWRR_{v2}$ and finally ReliefF. An interesting fact is that for all algorithms scores improve when the number of training instances increases. The opposite tends to happen for the number of relevant and irrelevant features, as they increase the score metric goes down. This is specially true for $BWRR_{v1}$ and $BWRR_{v2}$ in the majority problem, in this dataset, it appears that our algorithm's final subsets degrade in quality when the number of irrelevant features increases.

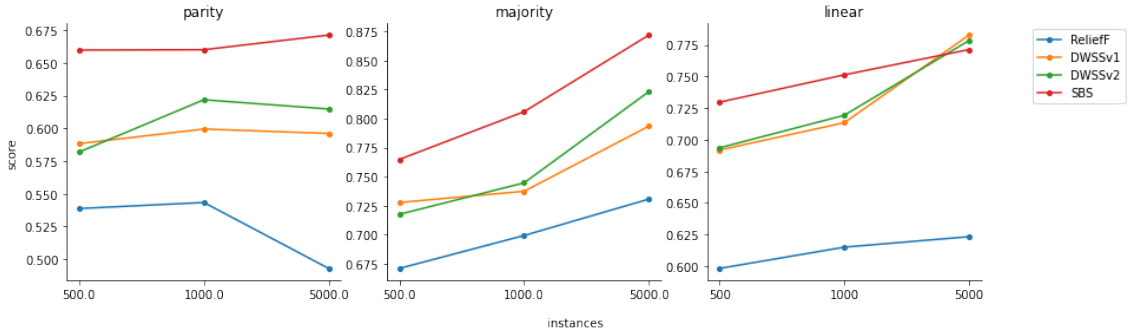

Figure 10: Instances and score interaction plot

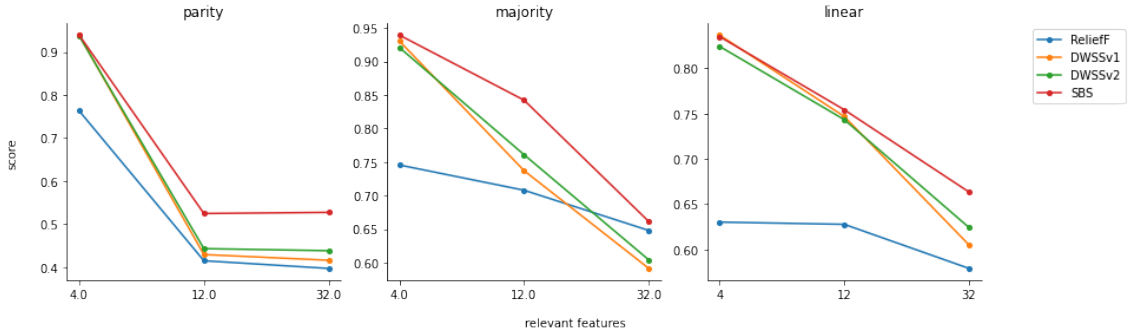

Figure 11: Relevance and score interaction plot

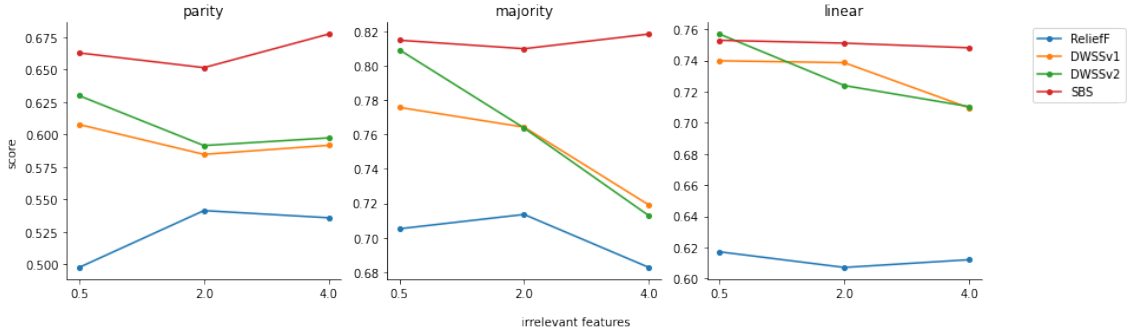

Figure 12: Irrelevance and score interaction plot

Time and wrapper evaluations

Figures 13 and figure 14 exemplify how many computational resources are needed to execute each algorithm as the number of features increases. As expected, the pure wrapper approach performs quite poorly, needing more time and wrapper evaluations than others. ReliefF is by far the fastest, as it requires zero wrapper evaluations. BWRR_{v1} and BWRR_{v2} provide a middle ground, being a bit more expensive than the pure filter approach. It is also worth noting that BWRR_{v2} is faster than BWRR_{v1}, specially as the number of irrelevant features increases. This improvement is likely caused by a reduction in the number of ReliefF reranks inside BWRR_{v2}.

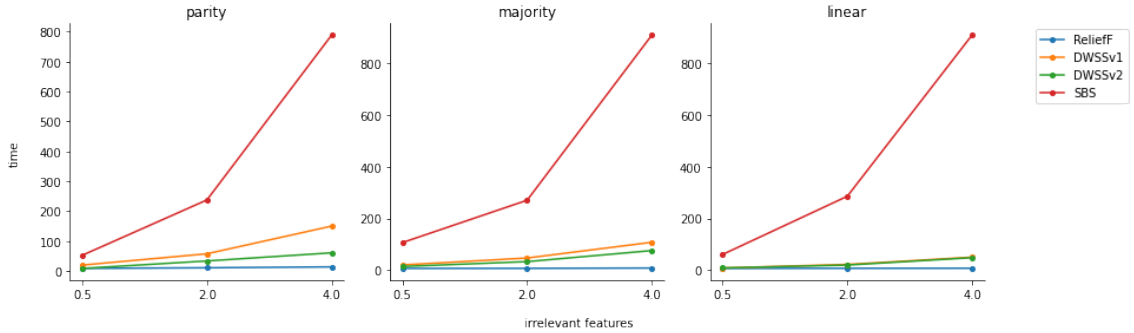

Figure 13: Irrelevance and time interaction plot

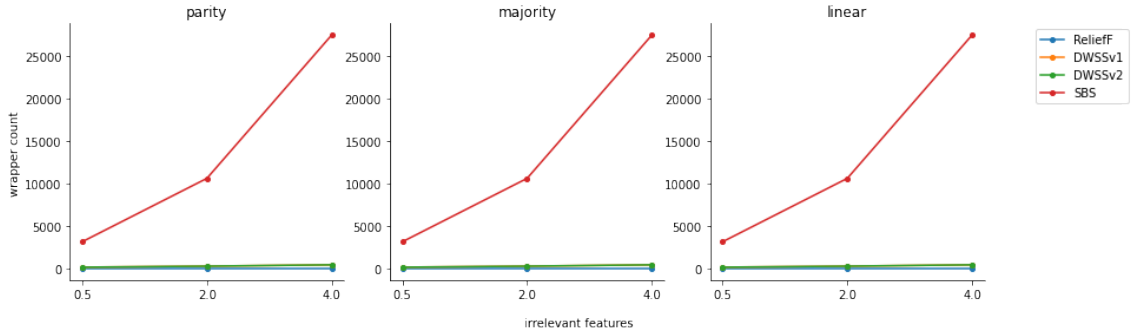

Figure 14: Irrelevance and wrapper count interaction plot

8.2 Real world experiment

8.2.1 Experiments on regular datasets

In this section we present the results obtained by performing a 10x5 cross validation on regular datasets. Table 5 shows the accuracy obtained and table 6 the execution times.

To test the statistical significance, we compared each result against BWRR_{v1} using a two-tailed Student t-test with $\alpha = 0.05$. These comparisons are shown in the following tables, where a (+) represents that the algorithm performs better than the competition, a (-) signifies a worse performance and (=) is used to show that the results are statistically similar.

datasets	Algorithm							
	Relief	**BIRS**	**IWSS**	**IWSS$_s$**	**iSFSM**	**IGIS**	**BWRR$_{\text{v1}}$**	**BWRR$_{\text{v2}}$**
breast-cancer	+	=	=	=	=	+	=	=
ionosphere	=	=	=	-	=	=	=	=
iris	=	+	=	=	=	=	=	+
parkinsons	-	=	-	=	-	=	=	=
sonar	-	+	=	+	-	+	=	-
wine	+	=	+	+	=	+	=	+

Table 3: Statistical comparison using two-tailed Student t-test of BWRR_{v1} against other state of the art algorithms in regular datasets

We do the same but this time comparing BWRR_{v2} versus the rest.

datasets	Algorithm							
	Relief	**BIRS**	**IWSS**	**IWSS$_s$**	**iSFSM**	**IGIS**	**BWRR$_{\text{v1}}$**	**BWRR$_{\text{v2}}$**
breast-cancer	+	=	=	=	-	+	=	=
ionosphere	=	=	-	-	=	-	=	=
iris	-	=	-	-	-	-	-	=
parkinsons	-	-	-	-	-	=	=	=
sonar	-	+	+	+	-	+	+	=
wine	=	-	-	-	-	-	-	=

Table 4: Statistical comparison using two-tailed Student t-test of BWRR_{v2} against other state of the art algorithms regular datasets

datasets	Accuracy						
	BIRS	**IWSS**	**IWSS$_s$**	**iSFSM**	**IGIS**	**BWRR$_{v1}$**	**BWRR$_{v2}$**
breast-cancer	0.917 ± 0.048	0.916 ± 0.038	0.912 ± 0.044	$\mathbf{0.918 \pm 0.037}$	0.882 ± 0.044	0.911 ± 0.043	0.903 ± 0.04
ionosphere	0.863 ± 0.062	0.884 ± 0.047	$\mathbf{0.885 \pm 0.044}$	0.874 ± 0.053	$\mathbf{0.885 \pm 0.053}$	0.868 ± 0.052	0.854 ± 0.067
iris	0.9 ± 0.095	0.947 ± 0.057	0.951 ± 0.053	0.947 ± 0.063	$\mathbf{0.96 \pm 0.055}$	0.955 ± 0.052	0.905 ± 0.061
parkinsons	0.808 ± 0.084	0.833 ± 0.086	0.812 ± 0.091	$\mathbf{0.85 \pm 0.083}$	0.786 ± 0.084	0.799 ± 0.096	0.773 ± 0.112
sonar	0.621 ± 0.103	0.71 ± 0.108	0.68 ± 0.114	$\mathbf{0.847 \pm 0.076}$	0.64 ± 0.089	0.732 ± 0.116	0.78 ± 0.115
wine	$\mathbf{0.931 \pm 0.069}$	0.831 ± 0.13	0.831 ± 0.125	0.91 ± 0.071	0.788 ± 0.138	0.92 ± 0.075	0.744 ± 0.111

Table 5: Accuracy on regular datasets.

datasets	Accuracy						
	BIRS	**IWSS**	**IWSS$_s$**	**iSFSM**	**IGIS**	**BWRR$_{v1}$**	**BWRR$_{v2}$**
breast-cancer	0.449	0.443	0.405	6.248	0.691	8.437	1.882
ionosphere	0.447	0.425	0.26	3.1	1.321	5.707	2.548
iris	0.044	0.042	0.037	0.083	0.097	0.244	0.179
parkinsons	0.231	0.242	0.136	2.015	0.229	2.266	0.387
sonar	0.602	0.585	0.389	5.603	0.761	5.053	1.719
wine	0.14	0.129	0.09	0.65	0.19	1.424	0.44

Table 6: Time on regular datasets.

8.2.2 Experiments on high dimensional datasets

Finally, we will analyse the performance of several hybrid algorithms in several high-dimensional datasets. This is the most important part of the experimentation because it is on these types of datasets where hybrid algorithms are used the most. The following results were obtained by performing a 10-fold cross validation.

To test the statistical significance, we now compared each result against BWRR_{v2} using a two-tailed Student t-test with $\alpha = 0.05$.

datasets	Algorithm						
	BIRS	**IWSS**	**IWSS$_s$**	**iSFSM**	**BWRR$_{\text{v2}}$**	**BWRR$_{\text{v3}}$**	**BWRR$_{\text{v4}}$**
colon	=	=	=	=	=	=	=
GLIOMA	+	+	=	+	=	=	=
leukemia	=	=	=	-	=	=	=
lung_discrete	=	+	=	=	=	=	=
Prostate_GE	=	=	=	=	=	=	=
TOX	+	=	=	+	=	=	=
madelon	=	-	-	-	=	=	+
arcene	=	=	=	=	=	=	-
gisette	-	-	-	=	=	-	-
Isolet	=	-	-	-	=	=	=

Table 7: Statistical comparison using two-tailed Student t-test of BWRR_{v2} against other state of the art algorithms in high-dimensional datasets

We do the same but this time comparing BWRR_{v3} versus the rest.

datasets	Algorithm						
	BIRS	**IWSS**	**IWSS$_s$**	**iSFSM**	**BWRR$_{\text{v2}}$**	**BWRR$_{\text{v3}}$**	**BWRR$_{\text{v4}}$**
colon	=	+	+	=	=	=	=
GLIOMA	=	=	=	+	=	=	=
leukemia	=	=	=	-	=	=	=
lung_discrete	+	+	+	=	=	=	=
Prostate_GE	=	=	=	=	=	=	=
TOX	+	=	=	+	=	=	=
madelon	=	-	-	-	=	=	+
arcene	=	=	=	=	=	=	=
gisette	-	-	-	+	+	=	-
Isolet	=	-	-	-	=	=	=

Table 8: Statistical comparison using two-tailed Student t-test of BWRR_{v3} against other state of the art algorithms high-dimensional datasets

Finally, we compare $BWRR_{v4}$ to the rest:

datasets	Algorithm						
	BIRS	**IWSS**	**$IWSS_s$**	**iSFSM**	**$BWRR_{v2}$**	**$BWRR_{v3}$**	**$BWRR_{v4}$**
colon	=	=	=	=	=	=	=
GLIOMA	+	+	=	+	=	=	=
leukemia	=	=	=	=	=	=	=
lung_discrete	=	=	=	=	=	=	=
Prostate_GE	=	=	=	=	=	=	=
TOX	+	=	=	+	=	=	=
madelon	-	-	-	-	-	-	=
arcene	+	+	+	+	+	=	=
gisette	+	=	-	+	+	+	=
Isolet	=	-	-	-	=	=	=

Table 9: Statistical comparison using two-tailed Student t-test of $BWRR_{v4}$ against other state of the art algorithms high-dimensional datasets

The results of the real-world experimentation in high dimensional data suggest that our algorithms can compete against other relevant hybrid algorithms. In fact, in several datasets like GLIOMA, TOX or lung_discrete they tend to outperform competition. Amongst our proposed algorithms $BWRR_{v4}$ seems to perform slighly better, having better results in bigger datasets like gisette or arcene. In general, $IWSS_s$ seems to be the best performing algorithm, followed closely by $BWRR_{v4}$. Overall, although some algorithms perform better on average, all of them still have some dataset where they perform the best.

If we compare the execution times shown in table 6 we can see that in general our proposed algorithms are a slower than the rest. This is probably caused by the more complex filter used and the backward search. $BWRR_{v2}$ is in average much slower than others. The improvements introduced in $BWRR_{v3}$ and $BWRR_{v4}$ seem to be effective. $BWRR_{v3}$ is in average as fast as others while the strategy proposed in $BWRR_{v4}$ seems to be hit or miss, in some datasets it is the fastest while in others its times resemble that of $BWRR_{v2}$.

datasets	Accuracy						
	BIRS	**IWSS**	**IWSS$_s$**	**iSFSM**	**BWRR$_{v2}$**	**BWRR$_{v3}$**	**BWRR$_{v4}$**
colon	0.743 ± 0.13	0.726 ± 0.072	0.726 ± 0.072	0.743 ± 0.124	0.774 ± 0.109	$\mathbf{0.826 \pm 0.105}$	0.721 ± 0.197
GLIOMA	0.6 ± 0.179	0.62 ± 0.108	0.7 ± 0.257	0.56 ± 0.233	$\mathbf{0.78 \pm 0.209}$	0.72 ± 0.183	$\mathbf{0.78 \pm 0.189}$
leukemia	0.93 ± 0.095	0.957 ± 0.091	0.957 ± 0.091	$\mathbf{0.971 \pm 0.057}$	0.902 ± 0.091	0.848 ± 0.205	0.902 ± 0.128
lung_discrete	0.698 ± 0.123	0.695 ± 0.108	0.695 ± 0.122	0.716 ± 0.133	0.793 ± 0.129	$\mathbf{0.821 \pm 0.139}$	0.779 ± 0.112
Prostate_GE	0.872 ± 0.158	$\mathbf{0.894 \pm 0.12}$	$\mathbf{0.894 \pm 0.12}$	0.873 ± 0.129	0.875 ± 0.102	0.834 ± 0.108	0.851 ± 0.136
TOX	0.544 ± 0.063	0.725 ± 0.09	0.72 ± 0.093	0.631 ± 0.118	0.748 ± 0.119	0.773 ± 0.139	0.742 ± 0.093
madelon	0.828 ± 0.112	$\mathbf{0.87 \pm 0.022}$	0.846 ± 0.043	0.867 ± 0.02	0.799 ± 0.045	0.786 ± 0.025	0.756 ± 0.039
arcene	0.77 ± 0.078	0.805 ± 0.085	0.795 ± 0.072	0.75 ± 0.095	0.78 ± 0.103	0.815 ± 0.112	$\mathbf{0.875 \pm 0.056}$
gisette	0.95 ± 0.007	0.966 ± 0.005	$\mathbf{0.967 \pm 0.005}$	0.923 ± 0.005	0.906 ± 0.035	0.935 ± 0.02	0.962 ± 0.006
Isolet	0.613 ± 0.075	$\mathbf{0.925 \pm 0.013}$	0.922 ± 0.014	0.774 ± 0.034	0.615 ± 0.111	0.625 ± 0.115	0.574 ± 0.085

Table 10: Accuracy on high-dimensional datasets.

datasets	Accuracy						
	BIRS	**IWSS**	**IWSS$_s$**	**iSFSM**	**BWRR$_{v2}$**	**BWRR$_{v3}$**	**BWRR$_{v4}$**
colon	15.558	15.298	7.907	5.866	17.553	13.306	3.083
GLIOMA	46.05	41.632	18.573	12.98	361.662	33.335	52.407
leukemia	64.216	59.542	19.803	17.375	97.448	18.81	17.476
lung_discrete	3.327	3.153	2.078	4.163	18.185	21.911	3.964
Prostate_GE	68.827	51.362	26.666	48.108	1228.576	31.509	242.622
TOX	123.686	124.028	56.995	43.009	4904.076	74.748	948.831
madelon	26.465	31.948	27.449	41.111	206.85	211.505	60.051
arcene	96.973	89.781	44.156	23.583	1359.733	31.947	807.115
gisette	1899.88	3223.75	578.598	330.56	19092.705	598.262	16041.196
Isolet	34.099	83.797	53.524	50.087	1943.175	295.498	18.798

Table 11: Times on high-dimensional datasets.

9 Conclusion

This section will work as a compilation of conclusions obtained in the different experiments. Moreover, we will derive general conclusions that can only be obtained when looking at all the experiments together.

In the synthetic experimentation, we tried to understand how the pure filter, wrapper and hybrid approaches performed under varying factors. The conclusions we obtained were that relevance and number of instances greatly affected the outcome of accuracy and score metric. Additionally, in the interaction plots, we could appreciate tendencies as each factor varied. The scores increased as the number of instances or redundancy increased, on the other hand as the number of relevant and irrelevant features increases the results degrade. In general, the noise was statistically insignificant.

The minimum goal to consider the project a success was to introduce new hybrid algorithms that provided a trade-off between speed and predictive scores. In the synthetic phase, we tested two of our algorithms against the pure filter and pure wrapper approaches. Results show that we accomplished our goal, BWRR_{v1} and BWRR_{v2} were both faster than wrapper and resulted in better scores than the filter. The goal of BWRR_{v2} was to improve performance in datasets with high number of irrelevant attributes while maintaining good results when compared to BWRR_{v1}. In general, we succeeded, as BWRR_{v2} was faster when the number of irrelevancy increased while maintaining comparable results.

In the real-world experiments, we explored how different versions of our algorithm compared against other state-of-the-art hybrid approaches. We will mostly focus on the results obtained in the high-dimensional settings as it is were such algorithms see more use. The results show that our algorithms outperform others in some datasets such as colon, TOX or GLIOMA while being worse in others. Moreover, we tested the statistical significance of these results and showed that the proposed algorithms do in fact improve significantly in several datasets.

When comparing BWRR_{v2}, BWRR_{v3}, BWRR_{v4} against each other we can see that BWRR_{v2} is not suited for these high dimensional datasets, being its execution time is orders of magnitude larger than the other algorithms. BWRR_{v3} and BWRR_{v4} performed up to par or better than BWRR_{v2} in most high dimensional problems while reducing drastically the execution time and number of wrapper evaluations. We would recommend using BWRR_{v3} and BWRR_{v4} to perform feature selection in high dimensional datasets, their execution times are only slightly higher than the other hybrid algorithms. However, the backward approach combined with the use of a more complex filter like ReliefF has proven to produce improvement in several of the datasets tested.

9.1 Future work

We achieved all goals we were set out to accomplish with this bachelor's thesis. Nevertheless, during the course of the project, we encountered several interesting ideas worth researching. Due to time limitations we could not continuously expand our project, that is why we left the ideas that deviated more from the original plan as potential future work.

- **Additional versions of the algorithm:** Several interesting ideas for new versions had to be left out. The most remarkable one was a version that was a three stage algorithm. We did not implement this algorithm because it required the use of a fixed learning algorithm. The problem is that this would then made it hard for us to compare against other hybrid approaches.
- **Extend the algorithm to missing data:** The algorithm we proposed does not work when there is missing data. As we used ReliefF which can work with missings, this limitation only comes from the wrapper algorithm we used.
- **Experiment with different learning algorithms:** For our hybrid and wrapper algorithms we only used a k nearest neighbour as our black box subset evaluator. It would be interesting to see how the performance changes when using other learning algorithms.
- **Experiment with different RBAs:** In our hybrid algorithm, we used ReliefF in classification problems and RReliefF in regression ones. However, the RBA family is much larger and many other choices could be possible. An interesting extension could focus on how a more complex RBA like Simba changes the results.
- **Explore other hybrid algorithms:** We mostly focused on algorithms that used a sequential wrapper search. Nevertheless, there are many hybrid algorithms that use other heuristics to guide the wrapper.
- **Improve performance:** We programmed the algorithms in an efficient manner. However, recent studies showed that hybrid algorithms could be accelerated by embedding wrappers like k nearest neighbour [33]. We leave the possible inclusion of this optimization plus some possible parallelization as future work.

10 Project Management

The project was planned to last 510 hours. The starting date is on February 15th and the delivery date is on June 28th. This gives us 134 days to distribute 520 hours of works. I plan to invest 4 hours per day during this period of time, which a priory seems enough to successfully complete the project. However the project finally had to be extended until October 25th for reasons that will be later explained.

10.1 Task definition

It is a common practice to subdivide projects into smaller units of work named tasks. This subdivision makes organizing the work between different members much easier. But even in a thesis where most of the work will be carried out by one member diving the work into tasks makes it easier to organize the project and estimate the amount of time required to complete it.

Project planning

This part is considered the initial phase of the thesis. The tasks included will not try to achieve an overall objective or sub-objective. Rather their work will serve as a stable foundation upon which the project will be built.

- **Context and scope:** Introduction and contextualization of the problem. Define the problem to be solved and the objectives and sub-objectives to be achieved. Establish a methodology and validation plan for the thesis.
- **Temporal planning:** Divide the project into different tasks. Establish a temporal plan to ensure that the project will be finished in time. Define a risk contingency plan to undermine the effects of possible unexpected events.
- **Budget and sustainability:** Estimate the total monetary cost of the project. Analyse the potential environmental impact that this project could have.
- **Final project integration:** Modify or correct all the parts according to the feedback received. Integrate all the corrected texts into a final document that will be evaluated.
- **Meetings:** Schedule biweekly virtual meetings with the director of the thesis to ensure the project is progressing successfully. Extraordinary sessions can also be arranged when needed.

Theoretical part

- **Read research papers related to the subject area:** Familiarize with different feature selection techniques. Get to know the state-of-the-art and commonly used in literature algorithms.

- **Conceptual design of the new proposed algorithm:** Define a new hybrid algorithm from a theoretical standpoint. This process can be subdivided into these smaller subtasks:
  - Write the pseudocode of the algorithm
  - Design the graph of the execution
  - Analyse the time and space complexity.
- **Design of experiments:** Study different experimental design techniques an assess which one to use in the project.

Practical part

- **Implement proposed algorithm:** Implement the proposed algorithm and its possible different versions in a programming language.
- **Implement other FS algorithms:** Implement other hybrid feature selection algorithms that will be used in the experimentation.
- **Test:** To ensure the correctness of the code written, some time will be dedicated to testing.

Experimentation and analysis

- **Select datasets:** Gather a list of real-world datasets from public repositories.
- **Synthetic dataset generation:** Implement a function that receives several relevant parameters as input and is able to quickly generate synthetic datasets matching the required characteristics.
- **Design of experiments:** Design the experimental set-up that will be used in both the synthetic and real-world benchmarks. Choose metrics to be used to compare the algorithms.
- **Experimentation:** Execute the algorithm in the selected datasets and store the performance data. This is by far the task that will be more computationally complex.
- **Analysis and conclusion:** Analyse the results obtained in the experimental part. Compare the results obtained with those of other well-known feature selection papers. Reach final conclusions.

Documentation

The documentation of the project will be carried out simultaneously with some of the other tasks. In fact, documentation will mostly be done near the end of each major group of tasks. Despite this, a final documentation task will also be included to correct spelling errors and possible past mistakes.

10.2 Summary of tasks

The following table contains a list of all tasks. Each task will be defined by the description of the work that involves, the number of hours required and some dependencies or tasks that need to be finished before it.

Task ID	Description	Time(h)	Dependencies
T1	**Project management**	**75**	
T1.1	Context and scope	30	
T1.2	Temporal planning	15	
T1.3	Budget and sustainability	15	
T1.4	Final document integration	15*	T1.1, T1.2, T1.3
T2	**Theoretical part**	**130**	
T2.1	Research	75	
T2.2	Conceptual design of the new algorithm	40	
T2.3	Write pseudocode	5	T2.2
T2.4	Execution graph	5	T2.2
T2.5	Complexity analysis	5	T2.2, T2.3
T3	**Practical part**	**55**	
T3.1	Implement the proposed algorithm	35	T2
T3.2	Testing	20	T3.1
T4	**Experimentation**	**130**	
T4.1	Make synthetic datasets	10	
T4.2	Select real world datasets	10	
T4.3	Experiments	60	
T4.4	Analysis and conclusions	50	T4.3
T5	**Meetings**	**20**	
T6	**Documentation**	**75**	
T7	**Defense preparation**	**25**	T6
Total		**510**	

Table 12: Tasks summary. [Compiled by the author]

* The duration of this task greatly depends on the feedback received by the GEP tutor.

10.3 Gantt chart

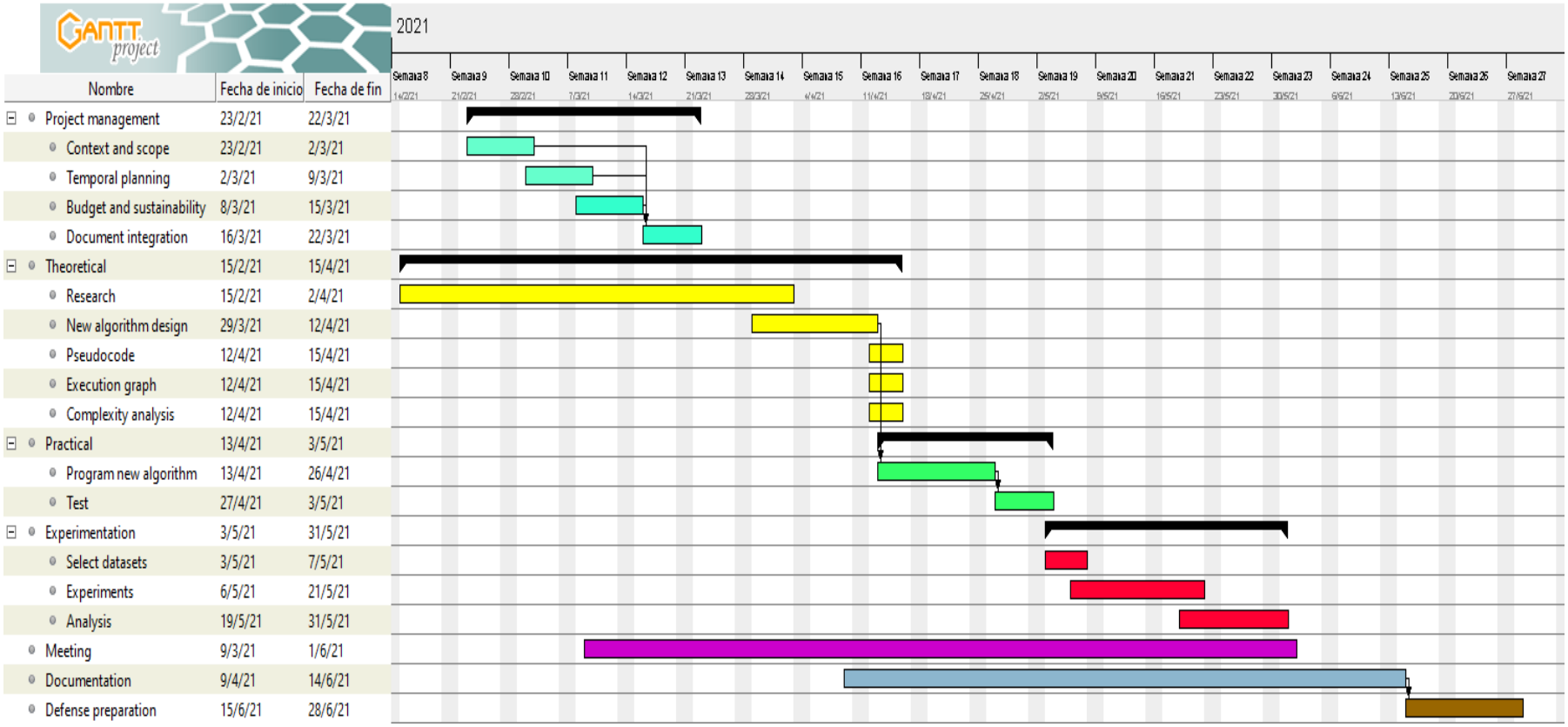

Figure 15: Gantt chart. [Compiled by the author]

10.4 Risk contingency plan

It would be naïve to expect that everything will carry out as defined in the original planning. Some obstacles are likely to appear during the development of the project that could jeopardize its progress. Some of these threads are very difficult to predict in the initial phases, nevertheless, some of them can be identified at the start of the project. The following contingency plan will help limit the impact of some of the potential risks we already identified. For each of them, the problem will be presented, along with a risk evaluation and a proposed solution. We will also include the final outcome explaining whether or not these potential risks played a major role in the project.

Tight schedule

- **Problem:** There is a fixed deadline marked by the university on which the thesis needs to be presented. Although a temporal plan has been established to reach this deadline comfortably, it is possible that some tasks end up taking longer than expected.
- **Risk:** High risk
- **Proposed solution:** In case we deviate from the original planning, the number of hours per day dedicated to the project will be increased until we catch up with the initial plan. If the deviation is considerable, some time will be spent adapting the plan to the new situation. To account for all this possible time loss, an extra 30 hour task dedicated to delay contingency and rescheduling is planned.

  In an extreme case, where the delay is so severe that it may put at risk the completion of the project, some shortcomings may need to be accepted in order to finish in time.
- **Outcome:** Due to some problems that came up in the later stages of development, I was faced with two possible solutions: accept some shortcomings to finish in time or delay the presentation of the project until the extraordinary semester. After discussing both options with the director of the thesis we decided to opt for the later.

Computational power:

- **Problem:** Some feature selection algorithms, especially when used in high-dimensional datasets, may take more computational power than I have at my disposal.
- **Risk:** Medium risk

- **Proposed solution:** This problem can be solved by using additional software resources like Google Colab. This one, in particular, allows for the execution of code in external GPUs, which would help speed up the necessary computations. Nevertheless, this can still take an extra 10 hours of work.
- **Outcome:** Thanks to a careful design of experiments we were able to reduce the workload significantly. That meant that no extra computational resources were needed.

Inexperience with the programming language:

- **Problem:** As the programming language that will be used has not been decided yet, I may lack the experience required to start implementing the algorithms.
- **Risk:** Low risk
- **Proposed solution:** Add a 20 hour task to get used to the programming language. The dependency table will also need to be updated because this new task will have to be completed before the implementation part. To help speed up the learning process, extra resources like books or software courses may be used.
- **Outcome:** Choosing the programming language was a big decision for the project. The two possible choices were Python and R. After doing an early scouting for feature selection packages implemented in both languages i decided to choose Python because it had a recently implemented RBA package. In hindsight, the best decision would have probably been doing the implementation with R. This is because the lack of some basic utilities that were not implemented with Python made me waste more time than initially planned.

Implementation errors and bugs:

- **Problem:** Bugs may pass unnoticed for a long time. If this were to happen tasks completed thereafter may be impacted negatively and need to be redone.
- **Risk:** Low risk
- **Proposed solution:** A testing task has been added to the implementation part to catch these bugs as soon as possible. In the case a bug is detected after the experimentation, a new task will be added to remake the affected parts. The duration of this task will depend on the impact of the bug and how late it is caught. It is hard to estimate the exact duration of this task, however as we will be doing tests regularly, 10 extra hours of work should be enough.
- **Outcome:** All bugs have been caught in a reasonable time. No major bugs have gone unnoticed severely impacting the project.

Coronavirus

- **Problem:** If another lockdown were to happen, it would be impossible to make physical meetings with the tutor. Moreover, the thesis defense may also be presented online.
- **Risk:** Low risk
- **Proposed solution:** Arrange virtual meetings with my tutor using free software available like Meet. If the thesis were to be presented online, I would have to buy a new microphone and webcam. This risk does not require extra hours of work, rather it will have a monetary impact and therefore needs to be accounted for in the budget.
- **Outcome:** All meetings have been held online without a problem.

10.5 Resources

After having defined the tasks that compose the project, it is also necessary to analyse the resources needed to carry them out. Each resource will be classified into software, hardware, human or material.

Human resources

There will be three people involved directly with the project. The main human resource is the researcher who will be carrying out most of the work. The director will be responsible for guiding the researcher and ensuring the work done meets certain standards. Finally, the GEP tutor will be involved during the initial phase of the projects and his main responsibility will be to provide feedback on the scope, organization plan and economic dimension of the project.

Material resources

As this project is built upon previous research works, it is necessary to include some material resources such as books or scientific papers.

Hardware resources

Nearly every task involves the use of a computer in one way or another. This project will be done with a desktop computer with 16GB of RAM and Intel(R) Core(TM) i5-7600 CPU @ 3.50GHz.

Software resources

During the project there will be a need for several software resources, each with its own distinct functionality. To document the project we will mainly use Overleaf,

which is a well-known Latex editor. However extra software may be used sporadically, such as Ganttproject to design the Gantt charts or BibTex to make the bibliography. In the practical part, some programming languages will be used to implement the algorithms. In order to make the implementation phase easier, an IDE such as Visual Studio Code will be employed. Other software resources that will be used are Github as a version manager, Trello to visually organize the tasks and Google Meet to do virtual meetings with the thesis director. In order to communicate with the director and the GEP tutor, an email service or Atenea will be used.

10.6 Methodology and validation

To maximize the usefulness of the meetings with the tutor, we will work following a combination of the agile and Kanban methodologies.

The major advantage of following an agile approach is that we will be able to subdivide the work into two-week sprints. At the end of each sprint, a meeting with the tutor will be held. This meeting will work as a feedback loop. Depending on the assessments of the tutor, future sprints may be modified to for example correct something that was pointed out to be wrong.

Each sprint will be internally organized using the Kanban approach. For each sprint, we will use Trello to set up a Kanban board with all the work that needs to be done. The main reason to choose this technique for project management is that it is a visual tool that helps to easily identify what needs to be done and what is already finished. We will set up the cards in the following way:

- **To Do:** This group will include all the tasks that have not been started.
- **Doing:** This group will include all the tasks that are in progress of being made.
- **Testing:** This group will include all the tasks that are finished but are not yet tested. Tasks in the testing group will mostly be software-related.
- **Done:** This group will include all the tasks that are finished and tested.

In this project, we will use Github as our tool of version control. Having a version control will also help us recover past versions in case of a fatal mistake and pinpoint when some bug was introduced to the codebase. As for the majority of projects, a master branch will contain all the tested code and a development branch will be used as a testing environment.

Depending on the availability of the tutor, virtual weekly or biweekly meetings will be arranged. These meetings will be used both to maintain a constant communication channel where we can discuss the progress of the project and to validate that the work done up to that point is correct.

10.7 Budget

This section will try to estimate all the potential costs involved in the project. From personnel payments to other direct and indirect costs.

10.7.1 Staff costs

Personnel payments will take up the vast majority of the budget. The project involves different types of workers with different levels of experience and salaries. The team will be formed by a *junior project manager* who will plan the project, a *junior developer* and a *tester* that will implement the algorithms and test their correct functionality. There is also a need for a *junior researcher* who will be in charge of the theoretical part of the work. All the aforementioned roles will be carried out by the student that is doing the thesis.

Additionally, two senior project managers will also be included, they will be in charge of guiding and managing the project. These two figures correspond to the director of the thesis and the GEP tutor.

Each member of the group will be assigned different tasks. Therefore, the hours spent working on the project will differ greatly between members. For this reason, cost per hour is the most informative metric to estimate the final personnel costs. However, this metric is rather difficult to find. To estimate it we first gather the annual salaries for each position, multiply them by 1.35 to approximate social security payments and finally divide them by the number of hours worked in a year (1750 hours). Another factor that can influence the costs is the location. As all the members of the project are located in Barcelona, salaries from only this area will be gathered.

Position	**Annual salary (€)**	**Price per hour (€/h)**
Senior project manager*	44800	34.56
Junior project manager*	27800	21.33
Junior developer	21000	16.2
Junior researcher	18700	14.31
Tester	17500	13.5

Table 13: Salaries summary. [Compiled by the author]

* To estimate the salary of the project managers. A high percentile salary has been selected for the senior manager and a low percentile salary for the junior manager.

To obtain the final staff costs it is necessary to determine which member will carry out each task.

Task ID	Description	Time(h)	Senior manager	Junior manager	Researcher	Developer	Tester
T1	**Project management**	**75**	**15**	**75**	**0**	**0**	**0**
T1.1	Context and scope	30	3	30	0	0	0
T1.2	Temporal planning	15	3	15	0	0	0
T1.3	Budget and sustainability	15	3	15	0	0	0
T1.4	Final document integration	15	6	15	0	0	0
T2	**Theoretical part**	**130**	**5**	**0**	**130**	**0**	**0**
T2.1	Research	75	5	0	75	0	0
T2.2	Conceptual design	40	0	0	40	0	0
T2.3	Write pseudocode	5	0	0	5	0	0
T2.4	Execution graph	5	0	0	5	0	0
T2.5	Complexity analysis	5	0	0	5	0	0
T3	**Practical part**	**55**	**0**	**0**	**0**	**35**	**20**
T3.1	Implementation	35	0	0	0	35	0
T3.2	Testing	20	0	0	0	0	20
T4	**Experimentation**	**130**	**0**	**0**	**130**	**0**	**10**
T4.1	Make synthetic datasets	10	0	0	10	0	0
T4.2	Select real world datasets	10	0	0	10	0	0
T4.3	Experiments	60	0	0	60	0	10
T4.4	Analysis and conclusions	50	0	0	50	0	0
T5	**Meetings**	**20**	**20**	**20**	**20**	**20**	**20**
T6	**Documentation**	**75**	**0**	**0**	**75**	**0**	**0**
T7	**Defense preparation**	**25**	**0**	**25**	**0**	**0**	**0**
Total		**510**	**40**	**120**	**355**	**55**	**50**

Table 14: Tasks distribution summary. [Compiled by the author]

Multiplying the average costs per hour shown in 13 by the number of hours computed in 14 we can obtain the final personnel budget.

Position	Hours	Cost (€)
Senior project manager	40	1382.4
Junior project manager	120	2559.6
Junior developer	55	891
Junior researcher	355	5080
Tester	50	675
Total		10588

Table 15: Staff costs. [Compiled by the author]

The final budget shown in 15 can also be subdivided further into several individual task budgets. This will allow us to detect budget deviations much earlier.

Task ID	Description	Time(h)	Budget (€)
T1	**Project management**	**75**	**2118.15**
T1.1	Context and scope	30	743.58
T1.2	Temporal planning	15	423.63
T1.3	Budget and sustainability	15	423.63
T1.4	Final document integration	15	527.31
T2	**Theoretical part**	**130**	**2033.05**
T2.1	Research	75	1246
T2.2	Conceptual design of the new algorithm	40	572.4
T2.3	Write pseudocode	5	71.55
T2.4	Execution graph	5	71.55
T2.5	Complexity analysis	5	71.55
T3	**Practical part**	**55**	**837**
T3.1	Implement the proposed algorithm	35	567
T3.2	Testing	20	270
T4	**Experimentation**	**130**	**1995.3**
T4.1	Make synthetic datasets	10	143.1
T4.2	Select real world datasets	10	143.1
T4.3	Experiments	60	993.6
T4.4	Analysis and conclusions	50	715.5
T5	**Meetings**	**20**	**1998**
T6	**Documentation**	**75**	**1073.25**
T7	**Defense preparation**	**25**	**533.25**
Total		**510**	**10588.05**

Table 16: Budget per task. [Compiled by the author]

10.7.2 Generic costs

Amortization

Another aspect to take into account is the amortization of the resources that will be utilized in the project. Software resources will be open-source and free to use. Therefore, this section will mostly focus on the amortization of office equipment and different hardware devices such as a desktop computer and several peripherals. To compute the amortization the following formula is applied:

$$\text{Amortization} = \frac{\text{purchase price}}{\text{life expectancy} * \text{working days} * \text{hours per day}} * \text{total hours}$$

Where the working days are 220 and the total hours of the project 510.

Resource	Price (€)	Life expectancy (Years)	Amortization (€)
Desktop computer	1200	5	139.1
Asus monitor	150	5	17.38
Razer keyboard	80	3	15.45
Razer mouse	50	3	9.65
Desktop chair	200	5	23.18
Total			204.76

Table 17: Amortization costs. [Compiled by the author]

Indirect costs

In order to obtain a more realistic budget, indirect costs related to work also need to be considered. These costs may seem negligible at first, but start to take up a noticeable percentage of the budget when all of them are added up.

- **Internet cost:** Currently I am paying a fixed monthly fee of 75 euros per month for the internet service. Considering that the project lasts 5 months the total cost adds up to 75€/month * 5 months * 4h/24h = 62.5€.
- **Electricity cost:** My electricity provider has an average cost of 0.127 €/kWh. The power use of a desktop computer can vary depending on the activity it is performing. Nevertheless, it is reasonable to simplify the problem and consider an average power use of 200 W. The final electricity cost can be easily computed: 0.127 €/kWh * 0.2 kW * 4 hour/day * 134 days = 13.61 €.
- **Travel cost:** Due to the pandemic travel costs will largely be avoided. As of today, only the final defense of the thesis will be done face-to-face. To travel from my home to the university I will use the Barcelona Metro as transport. The price of a T-casual (10 trips) is 11.35 €. Thus, the final cost of transport will be: 2 trips * 11.35 €/10 trips = 2.27 €.

10.7.3 Risk control budget

Incidentals

In section 10.4, a complete risk contingency plan was developed to minimize the effect of possible foreseen threads. This section will try to estimate the possible budget deviations that may occur in case these events were to happen.

Risk	**Solution**	**Estimated cost (€)**	**Risk (%)**	**Cost (€)**
Thesis defense is online	Buy new webcam and microphone	150	70	105
Inexperience with the programming language	Add 25h work	405	20	82
Project deadline	Add 30h of work	429.3	50	214.65
Lack of computational power	Add 10h work	143.1	50	57.24
Bugs	Add 10h work	135	10	13.5
Total				472.39

Table 18: Incidental costs. [Compiled by the author]

Contingency budget

In this particular project, the indirect costs are not likely to vary, therefore, a 5% contingency margin is enough. On the other hand, staff expenses are more likely to change. This will normally occur due to delays or underestimation of the number of hours in the original planning. Therefore, a contingency fund corresponding to 20% of personnel costs is needed.

Type	**Budget**	**Margin (%)**	**Contingency fund (€)**
Staff costs	10588	20	2117.6
Generic costs	204.76	5	10.23
Total			2127.83

Table 19: Contingency costs. [Compiled by the author]

10.7.4 Total budget

The total budget of the project is just the sum of the costs obtained in the previous sections. In total we will have at our disposal **13392.98€** to spend.

10.8 Deviations

This chapter will try to explain the differences in planning and budget between what was initially stated to what ended up happening. The main deviation from the original planning has been the need to extend the project until October. The project was following the temporal planning shown in figure 15 up until week 20. However after a meeting with the director we decided to add another version of the algorithm and opted for a more complex design of experiments. If this were to happen, the solution proposed in the risk contingency plan was to add a 30 hour task. Despite this due to my inability to dedicate more hours per day during working days and the closeness to the final exams of other subjects, we considered that the best decision to avoid shortcomings would be to extend the project.

10.8.1 Tasks

First we will present the updated table of tasks detailing the real time each one took to complete as well as some new tasks that were not considered in the original plan.

Task ID	Description	Final Time	Expected time
T1	**Project management**	**65**	**75**
T1.1	Context and scope	30	30
T1.2	Temporal planning	15	15
T1.3	Budget and sustainability	15	15
T1.4	Final document integration	5	15
T2	**Theoretical part**	**130**	**130**
T2.1	Research	80	75
T2.2	Conceptual design of the new algorithm	30	55
T2.3	Study design of experiments	20	0
T3	**Practical part**	**90**	**55**
T3.1	Implement the proposed algorithm	30	35
T3.2	Implement other FS algorithms	40	0
T3.3	Testing	20	20
T4	**Experimentation**	**165**	**130**
T4.1	Make synthetic datasets	20	10
T4.2	Select real world datasets	5	10
T4.3	Experiments	90	60
T4.4	Analysis and conclusions	50	50
T5	**Meetings**	**20**	**20**
T6	**Documentation**	**70**	**75**
T7	**Defense preparation**	**25**	**25**
Total		**565 (h)**	**510 (h)**

Table 20: Tasks summary. [Compiled by the author]

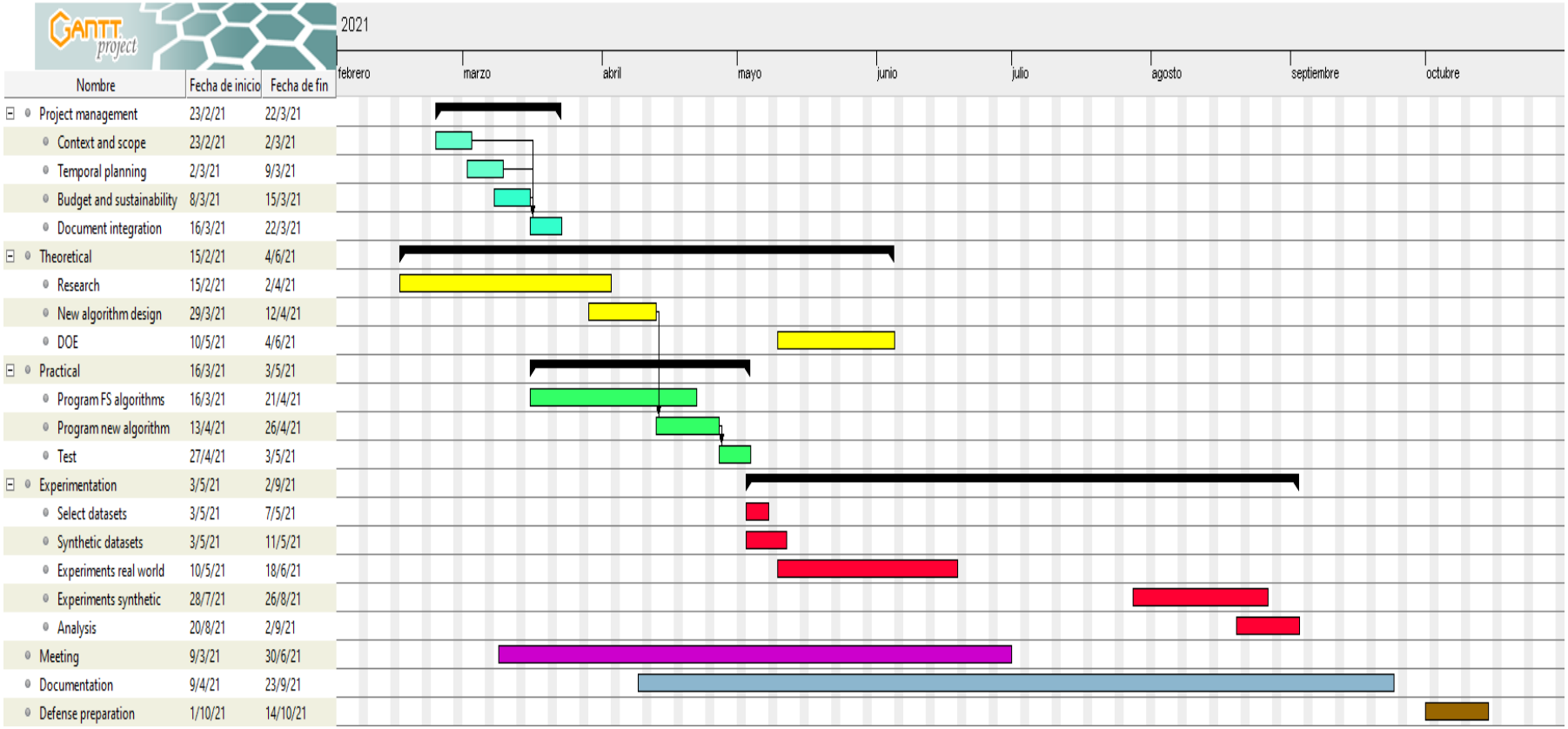

Figure 16: Final Gantt chart. [Compiled by the author]

Although some tasks had small deviations from the original temporal planning, in general these tended to balance out, making the total hours dedicated to each major task group in line with expectations. The only main discrepancies were the following:

- **Implementation:** We decided to program more versions of the proposed algorithm and more hybrid algorithms to compare to. This added a time cost of 35 additional hours.
- **Experimentation:** The use of high-dimensional datasets in the real-world experimentation combined with a more complex synthetic design raised the costs by 35 hours.

In total we needed to expend 70 extra hours to complete T2 and T3. However we also gained 15 hours from T1.4, T6. This gives us a balance of 55 extra hours, which is not that much of from the 30 hours mentioned in the contingency plan.

10.8.2 Budget

In this section we are going to discuss how the extra hours affected the final budget of the project.

Task ID	Description	Time(h)	SM	JM	R	D	T	Cost
T1.4	Final integration	-10	0	-10	0	0	0	-213.3
T3	Practical part	+35	0	0	0	+35	0	+567
T4	Experimentation	+35	0	0	+35	0	0	+500.85
T6	Documentation	-5	0	0	-5	0	0	-71.55
Total		**+55**	**0**	**-10**	**+30**	**+35**	**0**	**+783**

Table 21: Extra budget summary. [Compiled by the author]

* SM: Senior Manager, JM: Junior Manager, R: Researcher, D: Developer, T: Tester

Considering all the extra personnel cost our budget increased by 783€. Although it seems like a big increase, it is in fact well covered by the 2117.6€ staff contingency fund (Section 18). This still leaves with 1334.6€ left to spend on unforeseen extra personnel costs.

10.8.3 Methodology

The methodology proposed in the final GEP document was a combination of the agile and Kanban methodologies. There would be a sprint every couple of weeks and its tasks would be internally organized using the Kanban approach.

We followed this methodology for the first 6 weeks. In the initial stages of the project, this way of working was great because it allowed us to organize tasks around the biweekly meetings with the director. These meetings worked as a feedback loop in which all doubts were resolved and new tasks would be set.

After the seventh week, we decided to slightly change the methodology. The main difference was that we stopped working using an agile methodology. The Kanban approach was now used to organize the overall project rather than each sprint. The main reason for this change is that after the initial phase, most doubts were already resolved and I had a clearer idea of what needed to be done. Because of this and several other reasons, the time between meetings increased. As the methodology heavily relied on biweekly sprints followed by meetings, we decided that it was best to move to a new methodology in which the work was organized into larger tasks.

10.9 Sustainability

10.9.1 Self-assessments

In my opinion, there is a misconception about what exactly means to have a sustainable project. Generally speaking, when one thinks about how to evaluate whether or not a project is sustainable, the first thing that comes to mind is the economical dimension. A possible explanation for this may be because we live in a profit based-society where only those ideas that can be profitable end up being developed. This comes with a general disregard for the environmental or social impact that the project may cause, being more of a second thought rather than a criterion to be met.

Particularly in software-related projects, it is common to ignore the environmental sustainability because at first glance it is easy to think that they do not impact the environment. However, after having researched information about the different environmental indicators that are available, I reached the conclusion that they are quite useful and informative.

Another thing that I may have oversimplified in past projects is the economic aspect. While researching to develop the sustainability report, I have been introduced to many metrics I did not know about. To be honest, I also did not know that risk assessment and contingency planning were of such importance.

10.9.2 Environmental impact

Regarding PPP: Have you estimated the environmental impact of undertaking the project?

As the thesis is mostly focused on research and experimentation of feature selection algorithms there will not be any material resources needed. The only environmental impact may be the relatively high electricity consumption of those algorithms, especially when working with relatively large amounts of data

Regarding PPP: Have you considered how to minimise the impact, for example by reusing resources?

As the execution of feature selection algorithms increases in computational cost with respect to the size of the dataset, in the initial phase of experimentation, smaller datasets will be utilized to minimize the environmental footprint.

Regarding Useful Life: How is the problem that you wish to address resolved currently (state of the art)?

Feature selection is still an open area of research with several new techniques and different applications appearing every year. There is no definitive state-of-the-art, as techniques perform rather differently depending on the dataset type, size... In general, wrapper algorithms are mostly used in smaller datasets while filter, hybrid and embedded approaches are used when there is a higher dimensional space.

In what ways will your solution environmentally improve existing solutions?

The new proposed solution will be a hybrid filter-wrapper algorithm. In theory, it should outperform the filter method in predictive accuracy while being faster than a wrapper method.

10.9.3 Economic impact

Have you estimated the impact that your project will have, including both human and material costs?

Section 10.7.1 includes a detailed breakdown of human resources, it takes into account the cost per hour of different positions needed for the project and an estimate of the number of hours of work. Section 10.7.2 describes the different material resources and assigns a budget to each of them.

How is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution economically improve existing solutions?

As stated before there is no definite state-of-the-art algorithm in FS. Depending on the dataset a filter or wrapper technique may prove to be more effective. The proposed hybrid solution should be a middle ground between those techniques. It should result in higher accuracies than the filter method, a better performing model will guess right more often and thus in some applications increase profit. Moreover, it should also reduce execution time when compared with wrapper methods, hence reducing the electricity cost significantly.

10.9.4 Social impact

What do you think undertaking the project has contributed to you personally?

Personally speaking, this project has made me realize my love for investigation and innovation. It has also encouraged me to pursue a master and doctorate in the future to be able to work in a research group.

How will your solution improve the quality of life (social dimension) with respect other existing solutions?

The new solution will be another option that data analysts and computer engineers may need to consider when having to perform feature selection. It will be particularly used for those datasets where neither a filter nor a wrapper algorithm is a good solution. In cases where a middle ground solution is needed, researchers will no longer have to settle for either the filter or wrapper options.

Is there a real need for the project?

Both the tutor and the researcher agreed that this type of hybrid filter-wrapper FS is still underresearched. Therefore, although the final proposed technique may not become state-of-the-art, we think it is an area of research that is worth to explore.

11 Technical competences

In this section, we will explain the technical competences related to the project and justify how we met the established requirements.

CCO1.1: *To evaluate the computational complexity of a problem, know the algorithmic strategies which can solve it and recommend, develop and implement the solution which guarantees the best performance according to the established requirements.* [A little bit]

During the project we programmed most of the algorithms used in the experimentation. For each of them, we tried to implement them as efficiently as possible. This was crucial because as we dealt with datasets of high dimensionality, an inefficient implementation would most likely result in a large time penalty.

Moreover, for the algorithms we proposed, we have analysed both the time and space complexities.

CCO2.1: *To demonstrate knowledge about the fundamentals, paradigms and the own techniques of intelligent systems, and analyse, design and build computer systems, services and applications which use these techniques in any applicable field.* [Enough]

During the project, we used several different types of feature selection algorithms. We analysed RBAs and information-based filters that performed feature weighting and wrappers that used a learning algorithm to score subsets. We also studied several state-of-the-art hybrid algorithms. Thanks to all this traversal knowledge of the field of feature selection, we were able to algorithmically combine filters and wrappers obtaining an algorithm that integrates the best characteristics from both approaches.

CCO2.2: *Capacity to acquire, obtain, formalize and represent human knowledge in a computable way to solve problems through a computer system in any applicable field, in particular in the fields related to computation, perception and operation in intelligent environments.* [A little bit]

In the experimentation section, we have selected several real-world datasets that met our prerequisites. These datasets come from vastly different fields, however, the majority of them are high-dimensional microarray data.

Moreover, to complement the real-world experiments, we have also performed synthetic experimentation. In this part, we first needed to study which factors of the data are relevant for feature selection.

CCO2.4: *To demonstrate knowledge and develop techniques about computational learning; to design and implement applications and system that use them, including these ones dedicated to the automatic extraction of information and knowledge from large data volumes.* [In depth]

This competence is the one that is the most in line with our project. We have proposed and implemented several algorithms that use components of computational learning. These algorithms perform feature selection, which tries to extract the relevant information from the complete set of features.

Moreover, we have chosen hybrid feature selection because it is scalable to large volumes of data. In fact, the last two versions of the proposed algorithm implemented mechanisms specially designed for high-dimensional data.

12 Experiment results

12.1 Interaction plots

This section shows the complete set of interaction plots obtained from the synthetic experimentation. We will subdivide them into different groups depending on the predictor variable.

Accuracy - MSE

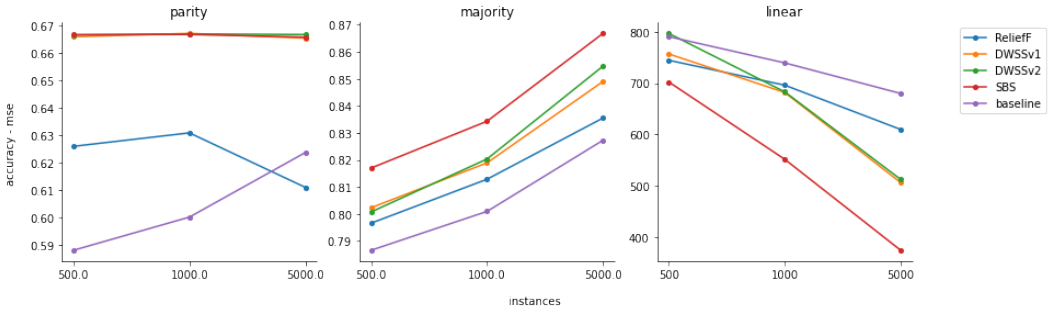

Figure 17: Instances and accuracy interaction plot.

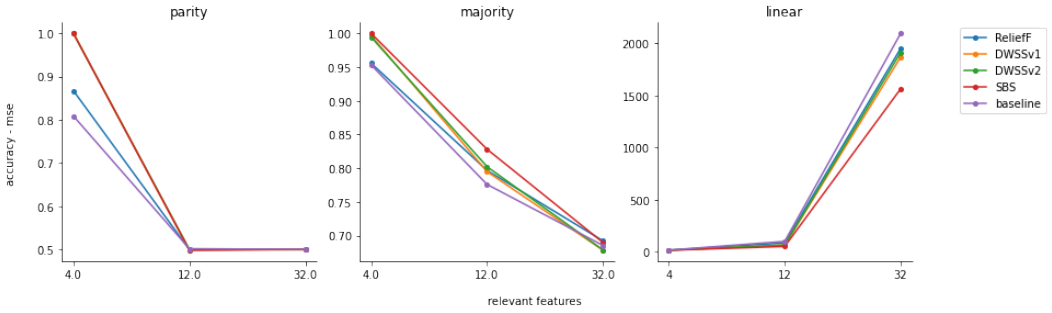

Figure 18: Relevance and accuracy interaction plot.

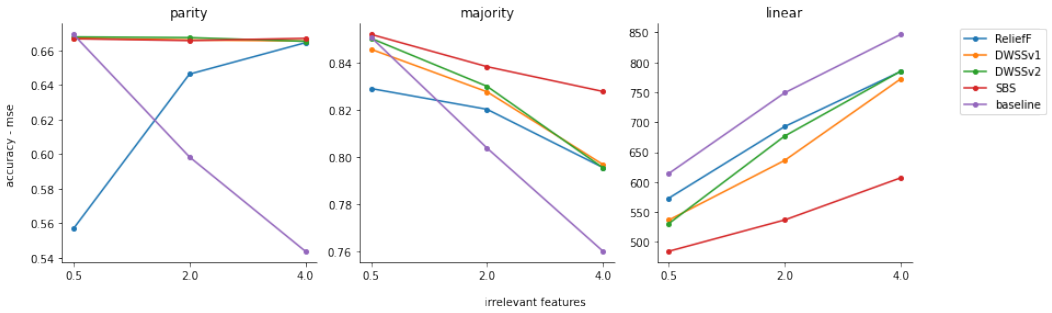

Figure 19: Irrelevance and accuracy interaction plot.

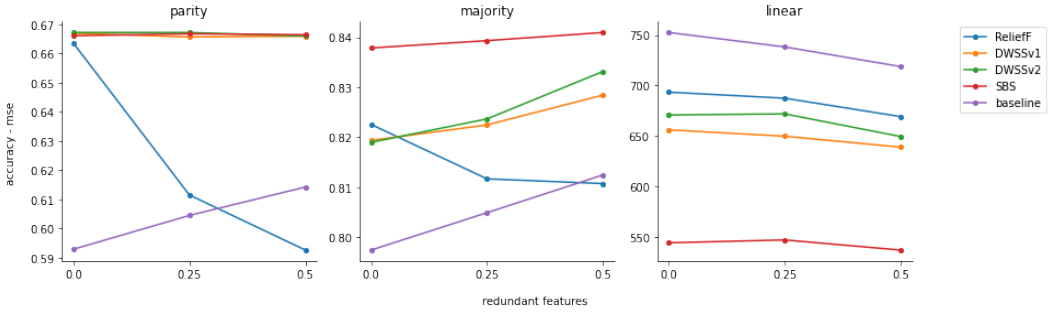

Figure 20: Redundancy and accuracy interaction plot.

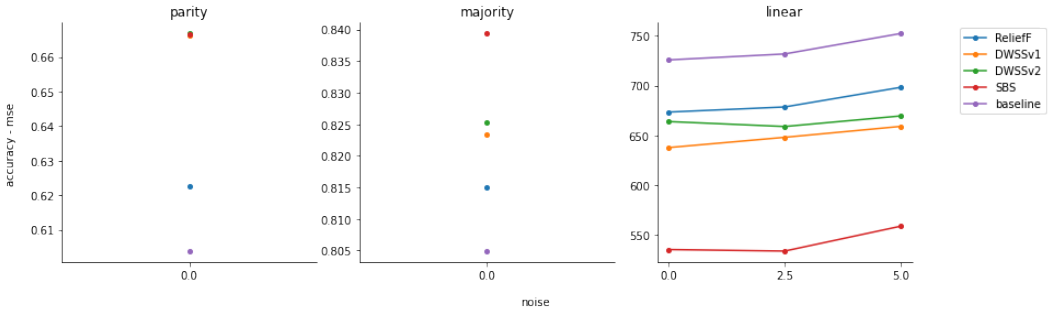

Figure 21: Noise and accuracy interaction plot.

Score

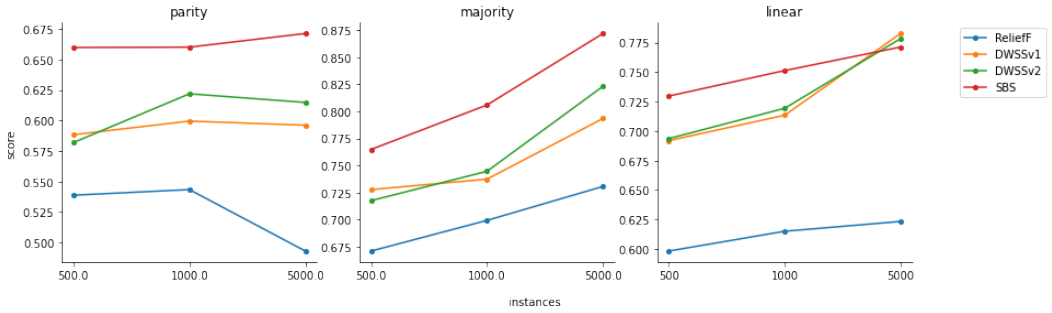

Figure 22: Instances and score interaction plot.

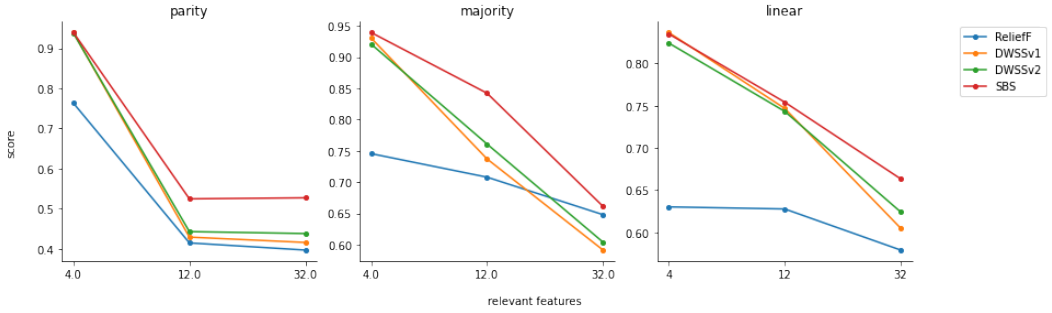

Figure 23: Relevance and score interaction plot.

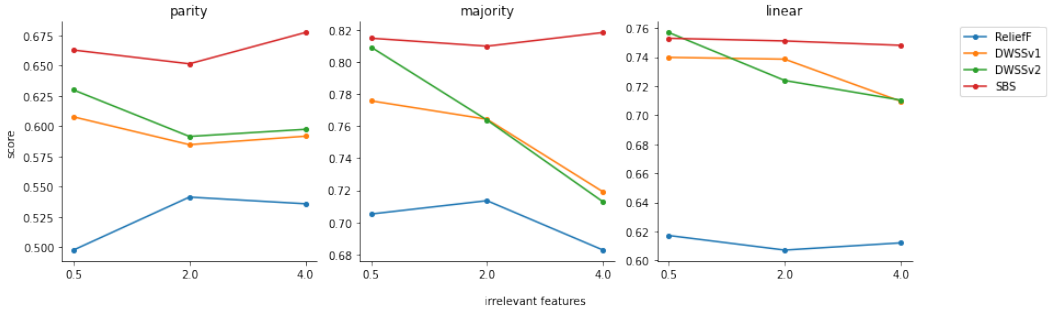

Figure 24: Irrelevance and score interaction plot.

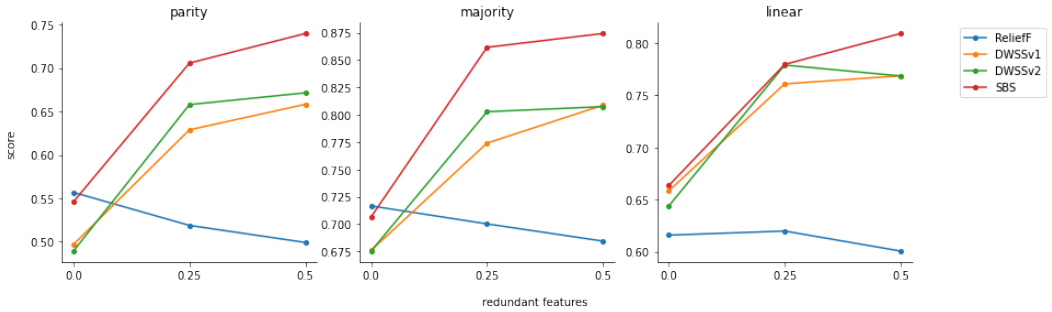

Figure 25: Redundancy and score interaction plot.

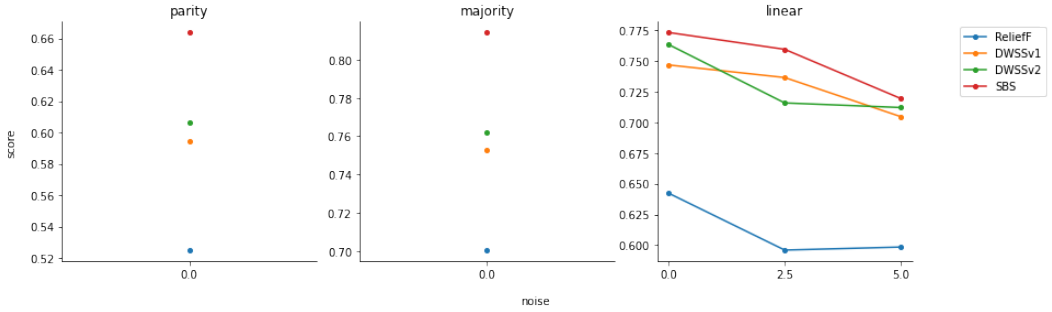

Figure 26: Noise and score interaction plot.

Time

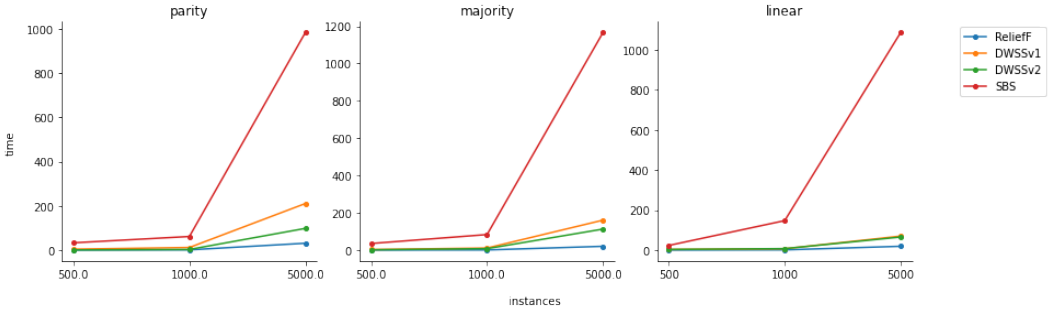

Figure 27: Instances and time interaction plot.

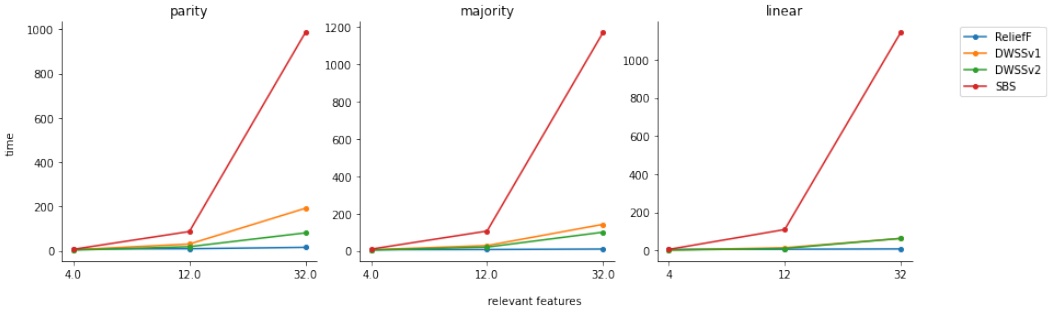

Figure 28: Relevance and time interaction plot.

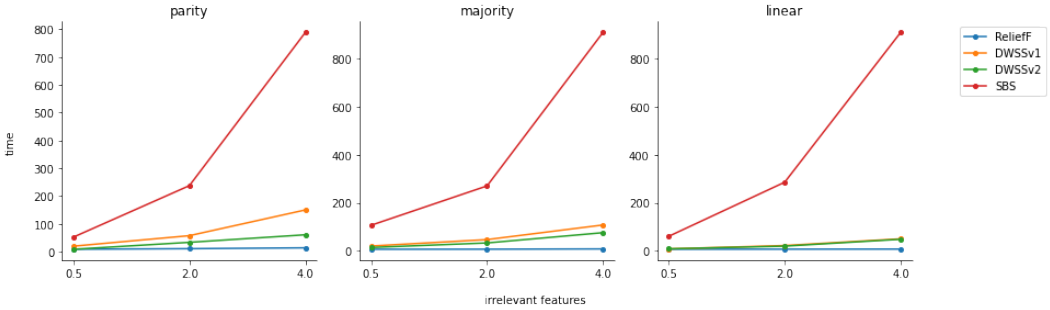

Figure 29: Irrelevance and time interaction plot.

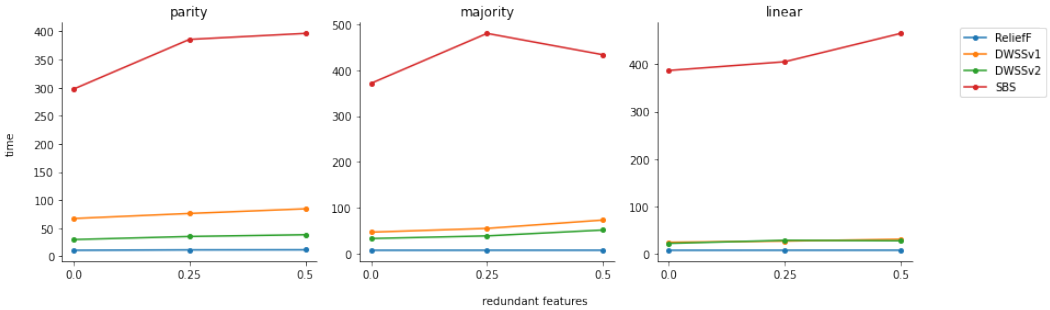

Figure 30: Redundancy and time interaction plot.

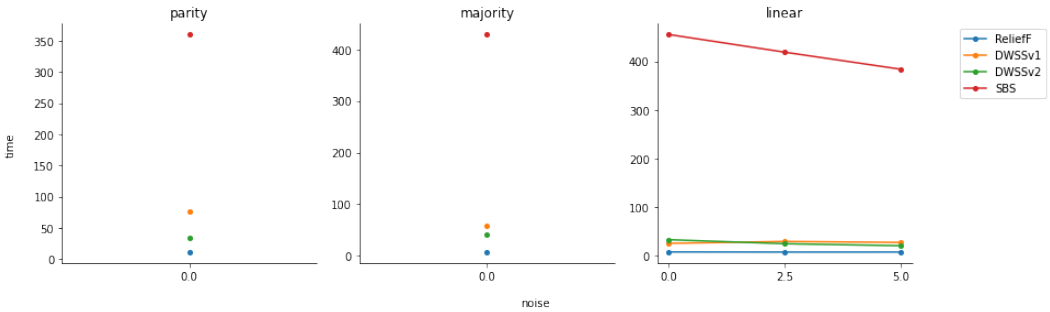

Figure 31: Noise and time interaction plot.

Wrapper count

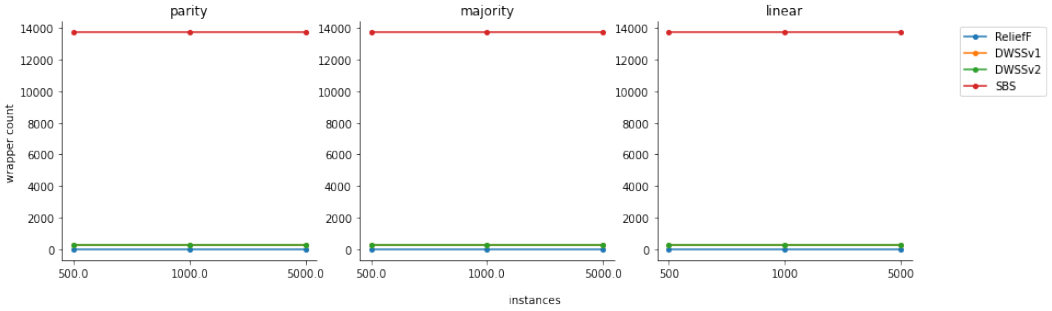

Figure 32: Instances and wrapper count interaction plot.

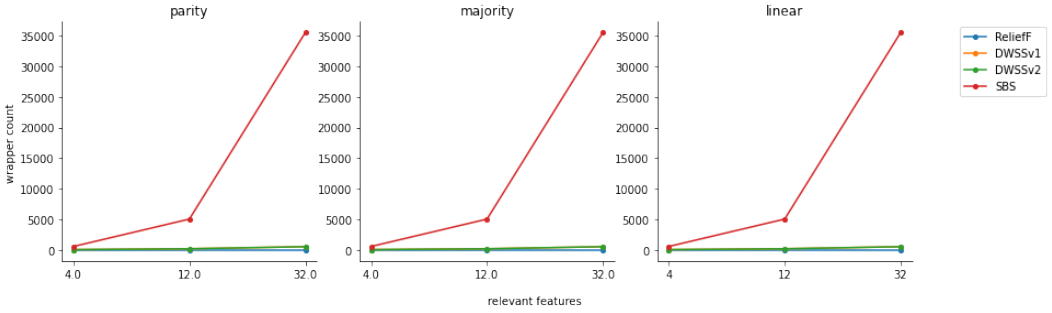

Figure 33: Relevance and wrapper count interaction plot.

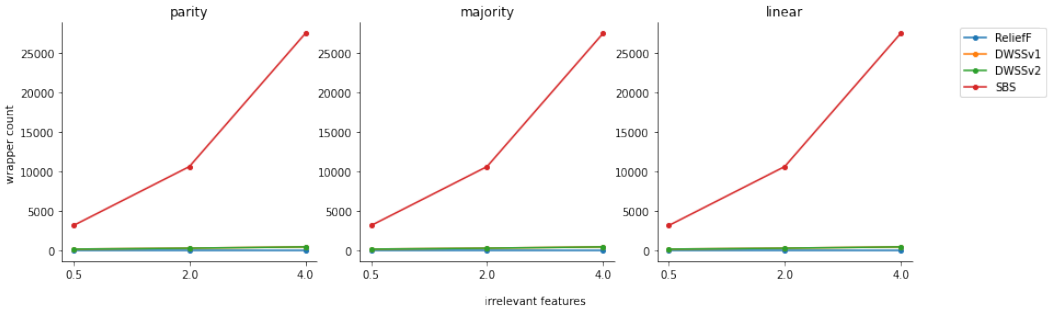

Figure 34: Irrelevance and wrapper count interaction plot.

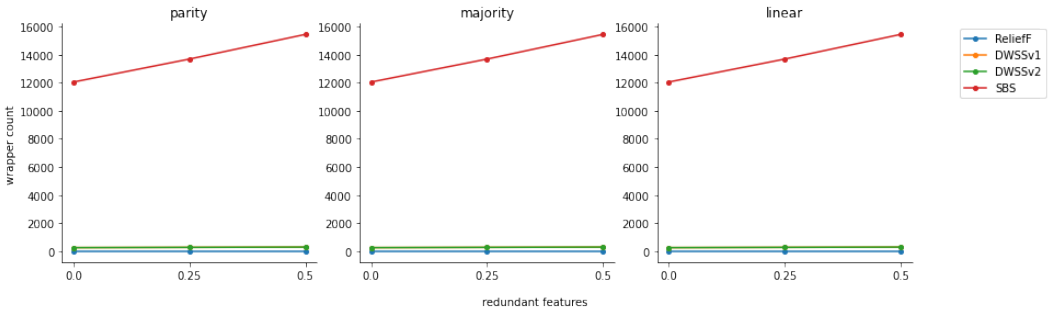

Figure 35: Redundancy and wrapper count interaction plot.

12.2 ANOVA analisys

In this section we display several tables of pvalues from the ANOVA analysis of synthetic datasets. We will only show up to two-level interaction of factors because higher order interactions are rarely relevant.

	Parity				Majority				Linear			
factors	ReliefF	DWSSv1	DWSSv2	SBS	ReliefF	DWSSv1	DWSSv2	SBS	ReliefF	DWSSv1	DWSSv2	SBS
inst	6.70e-01	9.81e-01	1.00e+00	9.83e-01	**1.56e-02**	**8.32e-03**	**1.13e-03**	**1.83e-04**	**8.01e-04**	**1.50e-08**	**4.03e-09**	**1.71e-13**
rel	**6.72e-09**	**6.91e-12**	**6.03e-12**	**8.31e-12**	**4.01e-25**	**1.68e-26**	**1.85e-28**	**5.50e-34**	**6.67e-127**	**5.80e-115**	**6.63e-114**	**1.92e-108**
irr	**2.67e-02**	9.73e-01	9.58e-01	9.94e-01	**2.55e-02**	**4.74e-03**	**6.71e-04**	5.72e-02	**1.79e-07**	**9.82e-08**	**4.86e-08**	**1.73e-03**
red	1.26e-01	9.84e-01	9.80e-01	9.94e-01	4.30e-01	5.92e-01	3.63e-01	8.03e-01	5.33e-01	6.89e-01	6.31e-01	8.53e-01
noise	-	-	-	-	-	-	-	-	5.23e-01	6.21e-01	8.99e-01	5.46e-01
inst:rel	6.65e-01	9.98e-01	9.96e-01	9.72e-01	2.77e-01	7.03e-01	2.04e-01	**3.27e-02**	**2.50e-05**	**1.55e-12**	**2.14e-13**	**1.17e-20**
inst:irr	9.20e-01	9.99e-01	9.98e-01	9.86e-01	8.80e-01	8.37e-01	4.91e-01	6.11e-01	8.26e-01	9.45e-01	9.19e-01	**3.43e-02**
inst:red	5.34e-01	9.82e-01	9.77e-01	9.88e-01	6.35e-01	9.28e-01	9.56e-01	6.61e-01	9.26e-01	6.17e-01	9.47e-01	9.74e-01
inst:noise	-	-	-	-	-	-	-	-	9.79e-01	8.19e-01	9.10e-01	9.90e-01
rel:irr	**2.51e-02**	9.91e-01	9.50e-01	9.93e-01	7.18e-02	3.56e-01	1.37e-01	6.27e-02	**1.59e-10**	**1.62e-11**	**5.75e-12**	**2.62e-05**
rel:red	1.19e-01	9.86e-01	9.99e-01	9.94e-01	1.89e-01	9.84e-01	6.28e-01	9.62e-01	4.09e-01	5.76e-01	4.83e-01	8.00e-01
rel:noise	-	-	-	-	-	-	-	-	8.24e-01	7.29e-01	4.05e-01	7.87e-01
irr:red	1.16e-01	9.91e-01	9.78e-01	9.91e-01	2.09e-01	5.55e-01	6.53e-01	9.53e-01	4.89e-01	3.88e-01	7.84e-01	9.33e-01
irr:noise	-	-	-	-	-	-	-	-	9.33e-01	8.09e-01	7.95e-01	7.41e-01
red:noise	-	-	-	-	-	-	-	-	9.16e-01	9.21e-01	2.15e-01	8.88e-01

Table 22: ANOVA analysis of factors with accuracy as predictor. The table shows the resulting pvalues, from which the ones inferior to 0.05 are **bolded**.

	Parity				Majority				Linear			
factors	ReliefF	DWSSv1	DWSSv2	SBS	ReliefF	DWSSv1	DWSSv2	SBS	ReliefF	DWSSv1	DWSSv2	SBS
inst	2.05e-01	9.48e-01	7.31e-01	7.73e-01	**1.10e-03**	**2.79e-03**	**6.33e-06**	**7.50e-07**	**1.90e-02**	**4.76e-11**	**4.15e-09**	**2.69e-03**
rel	**6.66e-10**	**3.72e-11**	**1.29e-10**	**1.53e-10**	**7.22e-08**	**1.58e-21**	**3.52e-21**	**1.17e-22**	**1.37e-09**	**9.90e-42**	**4.90e-33**	**1.31e-30**
irr	4.07e-01	7.94e-01	5.81e-01	7.21e-01	1.38e-01	**1.32e-02**	**4.64e-05**	8.25e-01	6.23e-01	**2.16e-02**	**1.26e-03**	6.95e-01
red	1.83e-01	**4.82e-03**	**1.41e-03**	**5.89e-05**	5.00e-02	**1.94e-07**	**1.02e-07**	**8.15e-13**	8.54e-02	**3.72e-14**	**1.69e-16**	**5.85e-25**
noise	-	-	-	-	-	-	-	-	**1.46e-06**	**2.19e-03**	**2.88e-04**	**2.14e-05**
inst:rel	4.58e-01	8.01e-01	5.54e-01	6.65e-01	1.24e-01	8.02e-01	6.39e-02	**9.89e-03**	**9.28e-04**	4.97e-01	4.87e-01	3.30e-01
inst:irr	2.15e-01	8.91e-01	9.35e-01	8.39e-01	7.91e-01	4.78e-01	8.50e-01	1.57e-01	9.86e-01	9.82e-01	3.43e-01	9.17e-01
inst:red	9.11e-01	7.64e-01	8.77e-01	9.05e-01	3.42e-01	8.27e-01	8.74e-01	3.53e-01	6.97e-01	5.69e-01	7.72e-01	3.31e-01
inst:noise	-	-	-	-	-	-	-	-	2.93e-01	1.93e-01	**8.32e-03**	3.65e-01
rel:irr	3.68e-01	8.20e-01	8.40e-01	8.54e-01	7.28e-01	3.77e-01	5.93e-01	3.99e-01	6.43e-01	1.12e-01	6.94e-01	**3.73e-02**
rel:red	7.00e-01	8.12e-01	6.44e-01	9.41e-01	**1.58e-03**	1.12e-01	2.08e-01	6.95e-02	9.34e-02	9.81e-01	4.27e-01	1.51e-01
rel:noise	-	-	-	-	-	-	-	-	**2.51e-07**	**9.08e-06**	**4.42e-03**	**3.86e-05**
irr:red	4.63e-01	9.21e-01	4.48e-01	7.58e-01	2.99e-01	8.45e-01	4.32e-01	8.92e-01	1.86e-01	3.18e-01	**1.19e-02**	6.23e-01
irr:noise	-	-	-	-	-	-	-	-	4.43e-01	2.67e-01	8.38e-01	9.23e-01
red:noise	-	-	-	-	-	-	-	-	6.90e-01	2.62e-01	7.45e-01	7.06e-01

Table 23: ANOVA analysis of factors with score as predictor. The table shows the resulting pvalues, from which the ones inferior to 0.05 are **bolded**.

	Parity				Majority				Linear			
factors	ReliefF	DWSSv1	DWSSv2	SBS	ReliefF	DWSSv1	DWSSv2	SBS	ReliefF	DWSSv1	DWSSv2	SBS
inst	**7.94e-49**	**8.24e-32**	**1.51e-33**	**2.35e-19**	**4.13e-68**	**6.01e-34**	**2.62e-39**	**1.10e-14**	**3.04e-226**	**4.18e-108**	**4.79e-40**	**4.21e-54**
rel	**8.61e-14**	**4.35e-28**	**4.00e-26**	**4.64e-19**	**5.86e-26**	**5.78e-29**	**4.99e-32**	**1.68e-14**	**1.23e-44**	**3.82e-95**	**2.80e-37**	**9.75e-58**
irr	**4.64e-08**	**6.21e-19**	**1.26e-16**	**7.81e-13**	**1.94e-14**	**3.89e-18**	**2.71e-16**	**1.91e-08**	**4.13e-09**	**1.28e-64**	**1.54e-18**	**1.57e-37**
red	3.15e-01	1.09e-01	7.33e-02	2.45e-01	2.27e-01	**6.98e-04**	**5.92e-03**	6.29e-01	2.15e-01	**1.79e-04**	1.84e-01	1.55e-01
noise	-	-	-	-	-	-	-	-	9.70e-02	2.69e-01	**3.35e-03**	1.94e-01
inst:rel	**3.68e-18**	**6.69e-34**	**5.63e-33**	**1.51e-23**	**1.29e-31**	**2.75e-34**	**4.38e-38**	**2.84e-18**	**1.80e-36**	**1.87e-105**	**6.26e-43**	**6.58e-65**
inst:irr	**2.33e-11**	**6.12e-24**	**1.81e-22**	**1.45e-16**	**1.10e-19**	**9.55e-22**	**1.41e-20**	**1.09e-10**	1.84e-01	**6.10e-73**	**4.91e-21**	**3.79e-40**
inst:red	1.06e-01	**4.51e-02**	**1.44e-02**	1.54e-01	**2.51e-02**	**6.33e-04**	**1.10e-03**	8.12e-01	3.35e-01	**1.16e-05**	1.31e-01	2.41e-01
inst:noise	-	-	-	-	-	-	-	-	**4.46e-02**	**2.57e-02**	**1.22e-03**	6.32e-02
rel:irr	**2.29e-05**	**1.19e-20**	**6.17e-17**	**4.86e-16**	**2.89e-11**	**2.69e-19**	**3.73e-17**	**6.51e-11**	**1.80e-10**	**2.79e-63**	**6.15e-23**	**4.40e-47**
rel:red	3.22e-01	8.32e-02	2.54e-01	1.63e-01	**1.76e-02**	**6.20e-05**	**7.99e-05**	6.15e-01	2.92e-01	**6.14e-05**	1.78e-01	6.47e-02
rel:noise	-	-	-	-	-	-	-	-	5.20e-01	2.43e-01	**1.33e-04**	1.19e-01
irr:red	8.91e-02	3.02e-01	9.55e-01	4.82e-01	8.10e-01	**4.14e-03**	1.22e-01	7.16e-01	6.59e-01	1.53e-01	5.14e-01	8.59e-01
irr:noise	-	-	-	-	-	-	-	-	8.59e-02	6.57e-02	**1.43e-03**	7.63e-02
red:noise	-	-	-	-	-	-	-	-	6.66e-01	4.24e-01	**1.40e-02**	3.44e-01

Table 24: ANOVA analysis of factors with time as predictor. The table shows the resulting pvalues, from which the ones inferior to 0.05 are **bolded**.

12.3 Real world experimentation tables

datasets	Accuracy						
	BIRS	**IWSS**	**IWSS$_s$**	**iSFSM**	**BWRR$_{v2}$**	**BWRR$_{v3}$**	**BWRR$_{v4}$**
colon	0.743 ± 0.13	0.726 ± 0.072	0.726 ± 0.072	0.743 ± 0.124	0.774 ± 0.109	$\mathbf{0.826 \pm 0.105}$	0.721 ± 0.197
GLIOMA	0.6 ± 0.179	0.62 ± 0.108	0.7 ± 0.257	0.56 ± 0.233	$\mathbf{0.78 \pm 0.209}$	0.72 ± 0.183	$\mathbf{0.78 \pm 0.189}$
leukemia	0.93 ± 0.095	0.957 ± 0.091	0.957 ± 0.091	$\mathbf{0.971 \pm 0.057}$	0.902 ± 0.091	0.848 ± 0.205	0.902 ± 0.128
lung_discrete	0.698 ± 0.123	0.695 ± 0.108	0.695 ± 0.122	0.716 ± 0.133	0.793 ± 0.129	$\mathbf{0.821 \pm 0.139}$	0.779 ± 0.112
Prostate_GE	0.872 ± 0.158	$\mathbf{0.894 \pm 0.12}$	$\mathbf{0.894 \pm 0.12}$	0.873 ± 0.129	0.875 ± 0.102	0.834 ± 0.108	0.851 ± 0.136
TOX	0.544 ± 0.063	0.725 ± 0.09	0.72 ± 0.093	0.631 ± 0.118	0.748 ± 0.119	0.773 ± 0.139	0.742 ± 0.093
madelon	0.828 ± 0.112	$\mathbf{0.87 \pm 0.022}$	0.846 ± 0.043	0.867 ± 0.02	0.799 ± 0.045	0.786 ± 0.025	0.756 ± 0.039
arcene	0.77 ± 0.078	0.805 ± 0.085	0.795 ± 0.072	0.75 ± 0.095	0.78 ± 0.103	0.815 ± 0.112	$\mathbf{0.875 \pm 0.056}$
gisette	0.95 ± 0.007	0.966 ± 0.005	$\mathbf{0.967 \pm 0.005}$	0.923 ± 0.005	0.906 ± 0.035	0.935 ± 0.02	0.962 ± 0.006
Isolet	0.613 ± 0.075	$\mathbf{0.925 \pm 0.013}$	0.922 ± 0.014	0.774 ± 0.034	0.615 ± 0.111	0.625 ± 0.115	0.574 ± 0.085

Table 25: Accuracy on high-dimensional datasets.

datasets	Accuracy						
	BIRS	**IWSS**	**IWSS$_s$**	**iSFSM**	**BWRR$_{v2}$**	**BWRR$_{v3}$**	**BWRR$_{v4}$**
colon	15.558	15.298	7.907	5.866	17.553	13.306	3.083
GLIOMA	46.05	41.632	18.573	12.98	361.662	33.335	52.407
leukemia	64.216	59.542	19.803	17.375	97.448	18.81	17.476
lung_discrete	3.327	3.153	2.078	4.163	18.185	21.911	3.964
Prostate_GE	68.827	51.362	26.666	48.108	1228.576	31.509	242.622
TOX	123.686	124.028	56.995	43.009	4904.076	74.748	948.831
madelon	26.465	31.948	27.449	41.111	206.85	211.505	60.051
arcene	96.973	89.781	44.156	23.583	1359.733	31.947	807.115
gisette	1899.88	3223.75	578.598	330.56	19092.705	598.262	16041.196
Isolet	34.099	83.797	53.524	50.087	1943.175	295.498	18.798

Table 26: Times on high-dimensional datasets.

datasets	Accuracy						
	BIRS	**IWSS**	**IWSS$_s$**	**iSFSM**	**BWRR$_{v2}$**	**BWRR$_{v3}$**	**BWRR$_{v4}$**
colon	**4.0 ± 1.342**	9.5 ± 3.008	9.3 ± 2.685	8.4 ± 2.107	22.3 ± 17.018	31.5 ± 24.647	30.0 ± 20.11
GLIOMA	**5.3±1.1**	10.7 ± 1.676	9.4 ± 3.137	8.4 ± 2.2	77.2 ± 44.678	43.9 ± 32.297	52.6 ± 20.126
leukemia	**3.0 ± 0.894**	4.5 ± 1.285	4.5 ± 1.285	5.3 ± 2.934	3.4 ± 1.02	4.3 ± 4.713	16.0 ± 10.412
lung$_d$*iscrete*	**7.3 ± 1.676**	18.8 ± 4.915	16.5 ± 4.674	16.4 ± 3.499	49.3 ± 11.296	49.7 ± 12.067	31.8 ± 9.6
Prostate_GE	**4.45 ± 1.117**	8.4 ± 2.538	8.2 ± 2.676	10.45 ± 2.711	64.9 ± 42.034	43.0 ± 29.809	63.3 ± 28.852
TOX	**8.9 ± 1.758**	31.9 ± 7.203	28.9 ± 7.176	38.3 ± 4.88	239.8 ± 125.186	56.7 ± 37.744	176.3 ± 106.607
madelon	**11.3 ± 3.437**	32.5 ± 6.652	29.7 ± 10.593	13.0 ± 2.049	27.9 ± 8.006	25.6 ± 5.8	99.0 ± 47.527
arcene	12.3 ± 2.369	23.2 ± 4.534	21.7 ± 4.88	**5.9±1.446**	49.5 ± 70.418	26.5 ± 23.872	54.5 ± 20.647
gisette	49.3 ± 3.494	134.4 ± 11.137	107.4 ± 11.056	36.9 ± 5.069	124.8 ± 143.451	**31.6 ± 22.486**	139.3 ± 68.088
Isolet	**10.9±2.385**	108.2 ± 9.968	105.7 ± 9.768	37.1 ± 2.914	57.0 ± 45.732	24.6 ± 21.708	15.5 ± 6.786

Table 27: Feature subset size on high-dimensional datasets.

datasets	Accuracy						
	BIRS	**IWSS**	**IWSS$_s$**	**iSFSM**	**BWRR$_{v2}$**	**BWRR$_{v3}$**	**BWRR$_{v4}$**
colon	10000.0 ± 0.0	10000.0 ± 0.0	4051.5 ± 1926.538	2333.5 ± 598.101	10005.0 ± 0.0	5005.0 ± 0.0	527.5 ± 356.73
GLIOMA	22170.0 ± 0.0	22170.0 ± 0.0	6222.0 ± 1815.84	2081.5 ± 694.316	22175.0 ± 0.0	5005.0 ± 0.0	3069.0 ± 788.396
leukemia	35350.0 ± 0.0	35350.0 ± 0.0	7160.0 ± 76.354	3114.0 ± 371.199	35355.0 ± 0.0	5004.5 ± 1.5	1404.5 ± 407.544
lung$_d$*iscrete*	1625.0 ± 0.0	1625.0 ± 0.0	779.5 ± 190.571	2424.0 ± 876.892	1630.0 ± 0.0	1630.0 ± 0.0	602.0 ± 332.748
Prostate$_G$*E*	29830.0 ± 0.0	29830.0 ± 0.0	7501.5 ± 1274.219	2207.25 ± 387.59	29835.0 ± 0.0	5005.0 ± 0.0	3785.0 ± 1210.372
TOX	28740.0 ± 0.0	28740.0 ± 0.0	9774.5 ± 3680.781	8044.0 ± 1124.355	28745.0 ± 0.0	2505.0 ± 0.0	12073.0 ± 5642.741
madelon	2500.0 ± 0.0	2500.0 ± 0.0	2018.0 ± 505.694	3743.5 ± 978.43	2505.0 ± 0.0	2505.0 ± 0.0	1280.0 ± 378.319
arcene	50000.0 ± 0.0	50000.0 ± 0.0	18265.0 ± 3972.179	3347.5 ± 761.444	50005.0 ± 0.0	5005.0 ± 0.0	15205.0 ± 4812.484
gisette	25000.0 ± 0.0	25000.0 ± 0.0	5000.0 ± 0.0	2990.0 ± 1034.338	25005.0 ± 0.0	2505.0 ± 0.0	16880.0 ± 5320.068
Isolet	3085.0 ± 0.0	3085.0 ± 0.0	2518.0 ± 321.179	4770.5 ± 1007.486	3090.0 ± 0.0	2505.0 ± 0.0	721.0 ± 211.433

Table 28: Wrapper evaluations on high-dimensional datasets.

12.4 Real world experimentation boxplots

In this section we display the results of the experimentation on high-dimensional datasets as several different boxplots.

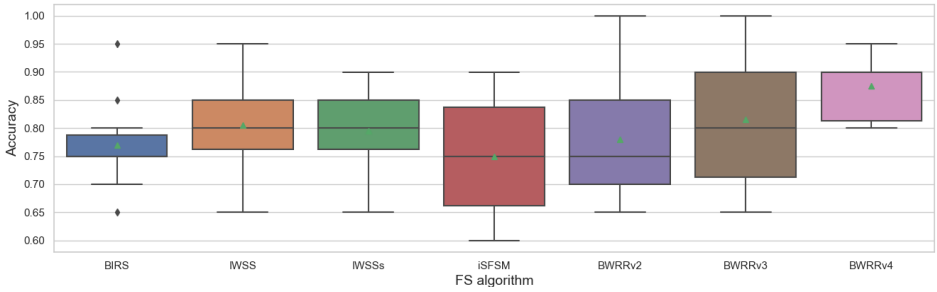

Figure 36: Accuracy boxplot on arcene dataset.

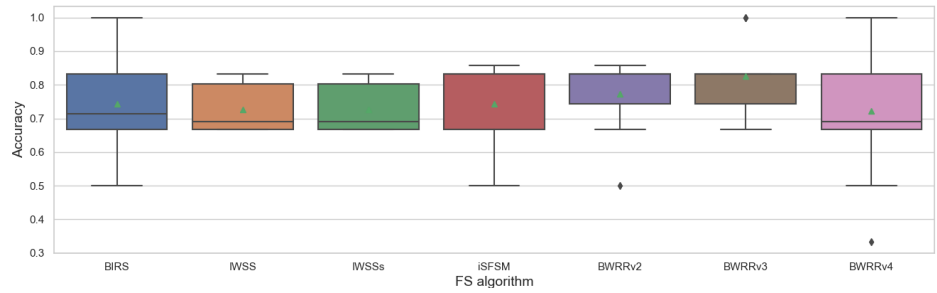

Figure 37: Accuracy boxplot on colon dataset.

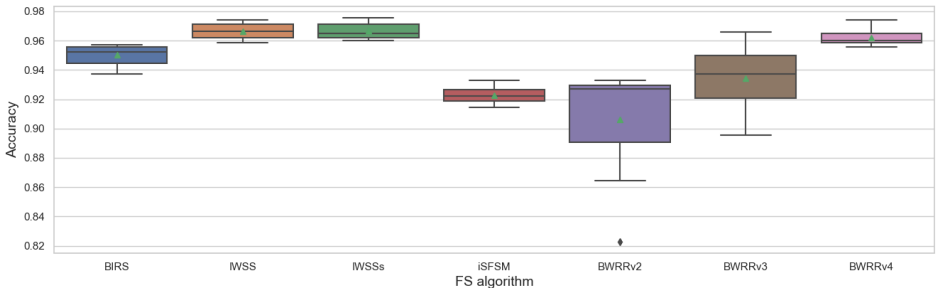

Figure 38: Accuracy boxplot on gisette dataset.

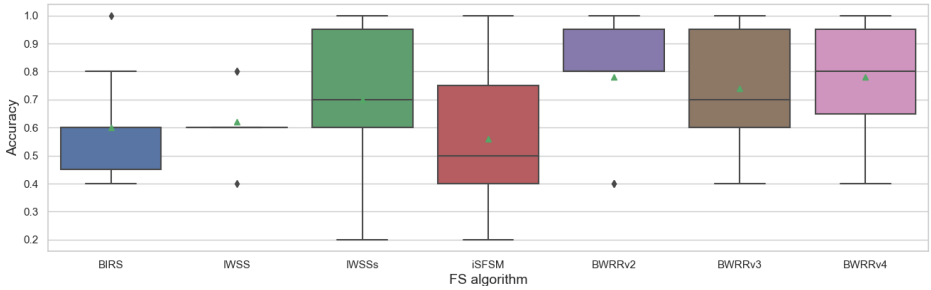

Figure 39: Accuracy boxplot on glioma dataset.

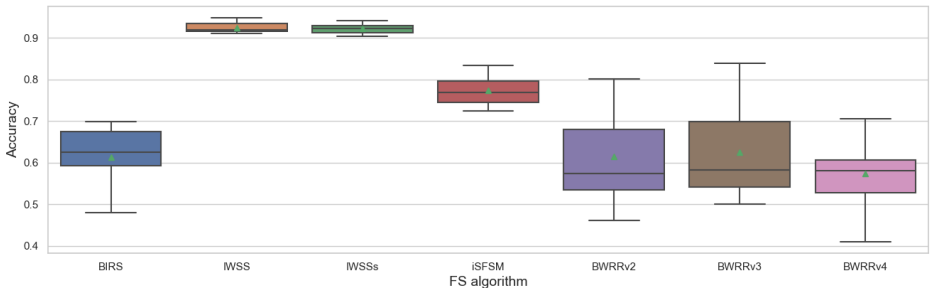

Figure 40: Accuracy boxplot on isolet dataset.

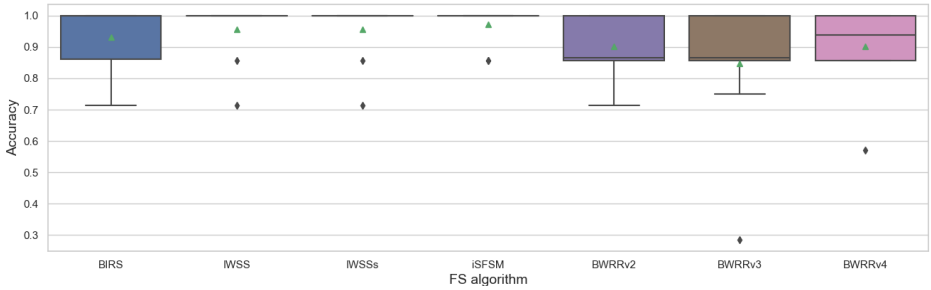

Figure 41: Accuracy boxplot on leukemia dataset.

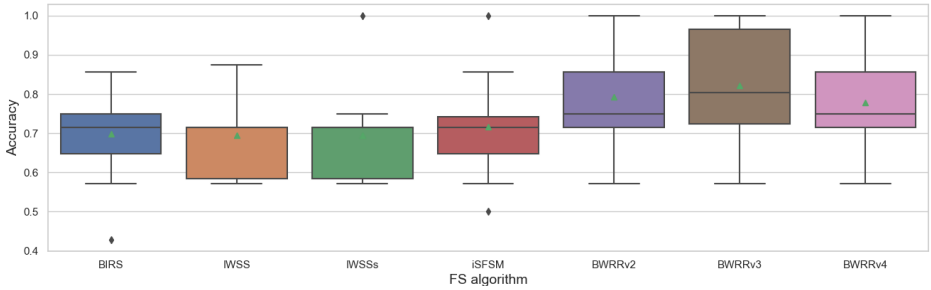

Figure 42: Accuracy boxplot on lung discrete dataset.

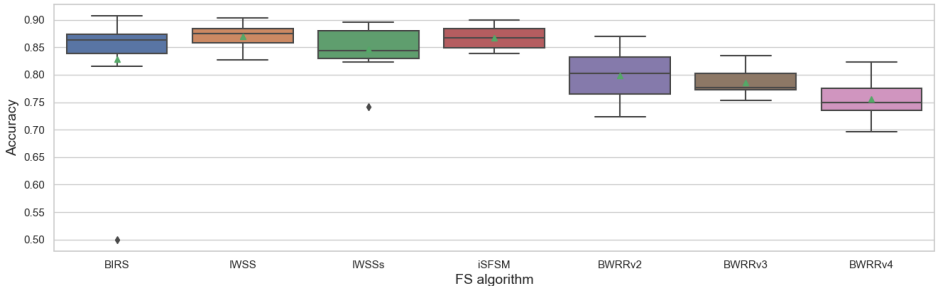

Figure 43: Accuracy boxplot on madelon dataset.

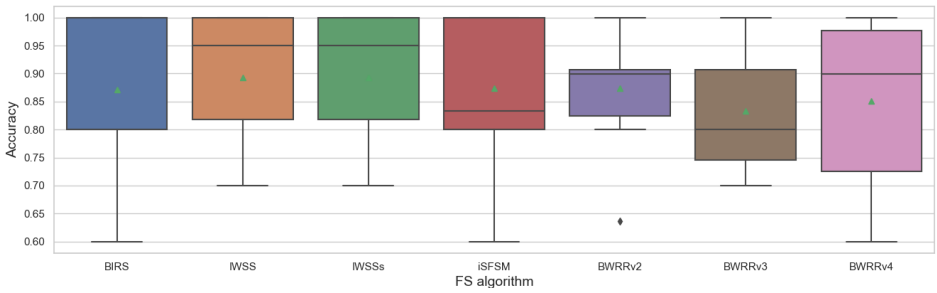

Figure 44: Accuracy boxplot on prostate dataset.

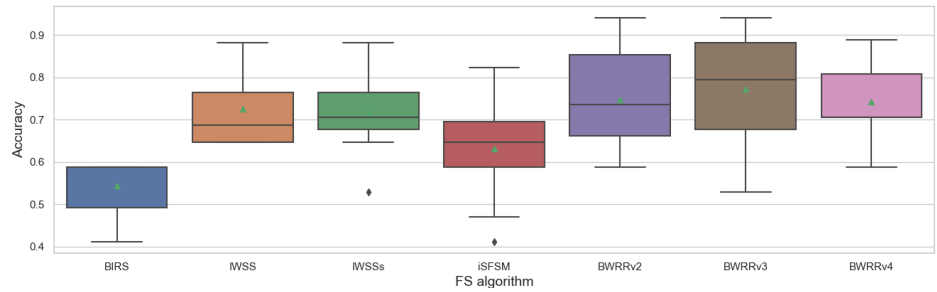

Figure 45: Accuracy boxplot on TOX dataset.

References

[1] Saeys, Y., Inza, I., and Larranaga, P., "A review of feature selection techniques in bioinformatics," *Bioinformatics (Oxford, England)*, vol. 23, pp. 2507–17, Nov. 2007. DOI: 10.1093/bioinformatics/btm344.

[2] Almugren, N. and Alshamlan, H., "A survey on hybrid feature selection methods in microarray gene expression data for cancer classification," *IEEE Access*, vol. 7, pp. 78 533–78 548, 2019. DOI: 10.1109/ACCESS.2019.2922987.

[3] Yu, L. and Liu, H., "Eficient feature selection via analysis of relevance and redundancy," *Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, Dec. 2004.

[4] John, G. H., Kohavi, R., and Pfleger, K., "Irrelevant features and the subset selection problem," in *Machine Learning Proceedings 1994*, Cohen, W. W. and Hirsh, H., Eds., San Francisco (CA): Morgan Kaufmann, 1994, pp. 121–129, ISBN: 978-1-55860-335-6. DOI: https://doi.org/10.1016/B978-1-55860-335-6.50023-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781558603356500234.

[5] Jović, A., Brkić, K., and Bogunović, N., *A review of feature selection methods with applications*, 2015. DOI: 10.1109/MIPRO.2015.7160458.

[6] Sánchez-Maroño, N., Alonso-Betanzos, A., and Tombilla-Sanromán, M., *Filter methods for feature selection – a comparative study*, Yin, H., Tino, P., Corchado, E., Byrne, W., and Yao, X., Eds., Berlin, Heidelberg, 2007.

[7] El Aboudi, N. and Benhlima, L., "Review on wrapper feature selection approaches," in *2016 International Conference on Engineering MIS (ICEMIS)*, 2016, pp. 1–5. DOI: 10.1109/ICEMIS.2016.7745366.

[8] Sun, L., Kong, X., Xu, J., Xue, Z., Zhai, R., and Zhang, S., "A hybrid gene selection method based on relieff and ant colony optimization algorithm for tumor classification," *Scientific Reports*, vol. 9, Jun. 2019. DOI: 10.1038/s41598-019-45223-x.

[9] Ruiz, R., Riquelme, J., and Aguilar-Ruiz, J., "Incremental wrapper-based gene selection from microarray data for cancer classification," *Pattern Recognition*, vol. 39, pp. 2383–2392, Dec. 2006. DOI: 10.1016/j.patcog.2005.11.001.

[10] Bermejo, P., Gámez, J., and Puerta, J., "On incremental wrapper-based attribute selection: Experimental analysis of the relevance criteria," Jan. 2008.

[11] Nakariyakul, S., "High-dimensional hybrid feature selection using interaction information-guided search," *Knowledge-Based Systems*, vol. 145, pp. 59–66, 2018, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2018.01.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705118300017.

[12] Leung, Y. and Hung, Y., "A multiple-filter-multiple-wrapper approach to gene selection and microarray data classification," *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, vol. 7, pp. 108–17, Jan. 2010. DOI: 10.1109/TCBB.2008.46.

[13] Vergara, J. R. and Estévez, P. A., "A review of feature selection methods based on mutual information," *Neural Computing and Applications*, vol. 24, no. 1, pp. 175–186, Mar. 2013, ISSN: 1433-3058. DOI: 10.1007/s00521-013-1368-0. [Online]. Available: http://dx.doi.org/10.1007/s00521-013-1368-0.

[14] Shannon, C. E., "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[15] Witten, I. H., Frank, E., and Hall, M. A., "Chapter 7 - data transformations," in *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*, ser. The Morgan Kaufmann Series in Data Management Systems, Witten, I. H., Frank, E., and Hall, M. A., Eds., Third Edition, Boston: Morgan Kaufmann, 2011, pp. 305–349, ISBN: 978-0-12-374856-0. DOI: https://doi.org/10.1016/B978-0-12-374856-0.00007-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123748560000079.

[16] Yang, H. H. and Moody, J., "Feature selection based on joint mutual information," in *In Proceedings of International ICSC Symposium on Advances in Intelligent Data Analysis*, 1999, pp. 22–25.

[17] "Feature selection using joint mutual information maximisation," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8520–8532, 2015, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2015.07.007.

[18] Kira, K. and Rendell, L., "The feature selection problem: Traditional methods and a new algorithm," in *AAAI*, 1992.

[19] Kononenko, I., "Estimating attributes: Analysis and extensions of relief," in *Machine Learning: ECML-94*, Bergadano, F. and De Raedt, L., Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 171–182, ISBN: 978-3-540-48365-6.

[20] Robnik-Sikonja, M. and Kononenko, I., "An adaptation of relief for attribute estimation in regression," *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, Feb. 2000.

[21] Moore, J. H. and White, B. C., "Tuning relieff for genome-wide genetic analysis," in *Evolutionary Computation,Machine Learning and Data Mining in Bioinformatics*, Marchiori, E., Moore, J. H., and Rajapakse, J. C., Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 166–175, ISBN: 978-3-540-71783-6.

[22] Robnik-Sikonja, M. and Kononenko, I., "Theoretical and empirical analysis of relieff and rrelieff," *Machine Learning*, vol. 53, pp. 23–69, Oct. 2003. DOI: 10.1023/A:1025667309714.

[23] Hu, Z., Bao, Y., Xiong, T., and Chiong, R., "Hybrid filter-wrapper feature selection for short-term load forecasting," *Engineering Applications of Artificial Intelligence*, vol. 40, pp. 17–27, Apr. 2015. DOI: 10.1016/j.engappai.2014.12.014.

[24] Ding, J. and Fu, L., "A hybrid feature selection algorithm based on information gain and sequential forward floating search," *Journal of Intelligent Computing*, vol. 9, p. 93, Sep. 2018. DOI: 10.6025/jic/2018/9/3/93-101.

[25] Hsu, H.-H., Hsieh, C.-W., and Lu, M.-D., "Hybrid feature selection by combining filters and wrappers," *Expert Syst. Appl.*, vol. 38, pp. 8144–8150, Jul. 2011. DOI: 10.1016/j.eswa.2010.12.156.

[26] Vollmer, M., Rutter, I., and Böhm, K., "On complexity and efficiency of mutual information estimation on static and dynamic data," in *EDBT*, 2018.

[27] Urbanowicz, R. J., Olson, R. S., Schmitt, P., Meeker, M., and Moore, J. H., "Benchmarking relief-based feature selection methods for bioinformatics data mining," *Journal of Biomedical Informatics*, vol. 85, pp. 168–188, 2018, ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2018.07.015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046418301412.

[28] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É., *Scikit-learn: Machine learning in python*, 2018. arXiv: 1201.0490 [cs.LG].

[29] Dua, D. and Graff, C., *UCI machine learning repository*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

[30] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H., "Feature selection: A data perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, p. 94, 2018.

[31] Guyon, I., Gunn, S., Ben-Hur, A., and Dror, G., "Result analysis of the nips 2003 feature selection challenge," vol. 17, Jan. 2004.

[32] Molina, L., Belanche, L., and Nebot, A., "Feature selection algorithms: A survey and experimental evaluation," pp. 306–313, 2002. DOI: 10.1109/ICDM.2002.1183917.

[33] Wang, A., An, N., Chen, G., Li, L., and Alterovitz, G., "Accelerating wrapper-based feature selection with k-nearest-neighbor," *Knowledge-Based Systems*, vol. 83, pp. 81–91, 2015, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2015.03.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705115001033.