UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

“Selección de Características basada en Mapas de Calor:
Un Algoritmo Multivariable para Datos de Altas Dimensionalidades”

Tesis que para obtener el grado de Doctor en Ciencias

Presenta:
M.C. Carlos Alberto Huertas Villegas

Bajo la dirección de:
Dr. J Reyes Juárez Ramírez

Co-dirigido por:
Dr. Christian Raymond

Tijuana, Baja California

Diciembre 2017

II

Acknowledgements

This section should be way easier in comparison with all the hard work that has been done for the past 5 years, but it is not..., it is not because it's very hard to include all the important people that made this possible.

It is highly probable that I will forget someone, sorry in advance.

I want to thank my family first, absolutely nothing like this would have been possible without their support, I owe every single of my achievements to them.

To my colleagues (and friends), in seat order, Chino, Alan, Angus, Sergio, Angeles, Paty, Sam, Reyes, we have shared so much in all these years, my achievements, even if not great, are the result of our collaboration, thanks again.

To all my friends that even space&time mismatches are always there to support, even if that support its just sending non-sense YouTube videos (Chuy), Thanks.

To my advisor, Reyes Juarez, its been 13 years now, yes... 13, it's incredible how we managed to do all this progress, it will never be enough to impress but I guess that's part of being a good advisor. Also special thanks to my co-advisor Christian Raymond who always challenged me to improve. Thanks.

To Mexico, it was not only the government (Conacyt) who financed my education, it was each and every hard-working Mexican, to all of you, thanks.

To you, reader, thanks for your time.

Resumen

Desde hace tres decadas, se ha visto el desarrollo de nuevas tecnologias en muchas areas de la investigacion, cada vez tenemos mejores y mas rapidos equipos de computo, asi como tambien sistemas de almacenamiento de informacion mas confiables, estas inovaciones han tenido un impacto directo en otras areas como la estadistica.

El analisis de la informacion se ha convertido en una tarea cada vez mas compleja debido a la explosion de informacion, este fenomeno se da debido al crecimiento exponencial de los datos que se recolectan cada periodo. Una de las areas afectadas es la inteligencia artificial, en particular, el aprendizaje maquina en donde la idea principal es crear un modelo que explique el fenomeno observado y sea capaz de generalizar un nuevos datos.

En un ambiente perfecto, el 100% de los datos que se utilicen para construir el modelo serian utiles, sin embargo, en la practica existen muchas otras variantes a considerar, por ejemplo, limitaciones de computo o memoria son importantes en aplicaciones de tiempo real, sin embargo, incluso cuando el poder de computo fuera vasto, incluir demasiadas variables en el modelo podria causar sobre-entramiento y no seria posible generalizar correctamente.

Con el objetivo de reducir el espacio de busqueda en los modelos, es que nacio el area de seleccion de caracteristicas, la idea es eliminar las variables que no aportan informacion util al modelo, dejando unicamente lo que apoye a la generalizacion. Hasta este dia, muchos enfoques se han desarrollado, sin embargo no existe un mejor algoritmo y multiples estudios independientes han demostrado que los resultados varian segun los datos.

Es de suma importancia continuar con el desarrollo de nuevas tecnicas por al menos dos razones, aumentar las posibilidades de encontrar el mejor subconjunto de variables al evaluar multiples algoritmos y segundo, crear nuevas metodologias que permitan estar a la par con el gran creimiento de informacion. Muchos algoritmos que en su momento fueron muy populares, en la actualidad ya no son utilizados debido a su orden de complejidad.

En esta tesis, se propone un algoritmo que consiste en comprimir el conjunto de datos previa seleccion, con esta tecnica es posible reducir el espacio de busqueda de cualquier conjunto de datos

en un 66% sin perdida en el poder de generalizacion. Multiples variantes del algoritmo se han implementado para abarcar un rango mas amplio de conjuntos de datos.

Los resultados obtenidos muestran que HmbFS (del ingles, Heat Map Based Feature Selection) es muy competitivo con otros algoritmos del estado del arte, mientras que mantiene un bajo nivel de consumo de memoria y una complejidad de orden lineal.

Abstract

Since three decades ago we have seen the development of new technologies in most areas of research, we have faster machines and better and more reliable storage, these enhancements have a direct effect on other areas such as statistics.

The analysis of data is becoming harder every time due to the information explosion, a phenomenon that arises due to the exponential growth of data being collected every year. One of the affected areas is artificial intelligence, in particular, Machine Learning, the main idea is to create a model that explains the observations but that is capable to generalize to new, unseen data.

In a perfect environment, the 100% of the data being fed to the model would help to improve the performance, however, in practice, there are many constraints to consider, limitations in memory and processing power are important for real-time applications, however, even if resources are vast, feeding the model with too much data in the form of variables to learn could produce a very narrow result caused by overfitting to the original data and not being able to generalize further.

In order to reduce the search space for the models, the area of feature selection was born, the idea is to remove unwanted useless information and build a model using only important data (features) to help to generealization performance. Many approaches have been developed, up to this day, there is no single best algorithm, and many indepedent research conclude that results vary from data characteristics.

It is important to continue with the development of new techniques for at least two reasons, increase the chances to get the best subset of features by trying multiple approaches and create fasters approaches that can keep up with the imminent increase in data side. Several algorithms that used to be popular and useful nowadays are rarely being used due to their order complexity.

In this thesis, a new technique is being proposed that compress data prior selection, with this enhancement its possible to reduce up to 66% the search space of any dataset with no loss of generality. Different variants of the selection algorithm have been developed to tackle multiple scenarios in datasets.

The results show that Heat Map Based Feature Selection (HmbFS) is a very competitive approach against other state-of-the-art approaches while keeping a low footprint and linear complexity.

Universidad Autónoma de Baja California

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

COORDINACIÓN DE POSGRADO E INVESTIGACIÓN

FOLIO No. 231

Tijuana, B. C., a 29 de noviembre de 2017

C. Carlos Alberto Huertas Villegas
Pasante de: Doctor en Ciencias
Presente

El tema de trabajo y/o tesis para su examen profesional, en la Opción TESIS

Es propuesto, por los C. Dres. J. Reyes Juárez Ramírez y Christian Raymond

Quienes serán los responsables de la calidad de trabajo que usted presente, referido al tema "Selección de Características basada en Mapas de Calor: Un Algoritmo Multivariable para Datos de Altas Dimensionalidades"

el cual deberá usted desarrollar, de acuerdo con el siguiente orden:

I.- INTRODUCCIÓN
II.- TRABAJO RELACIONADO
III.- SELECCIÓN DE VARIABLES BASADO EN MAPA DE CALOR
IV.- SELECCIÓN DE VARIABLE CON VALIDACIÓN CRUZADA
V.- ORDENAMIENTO DE VARIABLES UTILIZANDO MAPA DE CALOR
VI.- CONCLUSIONES Y TRABAJO FUTURO

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Dr. J. Reyes Juárez Ramírez
Director de Tesis

Dr. José Luis González Vázquez
Sub-Director Secretario

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

Dr. Christian Raymond
Co-Director de Tesis

Dr. Luis Enrique Palafox Maestre
Director

Contents

1 Introduction **1**

1.1 Machine Learning 1

1.2 Feature Selection 4

1.3 Heatmaps 7

1.4 Thesis Statement 10

2 Related Work **11**

2.1 Current common approaches 12

2.2 Open Issues 16

3 Heat Map Based Feature Selection **23**

3.0.1 Stage 1 - Compression 25

3.0.2 Stage 2 - Feature Selection 27

3.0.3 HmbFS Experiments 28

4 Heat Map Based Feature Selection with CV **51**

4.1 Introduction to HmbFScv 51

4.2 Experiments 54

5 Heat Map Based Feature Ranker **61**

5.1 Introduction to HmbFR 61

5.2 Experiments and Results 63

6 Conclusions **95**

6.1 Summary 95

6.2 Conclusions 96

6.3 Future Work 96

List of Figures

1.1 Most common example of Machine Learning 2
1.2 Binary Classification Example 2
1.3 Heatmap example 8
1.4 Late 1800's Shading Table 9
1.5 Mid 1900's Permuted Heatmap 9

2.1 Scalability behavior for top 5 algorithms 18
2.2 Scalability behavior for bottom 5 algorithms 19
2.3 Average Features Selected with top 5 algorithms 20
2.4 Average Features Selected with bottom 5 algorithms 20

3.1 HmbFS algorithm process overview: From data compression to final mapping. 25
3.2 PCA plots for different versions of Chowdary dataset. 48

4.1 Proposed architecture for optimal threshold selection 54
4.2 Comparison of best models for the Alon dataset 56
4.3 Comparison of best models for the Pomeroy dataset 57
4.4 Comparison of best models for the Chowdary dataset 58
4.5 Comparison of best models for the Tian dataset 59

5.1 Scalability behavior for class growth 65
5.2 Scalability behavior for samples growth 66
5.3 Scalability behavior for features growth 67
5.4 Carcinom Visualization 70
5.5 Carcinom reduced features SOM plane 71
5.6 Colon Visualization 72
5.7 Colon reduced features SOM plane 74
5.8 GLI-85 Visualization 75

5.9 GLI-85 reduced features SOM plane 76
5.10 Glioma Visualization 77
5.11 Glioma reduced features SOM plane 79
5.12 Leukemia Visualization 80
5.13 Leukemia reduced features SOM plane 82
5.14 Lung Visualization 83
5.15 Lung reduced features SOM plane 84
5.16 ORL Raws 10p Visualization 85
5.17 ORL reduced features SOM plane 87
5.18 Prostate GE Visualization 88
5.19 Prostate reduced features SOM plane 89
5.20 USPS Visualization 90
5.21 USPS reduced features SOM plane 91
5.22 Warp Pie 10p Visualization 92
5.23 Warp reduced features SOM plane 94

List of Tables

2.1 Algorithms Output Type 15
2.2 Algorithms variable handling 15
2.3 Datasets Characteristics 16
2.4 Classic algorithms evaluation 17

3.1 Datasets main attributes 29
3.2 Dimensionality reduction algorithms overall information 29
3.3 Classifiers used for testing 30
3.4 Alon dataset results 32
3.5 Burczynski dataset results 1/2 33
3.6 Burczynski dataset results 2/2 34
3.7 Chin dataset results 1/2 35
3.8 Chin dataset results 2/2 36
3.9 Chowdary dataset results 1/2 37
3.10 Chowdary dataset results 2/2 38
3.11 Singh dataset results 1/2 39
3.12 Singh dataset results 2/2 40
3.13 Su dataset results 1/2 42
3.14 Su dataset results 2/2 43
3.15 Tian dataset results 1/2 45
3.16 Tian dataset results 2/2 46
3.17 HmbFS performance comparison 47

4.1 Alon Dataset Accuracy(%) 55
4.2 Gravier Dataset Accuracy(%) 57
4.3 Chowdary Dataset Accuracy(%) 58
4.4 Tian Dataset Accuracy(%) 59

5.1 Test scenarios for algorithm order behavior 64
5.2 Summary of scale behavior 68
5.3 Benchmark datasets details 68
5.4 Classifiers setup 69
5.5 Logistic Regression for Carcinom 71
5.6 Random Forest for Colon 73
5.7 Logistic Regression for GLI-85 76
5.8 Random Forest for Glioma 78
5.9 Random Forest for Leukemia 81
5.10 Logistic Regression for Lung 83
5.11 Logistic Regression for ORL 86
5.12 Logistic Regression for Prostate-Ge 89
5.13 Naive Bayes for USPS 91
5.14 Naive Bayes for Warp Pie 10p 93

Chapter 1

Introduction

1.1 Machine Learning

The idea of machine learning (ML) is to have a entity capable of changing its internal structure based on external input, to improve its performance for a given task. However, as pointed out by Nilsson [1] a very important question might arise: Why not just build machines for that specific task in the first place?

One fundamental limitation for the proposed question is that although humans can have an overall understanding of a problem, they might not have a specific, well defined solution for it, if that were the case, then machine learning would not be required.

Given the previous premise we use ML for problems were no concise solution can be found, one of the reason a problem can not be explicitly defined by humans is the amount of data that needs to be processed, one of the most common examples nowadays is web ranking [2] as shown below:

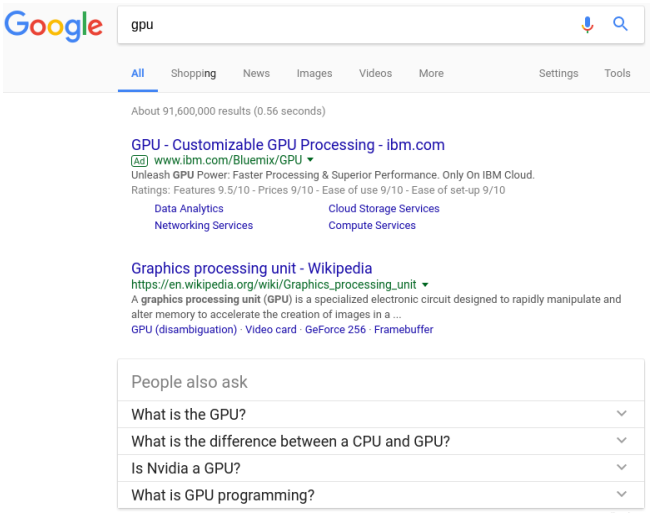

Figure 1.1: Most common example of Machine Learning

In order for the machine to provide useful resources given a query, it needs to *know* how associate the words and websites to rank them properly. Many more real life examples of ML are present in our lives nowadays, Amazon product rankings, Netflix movie recommendations, among others.

One of the most common tasks in ML is called *classification*, an everyday example happens with your email client, for instance in Gmail we have spam detection which classifies incoming messages as legit or scam/spam, this specific task is called binary classification as the outcome can only be one or the other. Next we show a visual representation:

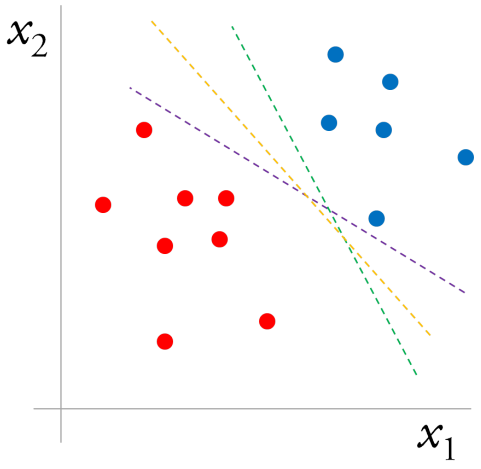

Figure 1.2: Binary Classification Example

As it can be seen in Figure 1.2, two different entities (red and blue dots), need to be classified, in

the example, we see a line that separates each dot, this line is a representation of a linear classifier, in other words, it groups each group o data given the position in reference to the line.

In order to be able to classify information, data need to be feed to the classifier, this information comes in form of random variables, one important aspect of these variables is their distributions, if the variable is discrete (i.e., it can take finite number of values) then we use a probably mass function (PMF) to describe them, such PMF must be non-negative and sum one. A common example is a coin toss, which outcomes can only be head or tails. this is described as:

$$P(X_{head}) = 0.5 \text{ and } P(X_{tails}) = 0.5 \tag{1.1}$$

However, if the random variable is continuous (i.e., an infinite number of possible values) the probabilities are defined by a probability density function (PDF), one of the most common distribution is the Gaussian (or normal) distribution which is defined as follows:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1.2}$$

Understanding the distribution and behaviour of a variable (or feature) is a key step in machine learning as noisy variables can hurt the learning process, one of the two key atributes to check for a feature is its mean and variance. The mean is expressed as follows:

$$\mathbb{E} = \int x dp(x) \tag{1.3}$$

This will help us to understand a likely value for a random variable, however this is usually never enough to understand its behaviour, as the variance will tell us how much this variable debiates from its mean, in other words how risky is to suddenly get a very different value, this is denoted as follows:

$$Var[X] = \mathbb{E}\left[(X - E[X])^2\right] \tag{1.4}$$

The importance of a feature does not only rely on this stability but in the case of machine learning problems in general we care about how a particular feature is related to the target in question, going back to the spam example, how a particular piece of information can help to determine if an email is or not spam. The problem of this relation feature-target can go even more complex as some features might be dependent of others, this is known as dependent random variables, and we care in particular about it conditional probability. For instance, What is the probability of going to the movie theather given if its or not weekend, this is described as $P(X = movies|Y = weekend)$ might be higher (or at least different) than $P(X = movies|Y = weekdays)$ more formally expressed, conditional probability es

denoted as follows:

$$p(x|y) = \frac{p(x,y)}{p(y)} \tag{1.5}$$

This is the origin of famous Bayes theorem [3] which describes the probability of an event based on prior knowledge of a feature that might be realted, for instance, cancer and smoking, using Bayes theorem the fact that someone is a smoker can be used to estimate the probability of developing cancer compared to a non-smoker.

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \tag{1.6}$$

This theorem can be used as a simple classifier, however machine learning in general is used for much bigger problems and we need more poweful classifiers to learn from a much bigger space of information (or features), this grow in complexity starts a whole sub-field in machine which tries to deal with all the information and determine what is useful to build a model. More about this is discussed in the next section.

1.2 Feature Selection

Feature Selection dates back since 1960 [4] and is one of the areas in machine learning that has received considerable attention by several researches, at some point there was doubts about the validity of this area [5] nonetheless, the research continued and it had an important boom starting in 1997 with special issues on its relevance [6] [7], however back then only a very minimal set of domains had more than 40 features [8].

The improved technology of recent era has made available measurements with very high resolution, hence there has been an exponential increase in data up to the point it has become unmanageable even for current algorithms as is the case with microarray research [9] where the data is of very high dimension and contains a lot of noise.

In order to determine which part of the information is really useful for the problem domain, different techniques has been proposed such as: automatic pattern recognition & knowledge discovery and data mining; however when the data dimension is too high, it is required to perform a pre-processing step to reduce such complexity and these approaches are known as Dimensionality Reduction, the two main techniques can be categorized as Feature Extraction and Feature Selection.

The Feature Extraction approach seeks to transform the high dimensional features into a whole new space of lower dimensionality, an example of these techniques could be Principal Component Analysis

(PCA) [10]. In this thesis we focus on the other approach which is Feature Selection, and it consists in reducing the dimensionality of the data by removing features that are noisy, redundant or irrelevant for a classifier.

Both approaches, Feature Extraction and Feature Selection, aims to have the smallest number of features [11] as experimentation has proved that a subset of features may work better than the entire set [12], hence reducing computation resources such as memory and processing, however in our opinion the Feature Selection approach is better as it keeps the original values after reduction and there is a perfect relation with the original data, as opposed to Feature Extraction where a transformation occurs and resultant data cannot be directly linked to the source.

Since 1980 we can see the development a lot a different algorithms for feature selection that have been successful in different areas such as: Text categorization [13], pattern recognition [14], image processing [15], bioinformatics [16], etc. As the number of algorithms increases, it becomes more difficult to select the right one for a given application, as researches suggests there is no such thing as best algorithm and the results depends on the characteristics and size of the data itself [17], another complexity to the problem is the called curse of dimensionality [18] which talks about a phenomena that occurs when analyzing data in high-dimensional spaces that does not occur in low-dimensional scenarios, hence the evaluation of large feature sets becomes unmanageable for some algorithms, making the removal of redundant and not relevant features a complex task. The key idea however remains the same for all those algorithms: ”try to keep the most relevant features and remove the rest”.

There are several proposals to define what a relevant feature is, however in this research we take the definition of John & Kohavi [19] which defines two different kind of relevance:

- Strong Relevance: are features that, if removed, would cause a drop in classifier performance.
- Weak Relevance: these features can be removed from the full set without considerably affecting the classifier performance.

Any feature that is not relevant is therefore irrelevant and could be divided into two main groups:

- Redundant Features: those that do not provide any new information about the class and therefore can be substituted by another feature.
- Noisy Features: includes the features that are not redundant but, do not provide useful information about the class either.

In the seek for these relevant features, there are at least 4 key parameters that affect the search performance [6]:

- Search Direction:
  - Forward: we start with an empty set of features and new ones are being added once they are evaluated as useful.
  - Backward: the full set of features are setup at first and then irrelevant features are being dropped.
- Search Strategy: These are based on corresponding heuristics for each algorithm, one example could be Greedy search.
- Evaluation Criterion: how the features are selected, the criteria or threshold that needs to be overcome in order to tag a feature as useful.
- Stopping Criterion: this determines if the algorithm will just stop after a given number of iterations or will hold until a given threshold could be reached, this has a direct impact on the feature set size, where the optimal size may be defined by the number of instances [20].

When designing a feature selection algorithm there are at least three challenges to address [21]:

- Small sample size: as the number of features increases, the number of instances required to produce a stable selection increase as well, however, there are many areas where samples are expensive to gather and the result is a dataset with thousands of features with only hundreds of samples which makes the process prone to overfitting.
- Data noise: if the algorithm is not robust enough, noisy features would guide the selection erroneously as the algorithm will try to learn the noise as if it were a real pattern.
- Selected Features: algorithms that only rank features have to deal with the issue of how many features to choose for the final subset. Some other approaches use thresholds to automatically select the number of features.

Supervised feature selection algorithms use the statistical relation between the data and the labels [8] to select the features. These algorithms can be divided into three major groups: Filter, Wrapper, and Hybrid [21]. For low-dimensional data, it is possible to use different techniques, even the wrapper approach could be used, but once the dimension grows at a considerable scale of thousand of features, the computation complexity becomes a problem, and the filter approach, being the more efficient becomes a very popular option. Our algorithm is a filter approach.

- Filter Model: these algorithms are completely independent of any classifier, hence the final selection depends only on the characteristics of the data itself and its relation to the target [22, 23]. Filter models are overall very efficient, scalable and their results are usually portable as they do not depend on external components such a guiding classifier, these advantages make them very suitable for high dimensional data.

- Wrapper Model: these algorithms use a classifier to evaluate their performance [7, 19], the initial feature selection is usually achieved by a greedy search strategy, the classifier performance is evaluated with the selected features and if the result is satisfactory the search stops, otherwise the whole process is repeated again but this time with a different subset. This is a very expensive process usually applied for low-dimensional data only.

- Hybrid Model: As we have seen the filter models are more efficient, while the wrapper models are usually more accurate, in order to achieve a balance, the hybrid model is proposed to fill the gap between those approaches [24]. In the search step they employ a filter selection to reduce the number of candidates, later a classifier is used to select the subset that provided the best accuracy.

For low-dimensional data, it is possible to use different techniques, even the wrapper approach could be used, but once the dimension grows at a considerable scale as thousand of features, the computation complexity becomes a problem and the filter approach being the more efficient becomes most of the time the only feasible option.

1.3 Heatmaps

One key element in this work is the application of Heat maps which main idea is to help to visualize any possible relations between a row (for our case, instances) and a column (features, or variables). Thanks to our current technology in high-resolutions displays, it is possible to plot millions of interactions in a single consumer screen.

Due to its practical application, according to Weistein [25], a heatmap is by far the most popular graphical representation choice in many sciences, including biological. Even if this approach was introduced two decade ago, thousands of publications still use them.

A tipical heatmap is shown in Fig 1.3

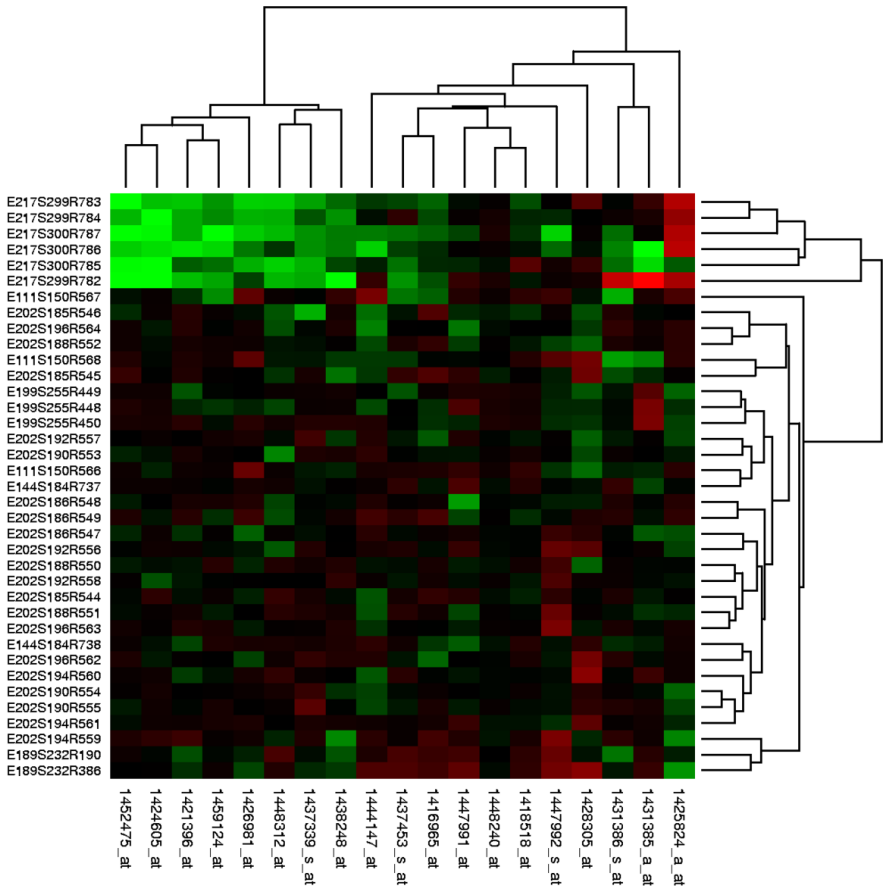

Figure 1.3: Heatmap example

As in can be seen in Fig 1.3, even without the original data, it is easy spot patterns that otherwise would require exhaustive search, for instance, to detect that the top-left part is clearly different.

The key element in a heat map is indeed difference in color representation which is a technique to organize information way older than the heat-map itself, one of the earliest approaches of this can be found in the work of Loua [26] in 1873, in Fig 1.4 we show the shading table that was described in his work.

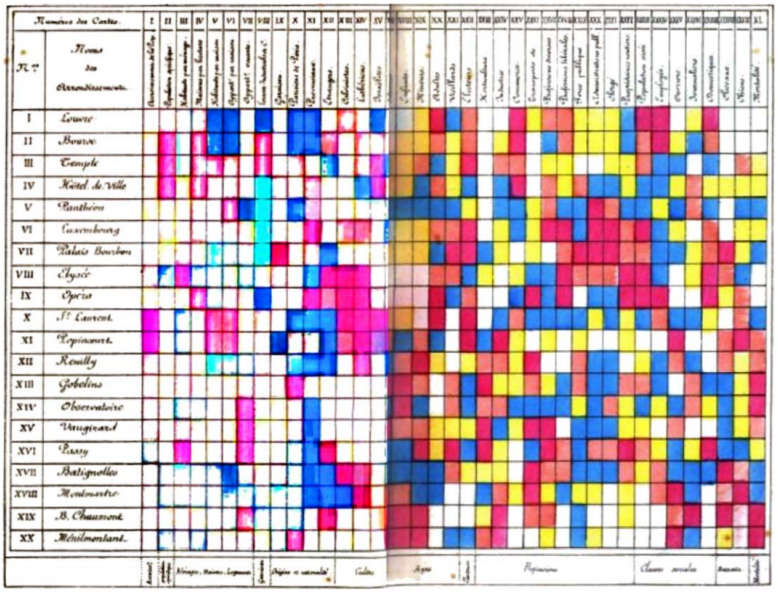

Figure 1.4: Late 1800's Shading Table

Since very early works, it can be seen the usefulness of heatmaps as a way to represent patters in data, however, it is also possible to discover new patters than the intended by the original heatmap author, this idea can be seen exploited in Bertin [27] work as shown below.

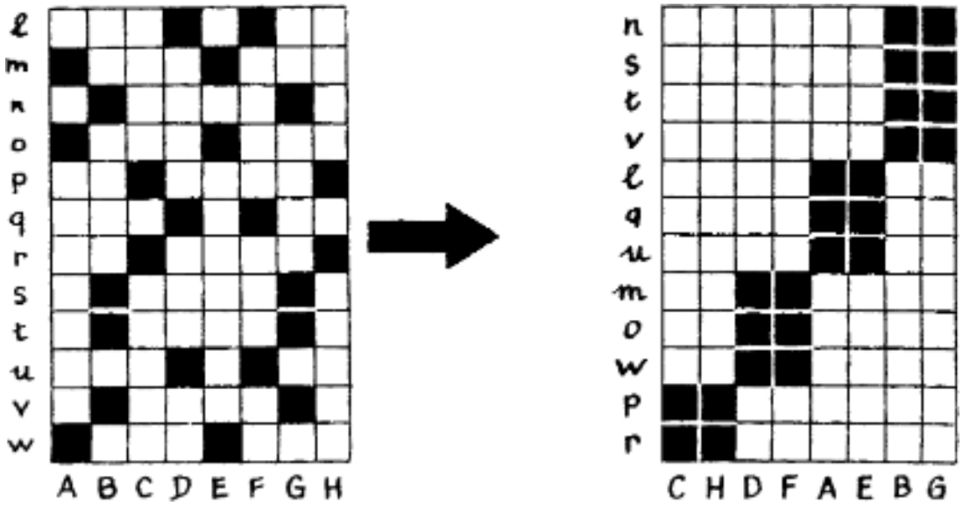

Figure 1.5: Mid 1900's Permuted Heatmap

From Fig 1.5 we can see how it is possible to discover new patters if there is a hidden structure in the data, this is the key element in this work and in this thesis we show how to use this approach in the field of feature selection. In the next chapter, we discuss about related works and how they perform with

specific interest in how they scale as the number of features we have nowadays is constantly increasing putting extra challenges in classification algorithms.

1.4 Thesis Statement

This thesis claims that feature selection in a supervised machine learning task can be optimized for high-dimensional data. In particular, this thesis investigates the following hypothesis:

"*In high-dimensional spaces, features importance can be estimated by the relation they hold with adjacent variables.*"

In order to evaluate the above hypothesis, an algorithm that compress data by constructing new features based on adjacent variables has been developed. Different versions of this algorithm are discussed across the thesis, each version handles a particular problematic, however, all of them work under the same premise, and all of them are used to test the hypothesis.

Heat Map Based Feature Selection (HmbFS) is an algorithm that can reduce any dataset to 33% of its original space thanks to its built-in compression of adjacent features. As with other filter algorithms, no external dependencies are required making it suitable for large datasets without performance compromise.

Our experiments show that our approach is very competitive or even better than other current techniques without requiring to perform univariate analysis, doing all the processing with the self-constructed new set of features that can be later be mapped to the original space for interpretability.

Chapter 2

Related Work

When working with high dimensional data scalability plays a very important role, some algorithms are only suitable for low-dimension spaces even if they are good solutions their computation complexity does not allow its usage. As we continue to review more works related with high-dimension, we will find most works working as a filter approach for medical applications as they present a complex scenario and finding useful features can lead to powerful insights to understand a medical condition (known as biomarkers).

In 2005, [28] used a bayesian neural network to identify biomarkers, using a binning approach as pre-preocessing they managed to remove almost 97% of the data in the NCI dataset, hence reducing the complexity their approach needed to tackle which lead to good results according to their experiments.

In 2006, [29] demonstrated how the algorithms of Bonferroni [30] and Westfall&Young [31] could be used as feature selection for biomarker discovery, although experiments were carried with low resolution data only and it appears unclear as to how they would handle higher resolutions.

In 2009, [32] proposes a new application of wavelet feature extraction that in conjunction with Support Vector Machine (SVM) can be used for biomarker detection and therefore improve disease classification. In this work we use SVM not only as a classification algorithm but as a feature selection as well to have a comparison reference as how useful SVM based techniques can be.

In 2010, [33], used NCI low-resolution ovarian dataset and proposed a sparse representation based FS for MS data. Authors managed to remove around 68% of the data by limiting the m/z range from 1,500 to 10,000; this decision is based on the rationale that points lower than 1,500 are distorted by those of energy-absorbing molecules (EAM). Further reduction is achieved by baseline correction using local linear regression [34] and normalization. Although results are promising, it would be very interesting to find out if the machine alone is capable to decide which spectrum range to use.

In 2010, [35], present their proposal of using multiple instances of SVM-RFE, each trained in a

different subsample of the data to later be assembled in order to produce a more robust feature selection, they perform experiments on four datasets and three of them are including in our current work.

In 2011, [36], presented an approach that tries to diminish a problem caused by univariate analysis where no interactions between features are considered, in their proposed methodology each feature must first pass series of significance tests before being considered for further relation analysis. In their results they report an improvement over RFE with SVM, however the fact that the number of features to select need to be set in advance presents a drawback in most scenarios.

In 2013, [37], presented a genetic programming (GP) approach for FS in MS data. The key idea in this work is to combine two well known FS algorithms Information-Gain (IG) [38] and on the other hand Relief-F [39] and use their outputs as terminals for the GP method. However the process is very exhaustive as the GP method is designed in a wrapper approach. Good results can be achieved with this technique, but the computational cost of any wrapper approach needs to be considered.

In 2014, [40], proposed an algorithm called eTAFS that combines a simulated annealing and incremental joint entropy to select subsets of features. According to the authors, the proposed combination of techniques is more accurate than competing methods while being fast, effective and requires no critical parameters to tune.

2.1 Current common approaches

By reviewing literature, we can find many different algorithms, some algorithms propose very interesting ideas, however, as technology advances and more data is being collected, some of this algorithms can become infeasible solutions as we will discuss in this chapter. Using the research repository from Arizona State University by Zhao et. al [41] and [42] project we can identify some basic well known algoritms described below:

CFS The Correlation-based feature selection (CFS) uses heuristics to evaluate the worth of merit of subsets of features, such heuristics take in consideration how useful is a given feature to classify a class. In order to be able to perform the selection, CFS requires all features to be of the same type, hence a discretization pre-processing step is required [43]. The core of this algorithm is based on Pearson's correlation coefficient, which is a measure of linear dependency (correlation) between two variables X and Y, developed by Karl Pearson based on the idea of Francis Galton who discovered it in 1888 [44].

Chi2 The Chi Square (Chi2) algorithm is based on Kerbers ChiMerge [45] which is used to discretize numeric attributes based on the X2 statistic. The Chi2 approach aims to solve ChiMerge limitation

of finding the optimal significance level automatically, the way this is accomplished is by wrapping ChiMerge in a loop that automatically play with different X2 thresholds. A consistency checking is used as the stopping criteria and thus completing the automatic parameter selection which represents Chi2 phase 1. The phase 2 starts with the results obtained in phase 1, each attribute is associated with a significance level and the merging step begins, if the consistency rate is not archived for the i-attribute, the parameters are tuned and the attributes wait for a next merging round. The phase 2 continues until no more attributes can be merged [46].

FCBF The Fast Correlation-Based Filter (FCBF) solution tries to find the best feature subset based on goodness of features. In general to FCBF a good feature is one that is relevant to the class concept and not redundant to any other features, hence a correlation between variables is used to determine feature goodness [47]. The two main approaches to measure correlation are based on classical linear correlation and the other is based on information theory, however it is not safe to assume that in real world data there will be always a linear correlation between features, to overcome that problem FCBF uses an approach based on the information theory concept of entropy [30] [47].

Fisher The Fisher Score is a univariate algorithm that selects features that assign similar values to the sample of the same class and different values to samples from different classes [48]. The evaluation criteria according to He et al. [49] is a special case of Laplacian Score, however since Fisher Score evaluates each feature individually it is unable to handle any redundancy between them.

Gini The Gini Index [50] is a univariate algorithm that quantifies how easily a feature alone can distinguish between classes, the smaller the value the more related a feature is related to a class, so in this algorithm the top k-features with the smallest values are selected. As with other feature weighting algorithms, the k number of features to be selected needs to be set manually. One particular downside of this algorithm is the inability to deal with redundancy.

InfoG The Information Gain (InfoG) is one of the most popular techniques as it is very easy to interpret and compute. InfoG measures the reduction in the entropy of X that is caused after observing a random variable Y [51]. Since this evaluation is univarate it does not deal with redundancy.

KW The Kruskal-Wallis algorithm is non-parametric, therefore no assumption about the distribution of the data is being done [52]. The measures are done by comparing the population medians among groups, to do that a ranking initial step is required to merge all groups in a common scale. As with other univariate algorithms, no redundancy reduction is possible.

mRmR The Minimum-Redundancy Maximum-Relevance (mRmR) algorithm is based on mutual information. Given two random variables X and Y, their mutual information is defined in terms of their probabilistic density functions. In Max-Relevance, the idea is to select features Xi with the largest mutual information over the target class C, however it has been recognized that the combination of individual good features do not necessary lead to good classification performance [53]; mRmR deals with this problem incorporating the Minimum-Redundancy factor, where features that have been already selected as relevant, are evaluated to find out the dependency between them, if they are found to be highly dependent the less relevant feature is eliminated. Since mRmR is a feature ranking algorithm, it does not need to select a subset of features but instead the user is required to choose among the best ranked features.

RelieF The original Relief algorithm was proposed by Kira and Rendell [54], which was known for being very efficient, the main idea behind Relief is to estimate features according to how well their values distinguish among instances that are near each other, such instances are called neighbors. Relief looks for the two nearest instances, one of the same class labeled as nearest-hit and one of the other class labeled nearest-miss. The rationale behind the algorithm is that good features should have the same values for instances of the same class and different values for a different class. For discrete features the differences is either 1 (they are different) or 0 (they are equal) while for continuous data, it is the actual difference normalized to the [0-1] interval.

Relief-F is an extension to the original Relief where, instead of finding one near-miss M from a different class, the algorithm finds one near-miss M(C) for each class and averaged their contribution, the results are averaged with the prior probability of each class. Besides, to deal with the original Relief noise problems, Relief-F uses a user configured k-value for the number of neighbors to search.

t-T The t-test score is usually employed for binary classification problems [55], and comes particularly useful for unequal sample size and unequal variance. The t-test algorithm ranks features according to its capability to separate classes, ranking occurs with single features hence no redundancy reduction is possible.

From the reviewed algorithms we can find some key differences and similitude, as a summary we can say that all seeks for the same idea, which is: ”classes of the same type looks similar, while different classes should look different”. The main difference would be the way each algorithm computes such similarity. We can distinguish too different ways of presenting results, some algorithms only ranks features (feature weighting) and later a further selection needs to be done to select subsets from the ranking result, while other algorithms are capable of completely remove features (feature set) leaving a dataset with only the features that help discriminate the data. In Table 2.1 we show the algorithms in

terms of their output type.

Table 2.1: Algorithms Output Type

Feature Weighting	Feature Subset
Chi Square	CFS
Fisher Score	FCBF
Gini Index	
Information Gain	
Kruskal-Wallis	
mRmR	
Relief-F	
t-Test	

We can notice that there is a very disproportion between feature subset and feature weighting algorithms, one potential reason for this is that subset algorithms are more complex to design but have the advantage that no necessary additional step is required as with weighting that a very computational intensive step needs to be performed to select the right subset of features.

Another key element to classify algorithms would be the ability to evaluate combinations of multiple features, known as multivariate, grouping is shown next.

Table 2.2: Algorithms variable handling

Univariate	Multivariate
Chi Square	CFS
Fisher Score	FCBF
Gini Index	mRmR
Information Gain	
Kruskal-Wallis	
Relief-F	
t-Test	

From Table 2.2 we can see the same pattern, as multivariate design is more complex, the vast majority of algorithms fall in the univariate category. In theory, multivariate design should produce more compact results as they handle redundancy. In the next section we discuss how the increase in dimensions affect algoritms and why is important to keep developing new theories and implementations to keep up with technology advancements.

2.2 Open Issues

In this section we review the algorithms previously presented to understand the dynamics of how feature selection algorithm are evaluated and how they behave with dimensionality increase which let us understand common open issues in the area.

It is imperative to recreate the evaluation scenarios for all algorithms since most of them have been presented with very low-dimensional data, e.g. the FCBF algorithm proposed by Lei Yu [47] shows a remarkable comparison between FCBF, CorrSF, Relief-F and ConsSF algorithms evaluated against 10 different datasets, however their mean dimensionality size was 220, having datasets as small as 57 features and being the largest only 650 features. With the advances in data gathering, we can test the algorithms again but now with datasets more than 150 times larger such as microarray and biological data.

The data used for review is described below:

Table 2.3: Datasets Characteristics

Dataset	Type	Features	Instances	Classes
Arp	Image	2,400	130	10
Tox	MicroArray	5,748	171	4
Cll-sub	MicroArray	11,340	111	3
Smk-can	MicroArray	19,993	187	2
Gla-bra	MicroArray	49,151	180	4
Dorothea	Bio-data	100,000	800	2

To test each algorithm we utilize the ASU Feature Selection Algorithm Repository [41] which in turn uses implementations from Weka project [42], besides we have included a basic random feature selection algorithm to have a reference point in datasets that are too complex to process and using the full set of features is not feasible.

Since we are going to test algorithms that produce different type of results as shown in Table 2.1, we have the following two cases:

Feature Weight these algorithms rank the features instead of returning a list of features. In order to have meaningful results we run an iterative process to find a suitable subset of features. We start with the 10 best ranked features, and constantly increment features by 10 up to reaching 400.

Feature Set these algorithms provide ready to use results, so we proceed to test the quality of result with the returned sub-set of features.

In order to evaluate the quality of the selected features, we evaluate the classifiers accuracy obtained after the feature selection has been performed. For each dataset we randomly selected a fixed number of instances (60) for training in order to keep grow in the number of features only. The whole process is repeated 10 times using different parts of dataset each iteration. The classifiers used for comparison are SVM, J48 and Naives Bayes. Analysis in execution time grow rate is provided as well as the ability of reduce feature space.

In Table 2.4, we present the results in terms of accuracy for each classifier, the reported number represents the average value from all test and under all classifiers. Two specific scenarios might arise:

Non-trivial improvement expressed in **bold**, indicate the algorithm improve the accuracy obtained when the full set of features or the random sub-set selection is used.

Out of Memory (OM) since we are using a limit of 16GB for these datasets, this case might occur if the feature selection algorithm is poorly handling all the features and runs out of memory.

No Feasible (NF) means the given algorithm requires so much time that it falls out of practical usage.

Table 2.4: Classic algorithms evaluation

Data	Full	CFS	Chi2	Fcbf	Fisher	Gini	InfoG	KW	MrMr	ReliefF	tT
Arp	64.3	63.9	**67.7**	60.1	**68.4**	45.8	**68.1**	49.2	54.3	**69.0**	**69.3**
Tox	62.3	**63.3**	60.6	**63.6**	61.4	55.4	59.7	60.1	59.9	**62.7**	61.9
Cll	69.8	**NF**	67.6	69.8	62.4	65.4	67.5	65.5	67.2	**73.3**	63.8
Smk	**OM**	**NF**	**65.2**	**61.4**	**66.3**	**62.5**	**64.4**	**64.0**	**61.5**	**66.3**	**63.3**
Gla	**OM**	**NF**	**64.3**	**63.2**	**63.6**	60.0	**63.6**	**62.9**	**63.1**	**64.9**	61.9
Doro	**OM**	**NF**	**92.9**	**92.0**	**92.9**	**NF**	**92.9**	90.0	**93.1**	**NF**	90.2

From accuracy results in 2.4 can notice the necessity of feature selection. In low-dimensional problems it can be seen as a method to improve accuracy, but in high-dimensional it serves another purpose as well, which is reduce the computation complexity of a problem that otherwise would be unfeasible to manage, we can notice this effect as it was not possible to train the classifiers using the full set of features once we get closer to the 20,000 mark.

Some of the algorithms showed their scalability weakness as the number of features dramatically increased to 100,000; for instance CFS, Gini and Relief-F were unable to complete the experiments. On the other hand algorithms such as KW and mRmR were able to complete the task but did not get much encouraging results, sometimes loosing even more than 10.0% accuracy compared with the full dataset.

Results can be divided in two groups, full-set feasible and full-set not feasible, being the first 3 data sets in the feasible group, this section is of particular interest as most algorithms easily beat a random selection scenario, however when compared with the full number of features they seem to be filtering too much of information causing a decreas in classifier accuracy, e.g. with the Arp dataset, in 50% of cases, the usage of an algorithm result in a drop in accuracy, with the Tox dataset the problem rise to 70%, up to reaching the point where in the Cllsub dataset only one algorithm managed to improve accuracy.

Notable remark is that the top performing 3 algorithms (Chi2, InfoG, Fisher) are of feature weighting type, leaving FCBF to 5th place and the only remaining feature subset algorithm (CFS) not even completing the test. However feature weighting algorithms carry a computational expensive process to select a suitable group of features.

In Figure 2.1 & 2, we provide a summary of execution times composed by feature select/rank time, plus in the case of feature weighting algorithms, the extra step of finding feature subset to understand more clearly the algorithm grow rate. The quality of the algorithm programming is out of the scope of this paper; however implementations are done by Weka project which is a very well known tool in machine learning. (Note: since execution times for Chi2 and InfoG diverged less than 1% we have merged their results in C2&IG)

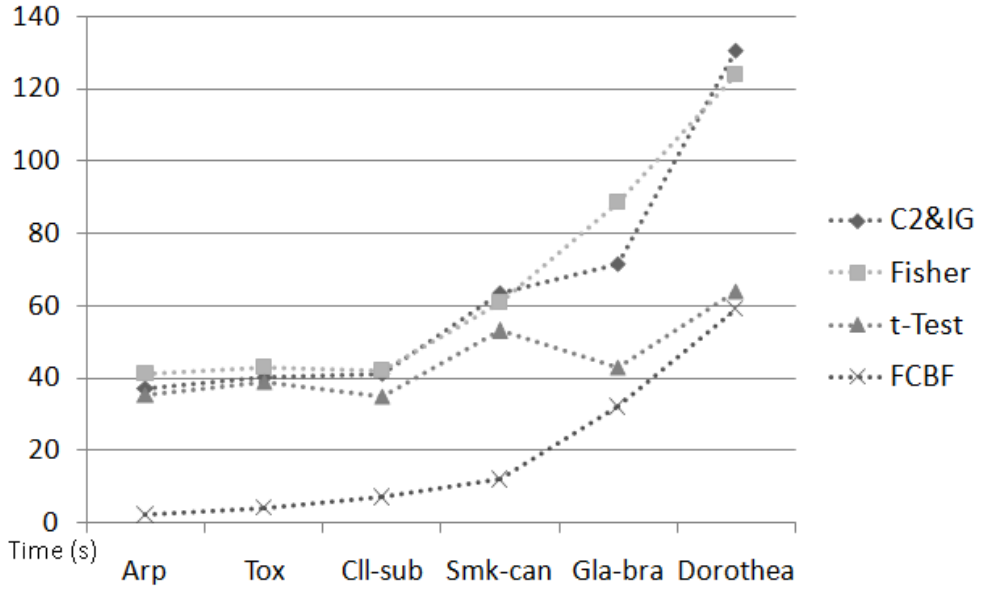

Figure 2.1: Scalability behavior for top 5 algorithms

For the bottom 5 algorithms, the scalability scenario is even worse as we can see in 2.2

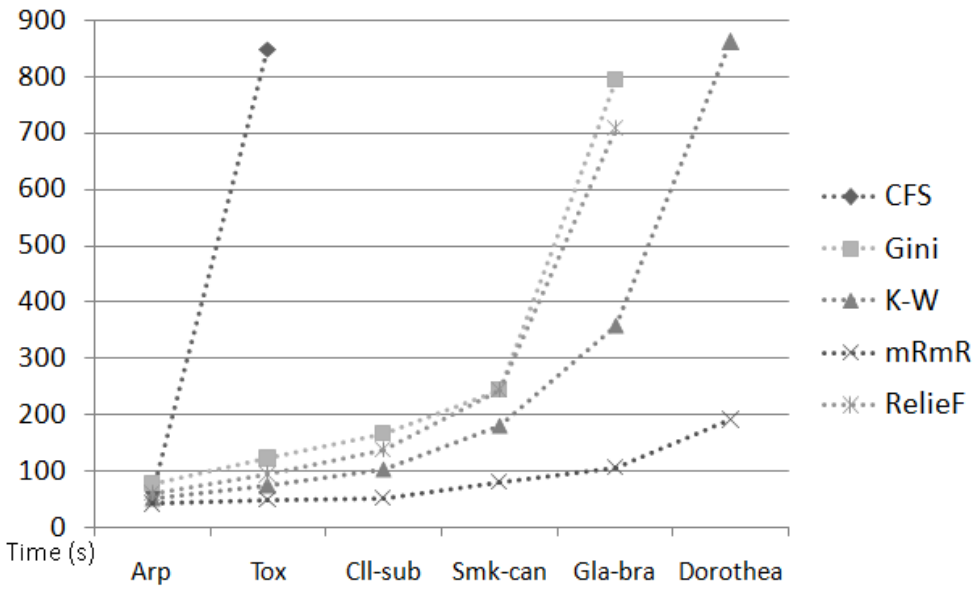

Figure 2.2: Scalability behavior for bottom 5 algorithms

Given the previous plots we can notice that in terms of performance, feature weighting have a penalty, such penalty is in function on how large we define the subset seek space, as mentioned before, we stopped at 400 features, but the larger the space the more pronounced the computation complexity. Particularly from Fig. 2.2 we can notice how unfeasible can become the usage of some of the algorithms as the dimension increases, for instance the K-W algorithm that in the first dataset perform just in average time, in the last test it required more than 14 times the processing power compared with the fastest algorithm overall (FCBF). Not included in figures but as a reference point training the classifiers with the full number of features took on average 9.4, 26.2 and 58.8 seconds for the first three datasets while providing very good results as shown in Table 2.4.

One particular problem with feature weighting is that there is no easy way to determine the appropriate feature subset space, we could have just stopped at best 200 ranked features however that would have caused a drop in accuracy, nevertheless we cannot know if a potential best accuracy could have been achieved with any greater number of features as seeking among all possibilities is just not feasible.

To understand how the algorithms helped to reduce dimensionality we show the average number of features selected that archived best overall accuracy among the 3 classifiers (SVM, Bayes, J48) for each dataset. Results are provided in the following plots.

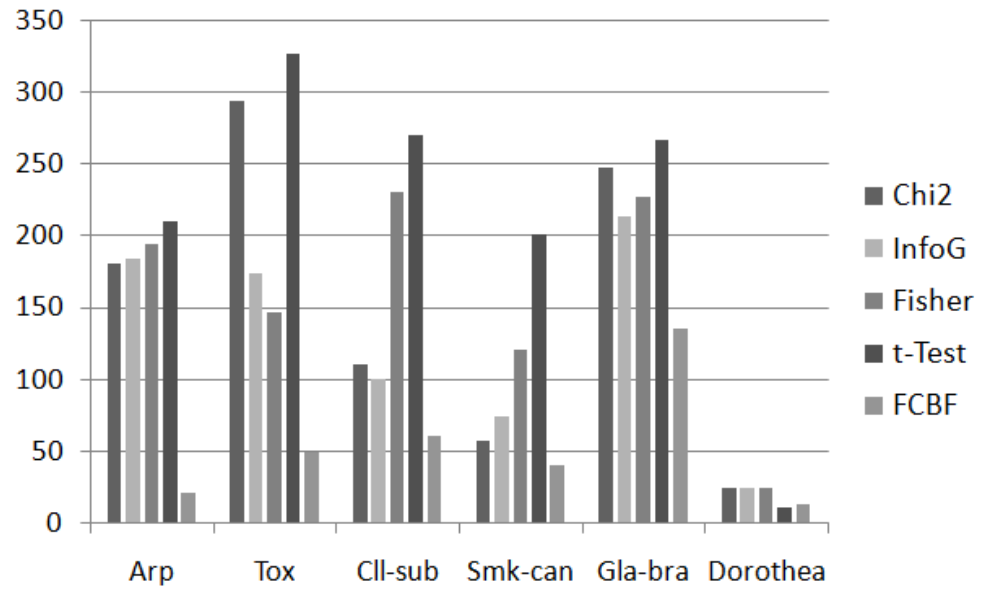

Figure 2.3: Average Features Selected with top 5 algorithms

In the next figure we present the same statistics but for the rest of algorithms.

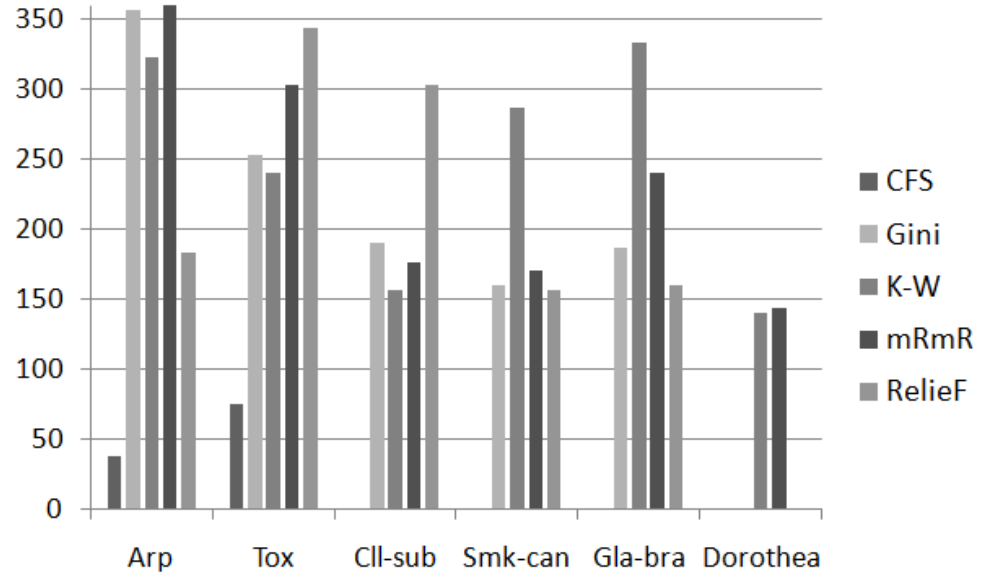

Figure 2.4: Average Features Selected with bottom 5 algorithms

After analyzing the number of selected features it is clear that the subset algorithms (CFS and FCBF) produces much more compact results and such results are portable to different classifiers. A different story applies with the feature ranking as results are notably less compact, e.g., the best overall

algorithm was Chi2 which average features across all dataset was around 152 while FCBF achieved a very similar accuracy ratio with a third of such dimension at 52 features average.

The feature selection problem is far from solved even when there has been work on it for more than 50 years. One particular problem in this area is that the term ”high-dimensional” is very subjective and in constant change, according to our results the fastest algorithm is FCBF, however, when first introduced [47] it was tested on high-dimensional data that these days could be considered very-low.

Information explosion has caused that the perception of high-dimension is constantly updated, in the review we just presented we have a mean of more than 30,000 features but a quick review in machine learning repositories such as UCI [56] reveal that the grow in information is advancing at a very high rate where we can now find datasets with millions of features, which makes no hard to believe that 100,000 features be considered low-dimension in a few years.

We can notice that 80% of our reviewed algorithms are of weighting type and that 70% of them perform univariate analysis, such trend in design seems to be simpler, but comes at a cost in performance as can be seen specially in Fig 2.1 with the first three datasets where FCBF (a subset type) performed more than 10 times faster than the best and fastest weighting-type algorithms, however FCBF being a multivariate algorithm showed one of the poorest scalability behaviors matching times with fastest weighting-type algorithm (t-Test) in last dataset. Nevertheless Fig. 2.2 shows an even less encouraging results, where 3 algorithms (CFS, Gini and ReliefF) were not even able to complete the experiments, a notable mention would be CFS (the only remaining subset type) which processing time increased 14 times when features increased only at double.

It is clear that there is no such thing as the final algorithm that always performs the best, one particular algorithm could perform better on some dataset than others and vice versa, two notable examples would be Chi2, which managed to have the highest average accuracy but did not performed the best on any of the datasets, and mRmR which felt to the 6th place but in Dorothea dataset got the best result.

A variety of algorithms is a good thing as it increases the possibilities to improve accuracy by trying different techniques, however for filter algorithms there is still a gap to fill as most algorithms fall in the feature weighting scheme and experimentation shows that subset provides much more compact and reusable results that can be easily utilized with different classifiers.

In this thesis, we present a proposal for a new feature selection algorithm that can scale in a linear fashion by taking advantage of a newly developed compression mechanism for high dimensional data. In the next chapter, we discuss in depth the algorithm design and validation

Chapter 3

Heat Map Based Feature Selection

HmbFS stands for Heat Map based Feature Selection and its an algorithm that takes advantage of the intrinsic order in high dimensional data to build a heat map that based on the data, all processing is evaluated in this map and then the results mapped to the original data. In the following sections, we formalize this approach.

The context of this work is focused on supervised machine learning, to formally express the place that feature selection (FS) takes in the learning process, we first must define the supervised learning process itself.

In order to build a model to solve a problem, for supervised learning it is required to have a set of data known as training set, this data is composed by a set of labeled (e.g. cancer vs normal) instances which are features vectors containing the information that is required to learn.

An instance is therefore formalized as an assignment of values $V = (v_1, ..., v_n)$ to a set of features $F = (f_1, ..., f_n)$, where each instance is labeled with at least one of the possible $c_1, ..., c_n$ classes of C. In the case of MS data, each possible value in the m/z spectrum is a feature, and the specific m/z concentration is the value assigned to that feature; later each instance is labeled with the condition they represent.

Since the training set is the input to build a classifier, it is clear that its performance is inherently dependent of the quality of such features. And as pointed out by [57], in theory, the more features we have, the more power to discriminate between classes we would have, however, in practice with a limited number of instances (as is the case of MS data) the excessive number of features not only causes the learning process to be slow but there is a high risk of overfitting the data, as irrelevant or noisy features may confuse the learning algorithm. To handle this problem is why feature selection is applied.

To formalize the concept of FS, let R be a reduced (subset) version of F and V_R the value vector for

R. So in general, FS can be defined [58] as finding the minimum subset R such as $Pr(C \mid R = V_R)$ is equal or close to $Pr(C \mid F = V)$, or in other words, if the probability distribution of different classes given the reduced subset is equal or close to the original distribution for the whole features in F.

Since our goal is to reduce the original space, it is required to identify such features that are relevant to the learning process and discard the rest. [59] propose a relevance categorization as follows: Let F be the full set of features, F_i a particular feature and $S_i = F - \{F_i\}$ a subset of features where F_i has been removed. Now we can identify three types of features, strongly relevant, weakly relevant and irrelevant given the following conditions:

A feature F_i is strongly relevant iff: $Pr(C \mid F_i, S_i) \neq Pr(C \mid S_i)$

A feature F_i is weakly relevant iff: $Pr(C \mid F_i, S_i) = Pr(C \mid S_i)$ and $\exists S'_i \subset S_i$ such as $Pr(C \mid F_i, S'_i) \neq Pr(C \mid S'_i)$

A feature F_i is irrelevant iff: $\forall S'_i \subset S_i \quad Pr(C \mid F_i, S'_i) = Pr(C \mid S'_i)$

Given these definitions, we have that a feature is strongly relevant if it is always required in order to keep or improve the discriminatory power of the optimal subset of features. A weakly relevant feature may or may not be required, however, under some circumstances it helps to contribute to the classes discrimination. Then an irrelevant feature is simply not required and its removal causes no impact at all in class discrimination power.

Our proposal hence consist in finding such relevant features in an automatic way employing our algorithm called Heat Map Based Feature Selection (HmbFS) which has been designed to work with very high dimensional datasets (such as MS data) with very low memory footprint thanks to a data compression mechanism.

The process is represented with a cancer problem in Fig 3.1 to show the two main stages: 1) Compression and 2) Selection in our approach.

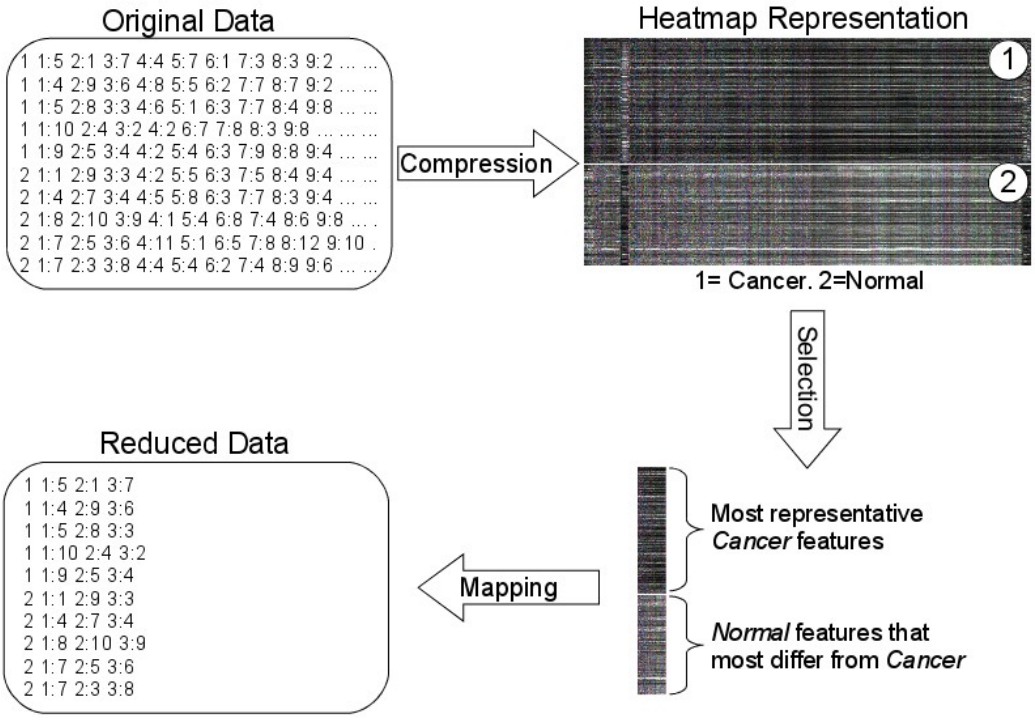

Figure 3.1: HmbFS algorithm process overview: From data compression to final mapping.

In the following sections to discuss in detail each stage.

3.0.1 Stage 1 - Compression

In order to handle the analysis of hundred thousands of features with efficiency, a key factor that takes our proposal apart from others is that we do not perform single feature analysis but instead we join features in groups, hence reducing the number of required iterations to compare the usefulness of features. Therefore our relevance definition had a slight change as we do not analyze an individual feature F_i anymore but instead a group G_i that is defined as $G_i = \{F_r, F_g, F_b\}$, these elements stand for $Feature_{Red}$, $Feature_{Green}$ and $Feature_{Blue}$ and their corresponding values represent color intensities, but together (i.e., as a group) the relation between them creates a true color. In other words, the relation between a particular group of features creates a pattern, such pattern is mapped to a color in a virtual heatmap and this color is what is further analyzed in order to decide if the group of features should be kept. These groups are built by continuous features, so in the original space F_1, F_2 and F_3 becomes F_1 in the compressed space.

The first step for compression is the normalization of each isolated feature at a time, i.e. each feature is treated independently, to formalize this, let I be the full set of instances (e.g. samples in MS

data) and I_x a particular sample for the whole dataset D, each value for F_i that belongs to a instance I_x (I_xF_i) is then normalized to a 0 to 255 interval as shown:

$$\forall I_xF_i \in D : \\ \widehat{I_xF_i} = \left(\frac{I_xF_i - min(FI_x)}{max(FI_x) - min(FI_x)} \right) * 255 \tag{3.1}$$

Where min(IF_i) and max(IF_i) represents the minimum and maximum value respectively that a feature gets across all the instances of the dataset. This process is repeated for every feature F_i across all instances I_x. Once the features have been normalized, each of the groups containing F_r, F_g and F_b can be mapped to a true color expressed in red/green/blue (RGB) format, together the three features can represent up to 16,777,216 different colors or patterns. However, since the idea is to build a generalization of the original data, we apply a technique called *Color Quantization* that allows the mapping of a true color to a lower depth color scale, in this case we reduce each true color to a 4-Bit 16-Colors scheme. The idea is to discard small data variations or uniqueness, and produce a new set of data which is more general and consistent, the process of quantization is performed by finding the minimum euclidean distance between the true color and the 16 reference colors, ranging from $R_{j=1}$ to $R_{j=16}$, the RGB values for each of those colors are the standard values defined in the HTML 4.01 specification [1].

To formalize the information quantization, let G be the set of all the built groups and G_i a particular group (which is composed by F_r, F_g and F_b) that belongs to a instance I_x, each of those I_xG_i combinations are then compared against all the reference colors R_j in set R to produce the new single-value compressed feature F_i that will represent the original 3-feature as shown below:

$$\forall I_xG_i \in D \; and \; \forall j \in \{1, ..., 16\} : \\ I_xF_i = min \left(\sqrt{(I_xG_iF_r - R_jF_r)^2 + (I_xG_iF_g - R_jF_g)^2 + (I_xG_iF_b - R_jF_b)^2} \right) \tag{3.2}$$

Once all the distances between true color and reference color have been calculated, we select the reference color R_j with the minimum euclidean distance, and this process gives origin to a new dataset that is purely built in reference colors, or in other words, it is a compressed lossy version of the original. Dataset reduction is possible due to the fact that the reference color is represented by a single value (e.g.: red, which is composed by RGB values of 255,0,0) instead of three different features, this compression makes possible to reduce the original dimension of any dataset to a third with very minimal loss of information, however although there is indeed a loss, this can be negligible as the data transformation is only used for the feature selection process, and the original information suffers no changes.

[1] https://www.w3.org/TR/html4/types.html#h-6.5

3.0.2 Stage 2 - Feature Selection

After compression is completed (Stage 1), selecting relevant features is based on the rationale that different classes should look different, hence their associated quantized colors should look different as well. Since FS occurs in the new compressed dataset, the process see regular features F_i although we know they represent a group in the original space. The *mode* is calculated for each feature-class (FC) relation i.e., a *mode* for $feature_1$ and $class_1$ (F_1C_1) and another *mode* for F_1C_2, and so on.

The conditional probability that a given value for *mode* belongs to a class or another is compared and if such probability exceeds other classes then we define the feature as useful. A threshold Th can be applied to produce more strict comparisons. In order to formalize the criteria, let $Pr(C_j \mid Mo(F_i))$ be the probability that a given mode in F_i be associated to a class C_j, then the feature F_i usefulness is defined as follows:

$$\begin{gathered}\text{Useful } F_i \text{ iff:} \\ \exists\{C_j, C_k\} \subset C \mid [Pr(C_j \mid Mo(C_j \mid F_i))] > [Pr(C_k \mid Mo(C_j \mid F_i)) * T_h]\end{gathered} \tag{3.3}$$

Hence, HmbFS defines that a feature is useful if it can find a pair of classes C_j and C_k where the probability that the *mode* (Mo) of the feature for C_j be greater than for C_k, or viewed from the heatmap point of view, if the most prominent (mode) color associated to a feature-class pair is significant larger (this significance can be controlled by T_h parameter) than the same color but for a different class.

After all useful features have been identified, we need to restore the original data, however such process is very efficient as each group is mapped to exactly 3 features, e.g., if the FS process selected (compressed groups) $F_i = \{2\}, \{5\}, \{7\}$ and $\{10\}$, we know they are mapped to the original space to $F_i = \{4,5,6\}, \{13,14,15\}, \{19,20,21\}$ and $\{28,29,30\}$. Below we present the algorithm pseudo-code:

Algorithm 1: HmbFS, Heat map based Feature Selection

Data: Dataset with full number of features
Result: Subset of best predictive features

Normalize Data 0-255;
while *there are more features* **do**
    \\Stage 1
    for F_i, F_{i+1}, *and* F_{i+2} **do**
        Build group G_i with features as 16-Bit RGB values;
        Save quantized 4-bit version of $G_i \rightarrow QG_i$;
    end
    Increase i by 3 ;
end
for *each group* QG_i **do**
    \\Stage 2
    for *each class* C_j **do**
        for *each class* C_k *!=* C_j **do**
            if $Pr(C_j \mid Color_{Mode(Cj|QGi)}) > [Pr(C_k \mid Color_{Mode(Cj|QGi)}) \times Th]$ **then**
                Mark QG_i as useful;
            end
        end
    end
end

As it can be seen from pseudo-code, our proposal is very programming friendly. The compression stage allows to reduce problem complexity and evaluate features in a smaller space without losing the original data because, at the end of the process we map our results to it as previously whoen in 3.1.

In order to evaluate the usefulness of HmbFS we have prepared a series of tests through different mass spectrometry datasets and compare individual as well as ensemble techniques for feature selection. In the next section we discuss more in depth the results.

3.0.3 HmbFS Experiments

As correctly pointed out by [60], reproduce experiments in works such as this one is sometimes hard to achieve, in this work we want to fill that gap by allowing scientists to easily replicate our experiments and compare with us, hence we have decided to employ a fully integrated framework, from reading

data up to cross-validation, the scikit-learn framework [61] provides all the feature selection algorithms we compared, as well as all the classifiers, so the validation can be as easy as to run a script with the parameters we show or execute the code we provide in the appendix.

In this work we performed our experiments with 7 MS datasets that deals with different pathologies. In Table 1, we summarize the main information for these datasets.

Author	Category	Classes	Samples	Features
[62]	Colon C.	2	62	2,000
[63]	Breast C.	2	104	22,283
[64]	Leukemia	2	72	7,129
[65]	Lymphoma	2	77	7,129
[66]	Prostate C.	2	102	12,600
[67]	Other	4	102	5,565
[68]	Myeloma	2	173	12,625

Table 3.1: Datasets main attributes

In order to compare the usefulness of HmbFS we selected seven currently used algorithms, including a non feature selection algorithm such as PCA [10] in order to have a reference point with other approaches of dimensionality reduction.

In Table 3.0.3 we provide an overview of the competing algorithms, some of their characteristics and implementation parameters. All selected algorithms can be found on the scikit-learn framework, in this work we included the Chi square (Chi2), False discovery rate (Fdr), False positive rate (Fpr), Anova F score (Fs), Family-wise error rate (Fwe), Principal component analysis (PCA) and Extra-Trees feature selection (Et).

Algorithm	Type	Parameters	Reference
Chi2	Ranking	Remove worst 25% feats	[69]
Fdr	Selection	F-value with alpha 0.05	[70]
Fpr	Selection	F-value with alpha 0.05	-
Fs	Ranking	Remove worst 25% feats	[71]
Fwe	Selection	F-value with alpha 0.05	-
HmbFS	Selection	Threshold = 1.5 and 2.0	This
PCA	Transform	No. Components = 90%	[72]
Et	Embedded	Number of trees = 10	[73]

Table 3.2: Dimensionality reduction algorithms overall information

We can notice that last two algorithms are not really in the same category as HmbFS, these being PCA which reduction of features can be substantial as it really does not select from the original features, but instead create new ones. And on the other hand, Et uses an embedded approach, training an

ensemble of trees and later selecting the most important features used in the splits, hence, requiring a classifier to work, however, including them can give a better overview of different techniques.

In order to test how effective the feature selection process was, we employed different algorithms to test the classification accuracy before and after feature selection. One key element in our tests is that our results are averaged across different classifiers, hence a feature selection algorithm which results are biased towards a particular classifier gets penalized, hence not producing a stable selection of features gets a much lower score due to final average of four classifiers. In Table 3.0.3 we show the main information of the classifiers used.

Algorithm	Reference
Gradient Boosting Machine (GBM)	[74]
K Nearest Neighbour (kNN)	[75]
Linear Support Vector Machine (LSVM)	[76]
Logistic Regression (LogReg)	[76]
Passive-Aggressive (PA)	[77]
Random Forest (RF)	[78]

Table 3.3: Classifiers used for testing

Our experiments can be categorized in three groups:

1) Learning with full features:

Although it has been claimed [79] that some classification algorithms such as decision trees are inherently built-in with feature selection, they can learn noise which will result in lower accuracy. In our experiments we have included two tree-based techniques, including boosting over decision trees, to demonstrate how even this algorithms can produce better results with the pre-filtering of features.

2) Learning with HmbFS reduced set:

We want to see how HmbFS compares with other dimensionality reduction techniques, as well as to evaluate if accuracy can be improved compared to the full set of features. Here is important to remark that for the nature of MS data, running HmbFS with a $T_h = 1.5$ is a very poor choice, however, for fair comparison with other techniques we chose the default value (1.5), as well as a more suited threshold (2.0) which produced a smaller set of features.

3) Ensemble of feature selection:

Two feature selection algorithms could perform better than one, that is what this experiment tries to investigate, since we are using HmbFS with a low threshold (not so aggressive reduction), it is possible to pipe its output so other feature selections algorithms can get better results than what they would get using the full number of features, and the same applies for other techniques, although algorithms with heavy reduction, such as tree-based ones, may limit the ability of the second stage process.

Summary of results

We have performed multiple experiments as a result of testing 7 datasets, each one tested under different setups of dimensionality reduction and classification algorithms combinations. Due to the inherent characteristics of MS datasets where the number of samples is very low, we used leave-one-out (LOO) as our cross validation scheme with accuracy as the main reported metric, however, precision and recall were analyzed as well and the provided code generates reports for every metric.

For every setup we provide the results for every setup of feature selection+classifiers. These results include the number of selected features, the accuracy for each of the four best classifiers in that dataset, as well as the final averaged value across all classifiers. Notice that the best individual result does not necessarily leads to the best selection algorithm as we want to evaluate the stability of the selected features across the classifiers, not just one result.

In Table 3.0.3, we present the results for the Alon dataset:

FS Method	No.Feats	KNN	LSVM	LogReg	PA	Average
Fs + Chi2	1125	85.48%	82.26%	87.1%	85.48%	85.08%
Fpr + Chi2	291	85.48%	85.48%	82.26%	85.48%	84.68%
Fpr + Fdr	386	85.48%	85.48%	82.26%	85.48%	84.68%
Fs + Fpr	389	85.48%	85.48%	82.26%	85.48%	84.68%
Fpr	389	85.48%	85.48%	82.26%	85.48%	84.68%
Chi2 + Fpr	389	85.48%	85.48%	82.26%	85.48%	84.68%
$HmbFS_{2.0}$	684	87.1%	80.65%	83.87%	85.48%	84.27%
Fs	1500	85.48%	80.65%	85.48%	85.48%	84.27%
Chi2	1500	85.48%	80.65%	83.87%	87.1%	84.27%
Fs + Et	117	80.65%	82.26%	85.48%	87.1%	83.87%
$HmbFS_{2.0}$ + Et	120	87.1%	79.03%	80.65%	88.71%	83.87%
Chi2 + Fs	1125	83.87%	80.65%	83.87%	87.1%	83.87%
Full	2000	83.87%	82.26%	83.87%	85.48%	83.87%
$HmbFS_{1.5}$	1074	85.48%	80.65%	82.26%	85.48%	83.47%
Fdr	65	83.87%	77.42%	79.03%	75.81%	79.03%
Pca	16	83.87%	69.35%	82.26%	74.19%	77.42%
Et	113	80.65%	74.19%	75.81%	77.42%	77.02%
Fwe	8	85.48%	64.52%	85.48%	64.52%	75.0%

Table 3.4: Alon dataset results

Although in the case of Alon dataset, the overall average accuracy did not improve notably, i.e., we can expect the improvement to be classifier dependent, important insights about the feature selection can be obtained, for instance, algorithms with unstable selection can lead to considerable drop in performance, and this affects algorithms with multivariable evaluation as well (e.g., HmbFS and Et). Combination of methods can lead to better performance than single algorithms, for instance, Et using the original 2000 features, selected 113 with a considerable loss in performance (-6.85%), however, the same algorithm (Et), when applied to the subset of features selected by HmbFS, using a similar number of features (120 vs 113) maintained the original performance of using the full set.

The best score using our proposed HmbFS algorithm (using Th=2.0) was in combination with KNN classifier achieved 87.1% accuracy performance, an improvement of 1.62% over the best score using the full dataset, an important remark is that improvement extends to precision and recall as well, precision was improved from 84.3% to 87.6% and recall from 83.6% to 83.8%, using only 684 features. The best individual result was only possible after applying HmbFS before using Et, achieving 88.71%

with PA algorithm, more than 3% than the same algorithm, but the full set of features.

Table 3.0.3 and 3.0.3, shows the results for Burczynski dataset.

FS Method	No.Feats	GBM	KNN	LSVM	LogReg	Average
Fpr + Fwe	1368	81.89%	84.25%	95.28%	95.28%	89.17%
Fwe + Chi2	874	81.89%	84.25%	95.28%	94.49%	88.98%
Fwe + Fs	874	82.68%	84.25%	94.49%	94.49%	88.98%
Fdr + Fwe	1442	81.1%	85.04%	95.28%	93.7%	88.78%
Chi2 + Fwe	1234	79.53%	85.04%	95.28%	94.49%	88.58%
Fs + Fwe	1234	79.53%	85.04%	95.28%	94.49%	88.58%
$HmbFS_{1.5}$ + Fwe	1062	81.1%	83.46%	94.49%	95.28%	88.58%
Fwe	1166	80.31%	84.25%	95.28%	94.49%	88.58%
Fwe + Fdr	1166	80.31%	84.25%	95.28%	94.49%	88.58%
Fwe + Fpr	1166	80.31%	84.25%	95.28%	94.49%	88.58%
$HmbFS_{1.5}$ + Et	255	83.46%	81.89%	92.91%	93.7%	87.99%
$HmbFS_{2.0}$ + Fwe	902	82.68%	83.46%	92.91%	92.13%	87.8%
$HmbFS_{2.0}$ + Fs	7983	83.46%	76.38%	95.28%	94.49%	87.4%
$HmbFS_{1.5}$ + Fpr	6844	81.89%	76.38%	96.06%	94.49%	87.2%
Fdr + Et	242	81.1%	78.74%	92.91%	95.28%	87.01%
$HmbFS_{2.0}$	10644	82.68%	74.02%	96.06%	94.49%	86.81%
$HmbFS_{1.5}$	16218	81.89%	72.44%	96.85%	96.06%	86.81%
$HmbFS_{2.0}$ + Fpr	4844	81.89%	76.38%	94.49%	93.7%	86.61%

Table 3.5: Burczynski dataset results 1/2

FS Method	No.Feats	GBM	KNN	LSVM	LogReg	Average
Fwe + Et	226	81.1%	79.53%	93.7%	92.13%	86.61%
$HmbFS_{2.0}$ + Chi2	7983	83.46%	74.02%	95.28%	93.7%	86.61%
$HmbFS_{2.0}$ + Fdr	3924	85.04%	75.59%	93.7%	91.34%	86.42%
$HmbFS_{1.5}$ + Fdr	5206	82.68%	75.59%	93.7%	93.7%	86.42%
$HmbFS_{1.5}$ + Chi2	12163	80.31%	72.44%	96.85%	95.28%	86.22%
$HmbFS_{1.5}$ + Fs	12163	78.74%	71.65%	97.64%	96.06%	86.02%
Fdr + Fs	4744	74.02%	79.53%	94.49%	94.49%	85.63%
Fdr + Chi2	4744	78.74%	76.38%	92.91%	92.91%	85.24%
Chi2 + Fs	12534	76.38%	73.23%	96.06%	95.28%	85.24%
Full	22283	75.59%	71.65%	96.06%	96.06%	84.84%
Fpr + Chi2	6610	76.38%	75.59%	93.7%	93.7%	84.84%
Chi2 + Fpr	8617	77.17%	75.59%	92.91%	93.7%	84.84%
Fdr	6326	76.38%	76.38%	92.91%	92.91%	84.65%
Chi2	16712	72.44%	72.44%	96.85%	96.06%	84.45%
Fpr	8814	74.02%	75.59%	92.91%	93.7%	84.06%
Fs	16712	73.23%	69.29%	96.85%	96.06%	83.86%
Et	253	77.17%	74.02%	91.34%	88.98%	82.87%
Pca	28	71.65%	74.02%	77.17%	81.1%	75.98%

Table 3.6: Burczynski dataset results 2/2

With the Burczynski dataset, only two algorithms managed to improve over the full dataset (Fwe and HmbFS), all others lost predictive performance specially PCA with a loss of almost 9%.

As the number of features increase, the usage of different threshold values for HmbFS can produce more radical results, in this case, 16,218 features vs 10,644, however, both produce a similar performance, as the features are inclusive (all 10,644 are included in the 16,218 subset). The Fwe algorithm performed better than our proposal with very few features, only 1,116, however HmbFS seemed to be useful in conjunction with Fwe as well, producing same overall performance with less features.

Best averaged performance was achieved with Fpr + Fwe, with a 4.33% improvement in overall accuracy vs the full dataset, with a small lose in precision of 0.9% and a lose of 1.78% for recall. Using HmbFS precision and recall dropped as well but by minimal margin of 0.3% and 0.5% respectively. The best individual score was achieved by $HmbFS_{Th=1.5}$ when used in conjunction with LSVM (96.85%), although other techniques such as Chi2 and Fs managed it as well, their selection was not stable enough to achieve a better averaged performance than the full set.

Table 3.0.3 and 3.0.3, shows our findings for Chin dataset tests

FS Method	No.Feats	KNN	LSVM	LogReg	RF	Average
Et + Fs	132	89.83%	88.98%	88.98%	90.68%	89.62%
Et + Chi2	132	88.98%	88.14%	89.83%	88.98%	88.98%
Et	176	89.83%	88.14%	90.68%	86.44%	88.77%
$HmbFS_{1.5}$ + Et	186	88.14%	88.14%	90.68%	86.44%	88.35%
Et + Fpr	84	89.83%	87.29%	88.14%	88.14%	88.35%
Et + Fdr	73	91.53%	88.98%	88.14%	83.9%	88.14%
Chi2 + Fwe	1023	87.29%	86.44%	88.98%	88.98%	87.92%
Fs + Fwe	1023	87.29%	86.44%	88.98%	88.98%	87.92%
Fdr + Et	169	89.83%	87.29%	88.14%	86.44%	87.92%
$HmbFS_{1.5}$ + Fwe	694	88.98%	85.59%	88.98%	86.44%	87.5%
$HmbFS_{2.0}$ + Fs	3001	87.29%	87.29%	86.44%	88.98%	87.5%
Fwe + Et	150	86.44%	85.59%	88.14%	88.98%	87.29%
$HmbFS_{2.0}$	4002	87.29%	86.44%	86.44%	88.98%	87.29%
Fwe + Fs	714	87.29%	84.75%	88.98%	88.14%	87.29%
Fwe + Chi2	714	87.29%	85.59%	88.98%	86.44%	87.08%
$HmbFS_{2.0}$ + Chi2	3001	87.29%	86.44%	86.44%	88.14%	87.08%
Chi2 + Fs	12495	88.14%	88.98%	86.44%	84.75%	87.08%
Fdr + Chi2	4071	87.29%	85.59%	87.29%	88.14%	87.08%
Fpr + Chi2	5684	87.29%	87.29%	87.29%	86.44%	87.08%
Fdr + Fwe	1278	87.29%	86.44%	87.29%	86.44%	86.86%

Table 3.7: Chin dataset results 1/2

FS Method	No.Feats	KNN	LSVM	LogReg	RF	Average
Fs + Chi2	12495	88.14%	87.29%	86.44%	85.59%	86.86%
Fs	16661	88.14%	86.44%	86.44%	85.59%	86.65%
$HmbFS_{2.0}$ + Et	163	89.83%	84.75%	85.59%	86.44%	86.65%
Fwe	953	87.29%	85.59%	88.98%	84.75%	86.65%
Fwe + Fdr	953	87.29%	85.59%	88.98%	84.75%	86.65%
Fwe + Fpr	953	87.29%	85.59%	88.98%	84.75%	86.65%
$HmbFS_{1.5}$ + Fs	5564	87.29%	87.29%	86.44%	85.59%	86.65%
Fpr + Fwe	1192	87.29%	86.44%	88.14%	83.9%	86.44%
Chi2 + Fpr	7579	87.29%	88.14%	88.14%	82.2%	86.44%
Fpr	7579	87.29%	88.14%	88.14%	82.2%	86.44%
Fpr + Fdr	7579	87.29%	88.14%	88.14%	82.2%	86.44%
Fs + Fpr	7579	87.29%	88.14%	88.14%	82.2%	86.44%
Chi2	16661	88.14%	88.14%	88.14%	81.36%	86.44%
Full	22215	88.14%	87.29%	86.44%	83.9%	86.44%
Fdr	5429	87.29%	87.29%	87.29%	83.05%	86.23%
$HmbFS_{1.5}$	7419	87.29%	86.44%	86.44%	83.9%	86.02%
Pca	75	86.44%	77.12%	78.81%	76.27%	79.66%

Table 3.8: Chin dataset results 2/2

For the Chin dataset, our approach with Th=1.5 performed slightly under the full dataset benchmark, losing 0.22% in average accuracy in exchange for a +66% reduction in features, however increasing the threshold to Th=2.0 corrected the issue with less features and better performance.

Worth noticing is that performing under the full benchmark can still produce good results when combined, HmbFS with a Th=1.5 managed to improve the Fwe algorithm, reducing its features from 963 to 694, and improving the performance by 0.85%.

Using Et resulted in very good results, with an average accuracy of 88.7%, which was further improved by a second stage selection of Fs, the result is only 132 features that performed 3.18% better in average accuracy, with improvements in precision (0.2%) and recall (1.6%).

Table 3.0.3 and 3.0.3, shows Chowdary results.

FS Method	No.Feats	KNN	LSVM	LogReg	RF	Average
$HmbFS_{1.5}$ + Fwe	49	96.15%	99.04%	98.08%	96.15%	97.36%
$HmbFS_{1.5}$ + Fdr	169	96.15%	98.08%	96.15%	98.08%	97.12%
$HmbFS_{1.5}$ + Chi2	177	97.12%	96.15%	97.12%	97.12%	96.88%
$HmbFS_{1.5}$ + Fs	177	96.15%	98.08%	96.15%	97.12%	96.88%
$HmbFS_{2.0}$ + Fs	54	94.23%	97.12%	98.08%	97.12%	96.63%
$HmbFS_{2.0}$ + Fwe	21	95.19%	96.15%	98.08%	97.12%	96.63%
Et + Fpr	79	98.08%	96.15%	95.19%	97.12%	96.63%
Fdr + Et	116	95.19%	97.12%	98.08%	96.15%	96.63%
$HmbFS_{1.5}$ + Fpr	180	96.15%	98.08%	96.15%	96.15%	96.63%
$HmbFS_{2.0}$ + Chi2	54	94.23%	97.12%	97.12%	97.12%	96.39%
$HmbFS_{2.0}$ + Fdr	52	94.23%	96.15%	97.12%	98.08%	96.39%
$HmbFS_{1.5}$ + Et	66	95.19%	97.12%	97.12%	96.15%	96.39%
Fs + Et	104	93.27%	98.08%	98.08%	96.15%	96.39%
$HmbFS_{1.5}$	237	97.12%	96.15%	97.12%	95.19%	96.39%
Et + Fdr	75	98.08%	96.15%	96.15%	95.19%	96.39%
Et + Fs	82	98.08%	96.15%	95.19%	96.15%	96.39%
Fpr + Et	108	96.15%	97.12%	96.15%	96.15%	96.39%
$HmbFS_{2.0}$	72	94.23%	98.08%	97.12%	95.19%	96.15%
Fdr + Fwe	174	96.15%	96.15%	96.15%	96.15%	96.15%
$HmbFS_{2.0}$ + Fpr	56	94.23%	97.12%	98.08%	94.23%	95.91%
Fwe + Chi2	102	96.15%	95.19%	95.19%	97.12%	95.91%
Fwe + Fs	102	96.15%	94.23%	96.15%	97.12%	95.91%
Et + Chi2	82	96.15%	95.19%	96.15%	96.15%	95.91%
Et	110	95.19%	96.15%	96.15%	96.15%	95.91%
Chi2 + Fwe	156	96.15%	95.19%	96.15%	96.15%	95.91%

Table 3.9: Chowdary dataset results 1/2

FS Method	No.Feats	KNN	LSVM	LogReg	RF	Average
Fs + Fwe	156	96.15%	95.19%	96.15%	96.15%	95.91%
Fpr + Fwe	161	96.15%	95.19%	96.15%	96.15%	95.91%
Fwe	137	96.15%	95.19%	96.15%	95.19%	95.67%
Fwe + Fdr	137	96.15%	95.19%	96.15%	95.19%	95.67%
Fwe + Fpr	137	96.15%	95.19%	96.15%	95.19%	95.67%
Et + Fwe	43	97.12%	95.19%	95.19%	94.23%	95.43%
$HmbFS_{2.0}$ + Et	20	93.27%	94.23%	97.12%	96.15%	95.19%
Chi2	16712	92.31%	97.12%	96.15%	94.23%	94.95%
Fwe + Et	34	96.15%	93.27%	94.23%	96.15%	94.95%
Chi2 + Fs	12534	88.46%	97.12%	96.15%	97.12%	94.71%
Fs + Chi2	12534	90.38%	97.12%	97.12%	94.23%	94.71%
Chi2 + Fpr	13576	89.42%	98.08%	96.15%	95.19%	94.71%
Fpr + Fdr	15095	89.42%	98.08%	96.15%	95.19%	94.71%
Fdr + Fs	9643	88.46%	97.12%	97.12%	95.19%	94.47%
Fpr	15096	89.42%	98.08%	96.15%	94.23%	94.47%
Fs + Fpr	15096	89.42%	98.08%	96.15%	94.23%	94.47%
Fpr + Fs	11322	88.46%	97.12%	97.12%	94.23%	94.23%
Fs	16712	90.38%	97.12%	97.12%	92.31%	94.23%
Fdr	12858	88.46%	97.12%	96.15%	95.19%	94.23%
Full	22283	91.35%	97.12%	96.15%	92.31%	94.23%
Fdr + Fpr	12858	88.46%	97.12%	96.15%	95.19%	94.23%
Pca	3	95.19%	84.62%	94.23%	94.23%	92.07%

Table 3.10: Chowdary dataset results 2/2

For the Chowdary dataset we got very good results with our approach, it seems that the data really benefit for multivariate analysis, therefore Et and HmbFS produced the biggest improvements, Et for instance achieved an overall improvement of 1.68%, although such improvement comes only from 2 classifiers (KNN & RF), on the other hand HmbFS with Th=2.0 achieved 1.92% improvement that comes from all classifiers, using a Th=1.5 improves a bit more to an overall accuracy of 96.39% just as high as the combination of two methods Fs + Et but at the expense of more features (237 vs 104).

Another important factor to consider is how a low performance algorithm can produce great results once assembled with other methods, as is the case of Fdr which produced no improvement over the base full data, however, in combination with $HmbFS_{Th=1.5}$ produced the second best averaged score at

97.12% which represents a 2.89% improvement in overall accuracy, as well as 1.2% in precision and 0.9% in recall.

For this dataset, improvement was very hard because the full data benchmark achieved good results, including a 97.12% with LSVM, and, although most algorithms were able to produce a better averaged score, very few managed to improve that individual score, Fwe for instance, got 95.15% with LSVM, a loss of 1.97%, however, if we first use HmbFS to pre-filter the features for Fwe, it is possible to get the best individual score of 99.04%, an improvement of 3.89% if using Fwe alone.

Table 3.0.3, shows the results for the Singh dataset.

FS Method	No.Feats	GBM	LSVM	LogReg	PA	Average
Fpr + Et	158	91.18%	94.12%	95.1%	94.12%	93.63%
Fdr + Et	178	90.2%	96.08%	94.12%	94.12%	93.63%
$HmbFS_{1.5}$ + Fwe	161	88.24%	94.12%	96.08%	92.16%	92.65%
Fwe + Et	47	85.29%	97.06%	95.1%	91.18%	92.16%
$HmbFS_{1.5}$ + Et	177	88.24%	94.12%	94.12%	92.16%	92.16%
$HmbFS_{2.0}$ + Fwe	133	92.16%	93.14%	93.14%	88.24%	91.67%
Fdr + Fwe	253	87.25%	95.1%	95.1%	89.22%	91.67%
$HmbFS_{2.0}$ + Et	151	87.25%	93.14%	93.14%	93.14%	91.67%
Fs + Et	191	86.27%	93.14%	94.12%	93.14%	91.67%
$HmbFS_{2.0}$ + Fpr	589	89.22%	94.12%	94.12%	89.22%	91.67%
$HmbFS_{2.0}$ + Fdr	494	89.22%	92.16%	94.12%	90.2%	91.42%
Chi2 + Fwe	185	85.29%	94.12%	94.12%	91.18%	91.18%
Fs + Fwe	185	85.29%	94.12%	94.12%	91.18%	91.18%
Fpr + Fwe	218	84.31%	94.12%	94.12%	91.18%	90.93%
Fwe	168	84.31%	93.14%	94.12%	91.18%	90.69%
Fwe + Fdr	168	84.31%	93.14%	94.12%	91.18%	90.69%
Fwe + Fpr	168	84.31%	93.14%	94.12%	91.18%	90.69%
$HmbFS_{2.0}$	1050	89.22%	92.16%	93.14%	88.24%	90.69%
Fpr + Pca	25	82.35%	94.12%	94.12%	91.18%	90.44%
$HmbFS_{2.0}$ + Fs	787	89.22%	93.14%	92.16%	87.25%	90.44%
$HmbFS_{1.5}$ + Fdr	994	87.25%	95.1%	93.14%	86.27%	90.44%

Table 3.11: Singh dataset results 1/2

FS Method	No.Feats	GBM	LSVM	LogReg	PA	Average
$HmbFS_{1.5}$ + Fpr	1290	86.27%	94.12%	94.12%	87.25%	90.44%
Fs	9450	87.25%	93.14%	93.14%	88.24%	90.44%
Fdr + Pca	25	82.35%	91.18%	96.08%	91.18%	90.2%
$HmbFS_{2.0}$ + Chi2	787	88.24%	91.18%	93.14%	88.24%	90.2%
$HmbFS_{1.5}$	2496	87.25%	94.12%	92.16%	87.25%	90.2%
Chi2	9450	85.29%	93.14%	92.16%	90.2%	90.2%
Et + Fs	131	86.27%	93.14%	93.14%	88.24%	90.2%
$HmbFS_{1.5}$ + Fs	1872	89.22%	93.14%	92.16%	86.27%	90.2%
Et	175	85.29%	94.12%	92.16%	88.24%	89.95%
Et + Fpr	99	86.27%	95.1%	93.14%	85.29%	89.95%
Fwe + Pca	27	83.33%	91.18%	93.14%	91.18%	89.71%
Fpr + Fdr	5537	83.33%	95.1%	93.14%	87.25%	89.71%
Fwe + Chi2	126	82.35%	92.16%	94.12%	89.22%	89.46%
Et + Chi2	131	86.27%	92.16%	93.14%	85.29%	89.22%
Et + Pca	28	80.39%	93.14%	93.14%	90.2%	89.22%
Full	12600	88.24%	91.18%	92.16%	84.31%	88.97%
Fwe + Fs	126	81.37%	93.14%	93.14%	88.24%	88.97%
Chi2 + Fpr	5538	80.39%	95.1%	93.14%	87.25%	88.97%
Fpr	5538	80.39%	95.1%	93.14%	87.25%	88.97%
Fs + Fpr	5538	80.39%	95.1%	93.14%	87.25%	88.97%
Chi2 + Fs	7087	77.45%	95.1%	93.14%	90.2%	88.97%
Fdr	3133	79.41%	95.1%	93.14%	82.35%	87.5%
Pca	32	74.51%	87.25%	87.25%	85.29%	83.58%

Table 3.12: Singh dataset results 2/2

The Singh dataset responded very well to ensemble of methods, especially when combined with Fwe and HmbFS, both algorithms managed the biggest improvement at 1.72%, however, HmbFS managed to improve across all four classifiers, while Fwe was a bit less stable, losing 3.93% when trained with GBM, this could be caused by the much less selected features, 168 vs 1050 from HmbFS.

The Et algorithm managed to reduce the dataset by a huge ratio (over 98% reduction), although with limited performance improvement (0.98%), however, once assembled with other techniques, such as Fpr or Fdr it was possible to have a considerable improvement, 4.66% overall increase in accuracy, however depending the algorithm, the difference could go up to 9.81% as was the case with PA.

With regard to precision and recall, the improvement correlates with accuracy, for instance, using $HmbFS_{Th=2.0}$, precision and recall both got improvement of 0.9% which although not a big difference, it confirms that the increased accuracy is not due increase of false positives or other cases of data overfitting.

In Table 9, we show our findings for the Su dataset

FS Method	No.Feats	KNN	LSVM	LogReg	PA	Average
Fdr + Et	156	99.02%	100%	100%	100%	99.75%
Chi2 + Et	163	99.02%	100%	100%	100%	99.75%
Fwe + Et	139	99.02%	99.02%	100%	100%	99.51%
Fwe + Pca	59	98.04%	100%	100%	99.02%	99.26%
Fs + Pca	65	99.02%	100%	99.02%	99.02%	99.26%
Chi2 + Pca	67	98.04%	100%	99.02%	99.02%	99.02%
Fs + Chi2	3129	99.02%	100%	98.04%	99.02%	99.02%
$HmbFS_{2.0}$ + Chi2	3426	99.02%	100%	98.04%	99.02%	99.02%
$HmbFS_{2.0}$ + Fs	3426	99.02%	100%	98.04%	99.02%	99.02%
Fdr + Chi2	3459	99.02%	100%	98.04%	99.02%	99.02%
Fpr + Chi2	3494	99.02%	100%	98.04%	99.02%	99.02%
$HmbFS_{1.5}$ + Chi2	3854	99.02%	100%	98.04%	99.02%	99.02%
$HmbFS_{1.5}$ + Fs	3854	99.02%	100%	98.04%	99.02%	99.02%
$HmbFS_{2.0}$ + Fdr	3856	99.02%	100%	98.04%	99.02%	99.02%
$HmbFS_{2.0}$ + Fpr	3885	99.02%	100%	98.04%	99.02%	99.02%
Chi2 + Fdr	3927	99.02%	100%	98.04%	99.02%	99.02%
Chi2 + Fpr	3934	99.02%	100%	98.04%	99.02%	99.02%
Chi2	4173	99.02%	100%	98.04%	99.02%	99.02%
$HmbFS_{2.0}$	4569	99.02%	100%	98.04%	99.02%	99.02%
Et + Pca	39	99.02%	99.02%	99.02%	99.02%	99.02%
Et + Chi2	112	99.02%	99.02%	99.02%	99.02%	99.02%
Et + Fs	112	99.02%	99.02%	99.02%	99.02%	99.02%
Et + Fwe	123	99.02%	99.02%	99.02%	99.02%	99.02%
Et + Fdr	141	99.02%	99.02%	99.02%	99.02%	99.02%
Et + Fpr	141	99.02%	99.02%	99.02%	99.02%	99.02%
Et	150	99.02%	99.02%	99.02%	99.02%	99.02%
Fs + Et	157	99.02%	99.02%	99.02%	99.02%	99.02%
$HmbFS_{1.5}$ + Et	159	99.02%	99.02%	99.02%	99.02%	99.02%
Fpr + Et	177	99.02%	99.02%	99.02%	99.02%	99.02%

Table 3.13: Su dataset results 1/2

FS Method	No.Feats	KNN	LSVM	LogReg	PA	Average
$HmbFS_{1.5}$ + Fwe	2274	98.04%	100%	99.02%	98.04%	98.77%
Fwe	2401	98.04%	100%	99.02%	98.04%	98.77%
Fwe + Fdr	2401	98.04%	100%	99.02%	98.04%	98.77%
Fwe + Fpr	2401	98.04%	100%	99.02%	98.04%	98.77%
Fpr + Fwe	2428	98.04%	100%	99.02%	98.04%	98.77%
Fdr + Fwe	2429	98.04%	100%	99.02%	98.04%	98.77%
Fs + Fwe	2449	98.04%	100%	99.02%	98.04%	98.77%
Chi2 + Fs	3129	99.02%	100%	98.04%	98.04%	98.77%
Fdr + Fs	3459	99.02%	100%	98.04%	98.04%	98.77%
Fpr + Fs	3494	99.02%	100%	98.04%	98.04%	98.77%
Fs	4173	99.02%	100%	98.04%	98.04%	98.77%
Fs + Fdr	4173	99.02%	100%	98.04%	98.04%	98.77%
Fs + Fpr	4173	99.02%	100%	98.04%	98.04%	98.77%
$HmbFS_{1.5}$ + Fdr	4288	99.02%	100%	98.04%	98.04%	98.77%
$HmbFS_{1.5}$ + Fpr	4327	99.02%	100%	98.04%	98.04%	98.77%
Fdr	4613	99.02%	100%	98.04%	98.04%	98.77%
Fdr + Fpr	4613	99.02%	100%	98.04%	98.04%	98.77%
Fpr	4659	99.02%	100%	98.04%	98.04%	98.77%
Fpr + Fdr	4659	99.02%	100%	98.04%	98.04%	98.77%
Fpr + Pca	66	97.06%	100%	99.02%	99.02%	98.77%
$HmbFS_{2.0}$ + Pca	68	98.04%	99.02%	99.02%	99.02%	98.77%
Fwe + Fs	1800	99.02%	99.02%	99.02%	98.04%	98.77%
Chi2 + Fwe	2425	98.04%	100%	98.04%	98.04%	98.53%
$HmbFS_{2.0}$ + Et	155	99.02%	98.04%	98.04%	99.02%	98.53%
Fwe + Chi2	1800	98.04%	99.02%	99.02%	98.04%	98.53%
$HmbFS_{2.0}$ + Fwe	2105	98.04%	99.02%	99.02%	98.04%	98.53%
$HmbFS_{1.5}$	5139	99.02%	99.02%	98.04%	98.04%	98.53%
Full	5565	99.02%	99.02%	98.04%	98.04%	98.53%
Pca	68	96.08%	99.02%	99.02%	99.02%	98.28%

Table 3.14: Su dataset results 2/2

The Su dataset represents a hard challenge for feature selection for two main reasons, first, low dimension of features makes the reduction more sensitive to changes, and second, a very high default

performance, over 98% using the full of set of features suggests that there is very little room for improvement. As was expected, HmbFS had issues to remove features, using a Th=1.5 the reduction was merely 426 features, however performance remained identical and using a higher threshold at 2.0 resulted in 996 features and an increase in overall accuracy, included a 100% score using Linear SVM.

Worth noting is the fact that most algorithms did very few reduction in the number of features, (PCA with 68 features, is not really a reduction per se, as the original features are mixed among those 68), the only exception was Et algorithm which managed a huge reduction, with only 150 features it was possible to improve the overall accuracy, although it was unable to capture the 100% accuracy with LSVM, on the other hand Chi2 managed the best individual performance (tied with HmbFS) with the fewer features.

There was a key breaking point at 99.02% overall accuracy, where many techniques did not managed to improve, however, although no improvement was possible, a reduction of features that still kept such performance was found at 3129 features (1044 less than Chi2). However, Fdr and Chi2 in combination with Et, managed an overall 99.75% accuracy, including three perfect scores for LSVM, LogReg and PA.

In Table 10, we show the results for the Tian dataset.

FS Method	No.Feats	KNN	LSVM	LogReg	RF	Average
Et + Fdr	33	79.77%	79.19%	82.66%	83.82%	81.36%
$HmbFS_{1.5}$ + Fpr	696	78.03%	83.82%	81.5%	81.5%	81.21%
Et + Fpr	74	82.66%	80.35%	79.19%	82.08%	81.07%
Et + Fwe	10	80.35%	76.88%	83.24%	83.82%	81.07%
$HmbFS_{1.5}$ + Fdr	118	80.92%	78.61%	79.19%	82.66%	80.35%
$HmbFS_{1.5}$ + Fs	2715	80.92%	80.35%	82.08%	78.03%	80.35%
Fdr	69	82.08%	76.88%	78.61%	83.24%	80.2%
Fdr + Fpr	69	82.08%	76.88%	78.61%	83.24%	80.2%
Fdr + Fwe	69	82.08%	76.88%	78.61%	83.24%	80.2%
Fdr + Et	33	80.35%	80.92%	77.46%	81.5%	80.06%
$HmbFS_{2.0}$ + Fs	1010	78.03%	81.5%	81.5%	79.19%	80.06%
$HmbFS_{2.0}$ + Et	259	79.77%	79.19%	79.19%	80.92%	79.77%
Fpr + Fs	1245	79.19%	78.03%	77.46%	84.39%	79.77%
$HmbFS_{2.0}$	1347	79.77%	78.03%	78.61%	82.08%	79.62%
$HmbFS_{2.0}$ + Chi2	1010	78.03%	79.19%	79.19%	80.92%	79.34%
Fpr + Et	258	82.66%	75.14%	78.61%	79.77%	79.05%
Fdr + Fs	51	81.5%	75.72%	75.14%	83.82%	79.05%
$HmbFS_{2.0}$ + Fdr	99	79.77%	76.88%	79.19%	79.19%	78.76%
$HmbFS_{1.5}$ + Chi2	2715	78.61%	78.03%	78.61%	79.77%	78.76%
Et	262	78.61%	77.46%	78.03%	80.35%	78.61%
Fpr + Fwe	22	81.5%	71.1%	76.88%	84.39%	78.47%
Fs + Et	264	80.35%	75.72%	76.3%	80.92%	78.32%
Chi2 + Fpr	1660	78.03%	75.72%	76.88%	82.08%	78.18%
Fpr	1660	78.03%	75.72%	76.88%	82.08%	78.18%
Fpr + Fdr	1660	78.03%	75.72%	76.88%	82.08%	78.18%
Fs + Fpr	1660	78.03%	75.72%	76.88%	82.08%	78.18%

Table 3.15: Tian dataset results 1/2

FS Method	No.Feats	KNN	LSVM	LogReg	RF	Average
Fdr + Chi2	51	82.08%	73.41%	74.57%	82.08%	78.03%
Chi2 + Et	255	80.35%	76.3%	75.72%	79.77%	78.03%
Chi2 + Fdr	110	79.77%	74.57%	73.99%	82.66%	77.75%
Fs + Fdr	110	79.77%	74.57%	73.99%	82.66%	77.75%
Fwe + Chi2	4	78.61%	73.99%	82.08%	76.3%	77.75%
Et + Fs	196	77.46%	78.61%	75.14%	79.77%	77.75%
$HmbFS_{1.5}$	3621	77.46%	78.03%	76.88%	78.61%	77.75%
$HmbFS_{2.0}$ + Fpr	329	78.61%	73.99%	75.72%	81.5%	77.46%
Fpr + Chi2	1245	78.03%	75.72%	76.88%	78.61%	77.31%
Fwe	6	76.88%	68.21%	83.24%	80.35%	77.17%
Fwe + Fdr	6	76.88%	68.21%	83.24%	80.35%	77.17%
Fwe + Fpr	6	76.88%	68.21%	83.24%	80.35%	77.17%
Fdr + Pca	12	79.77%	65.32%	83.24%	80.35%	77.17%
$HmbFS_{1.5}$ + Fwe	12	77.46%	70.52%	80.92%	79.77%	77.17%
$HmbFS_{1.5}$ + Et	279	79.77%	75.14%	75.14%	78.61%	77.17%
$HmbFS_{2.0}$ + Fwe	11	78.03%	67.63%	81.5%	80.35%	76.88%
Fs + Chi2	7101	79.77%	73.99%	74.57%	78.61%	76.73%
Fwe + Fs	4	78.61%	69.94%	82.66%	75.72%	76.73%
Chi2 + Fwe	10	75.72%	67.63%	79.77%	82.08%	76.3%
Fs + Fwe	10	75.72%	67.63%	79.77%	82.08%	76.3%
Et + Chi2	196	78.61%	72.25%	73.41%	80.92%	76.3%
Fs	9468	79.77%	73.99%	73.99%	76.88%	76.16%
Chi2 + Fs	7101	79.19%	73.41%	72.83%	77.46%	75.72%
Chi2	9468	80.35%	69.94%	72.83%	79.19%	75.58%
Fwe + Et	3	78.03%	65.9%	80.92%	75.72%	75.14%
Fwe + Pca	3	78.03%	65.9%	80.92%	75.72%	75.14%
Full	12625	80.35%	71.1%	71.68%	76.3%	74.86%
Pca	73	80.35%	47.4%	53.18%	76.3%	64.31%

Table 3.16: Tian dataset results 2/2

The Tian dataset presents many interesting insights, starting with Fwe algorithm that selected only 6 features and still managed to improve overall accuracy by 2.31%, however the poor stability (improvement only occurs with 2 algorithms) suggests the possibility of overfitting issues. other

techniques such as Fdr carried improvement regardless of the selected classifier and HmbFS was consistent with improvement in 3 of 4 classifiers regardless of the selected threshold.

The best two algorithms were Fdr and HmbFS, both managed to improve over the base full dataset by 5.34% and 4.76% respectively, however the difference in features was considerable better in favor of Fdr, although such heavy reduction in features limits the ability to ensemble the algorithm, still the top solutions were again carried out by HmbFS and Fdr in combination with other techniques, this is again in favor of Fdr with a minimal margin of 0.15% in accuracy performance.

When analyzing other metrics such as precision and recall, important issues appear, as the full set of features can achieve an accuracy as high as 80.35% but a recall as low as 55.84%, for instance, HmbFS managed to improve it by 9.76% when used alone with Th=1.5 and set the recall as high as 74.4% when used in conjunction with Fpr.

In Table 3.0.3, we present the overall overview of results, a direct comparison of HmbFS with different thresholds with other techniques, for each dataset we count which algorithm performed best.

Comparison	HmbFS Outperforms	Other Outperforms
$HmbFS_{1.5}$ vs Chi2	4	3
$HmbFS_{2.0}$ vs Chi2	6	1
$HmbFS_{1.5}$ vs Fdr	4	3
$HmbFS_{2.0}$ vs Fdr	6	1
$HmbFS_{1.5}$ vs Fpr	3	4
$HmbFS_{2.0}$ vs Fpr	6	1
$HmbFS_{1.5}$ vs Fs	3	4
$HmbFS_{2.0}$ vs Fs	7	0
$HmbFS_{1.5}$ vs Fwe	3	4
$HmbFS_{2.0}$ vs Fwe	5	2
$HmbFS_{1.5}$ vs Pca	7	0
$HmbFS_{2.0}$ vs Pca	7	0
$HmbFS_{1.5}$ vs Et	4	3
$HmbFS_{2.0}$ vs Et	6	1

Table 3.17: HmbFS performance comparison

As with any other algorithm, the performance of HmbFS can be controlled by its parameters, for instance with the Chi2 algorithm, you are required to decide how many features to keep, with HmbFS, you decide how strict you want the reduction to be, however such reduction is not linear, a Th=3.0 does not reduce double of features of Th=1.5, if features are good enough, they will say regardless the

threshold.

In this particular work, we kept the default 1.5 threshold for no-tuned comparison of algorithms, $HmbFS_{1.5}$ performed comparable to other techniques however increasing the Th to 2.0 result in very good performance across all datasets, worth noticing is that we did not tune the threshold for each individual dataset but merely selected a higher one than default to show the parameter impact in performance.

If we analyze the tables results for each dataset, we will see that feature selection does improve the performance, however, such improvement should not be considered as the only positive effect of the pre-filtering of features.

Let us take the Chowdary dataset for instance, using the full set of features, we can achieve an averaged accuracy as high as 94.3% and an individual best of 97.12% (LSVM), using $HmbFS_{2.0}$ we got an averaged improvement of 1.91%, but for LSVM individual score the improvement was only of 0.96%, which might discredit the use of feature selection. However, fewer features, or more in particular, better features makes models much more robust and simpler.

To investigate the difference in data complexity, we have plot a 2-dimension PCA analysis for the Chowdary dataset, which is shown in Figure 3.2.

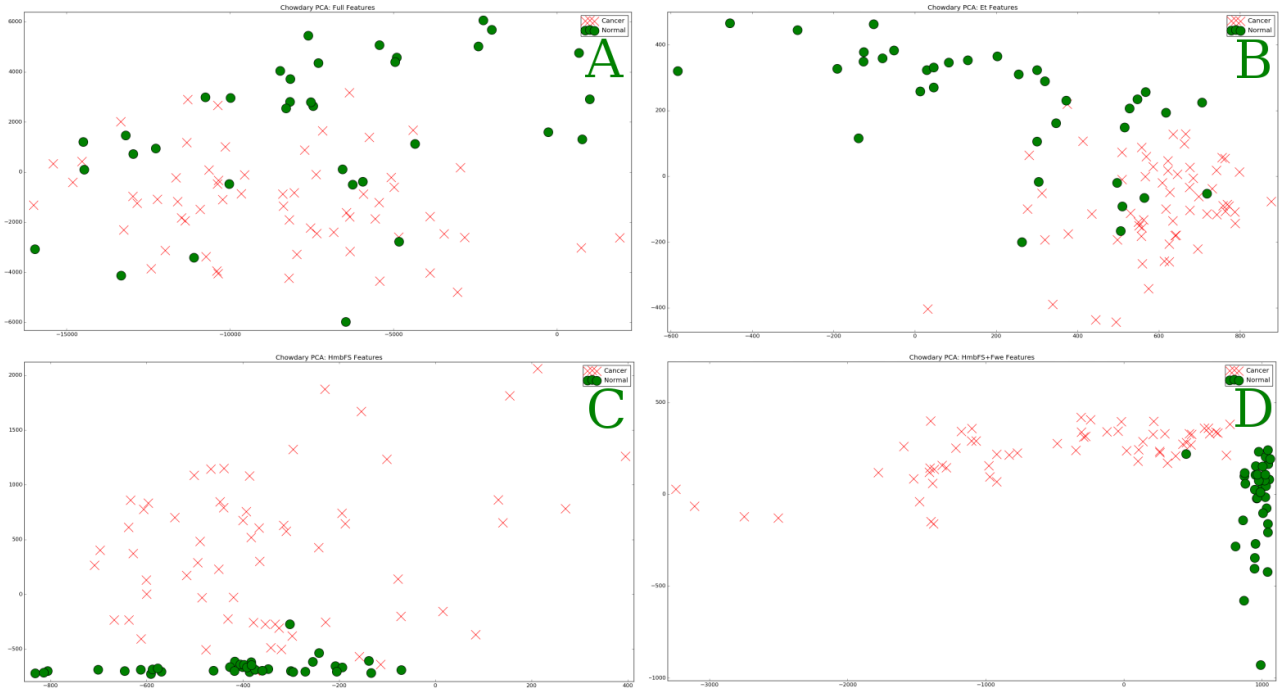

Figure 3.2: PCA plots for different versions of Chowdary dataset.

In Figure 3.2, we have four plots, A) Full Set of features, B) ExtraTrees Features, C) $HmbFS_{2.0}$

Features and D) $HmbFS_{1.5}$+Fwe,

It is clear that using the over 22,000 features the data seems very complex to separate in a 2D dimension space (section A (top left)), LSVM does an excellent job finding a way to separate the points, however such complex models can suffer from overfitting and be less useful in a real life scenario, if we pre-filter the features using ExtraTrees (Et) (section B (top right)) we get only 110 features (a 99.5% reduction) that once plotted shows a mild pattern that seems promising, however, the features selected by Et harmed a little bit the performance of LSVM, losing 0.97% in accuracy.

If we pre-filter the features using $HmbFS_{2.0}$, we get a small set of features of only 72 (a 99.7% reduction), that unlike the Et pre filter, does not harm any of the classifiers, improving as much as 2.88% for both KNN and RF, but the real improvement can be observed once we plot the data in section C (bottom left), we can see how much better clustered the data appears, considering the original dataset is over 22,000 features, the difference of 38 between HmbFS and Et is negligible, but the difference in how the data is represented is substantial.

Another outstanding result can be observed in section D (bottom right), when we pre-filter features first with $HmbFS_{1.5}$ and then pass the resulting features to a second selector Fwe, the result is only 49 features, or only 0.22% of the original dataset where the data is almost perfectly separated, such set of features are so powerful that was possible to reduce the error to less than 1% when using LSVM.

Given our results we can see that HmbFS is very competitive with other approaches, however, a key downside might be the need to tune the selection threshold. Since there is no way to know the optimal threshold value a priori, we have developed an extension to HmbFS with automatic tuning, this is discussed in the next chapter.

Chapter 4

Heat Map Based Feature Selection with CV

4.1 Introduction to HmbFScv

In the previous chapter, we showed how HmbFS can be successfully used for high dimensional reduction, however, one potential issue is the need to tune the selection threshold. In our previous experiments we can see that regardless of the threshold, HmbFS is very competitive but once tuned it can produce much better results.

In this section, we propose an automatic approach to auto-tune HmbFS with built-in cross-validation to create HmbFScv. The CV variant works in a similar fashion using two stages as before, Compression and Selection.

Compression

This stage is responsible of building a compressed version of the original dataset that is used for the actual feature selection stage, the key idea is to group continuous features (e.g., feature 1, feature 2 and feature 3) to build a RGB color pattern, the interaction between features produces different colors that are used to build a full heat map that represents a generalized version of the dataset. Since this process is scale sensitive, all features are first scaled to a 0-255 interval, using the Scikit-learn [61] library this can be achieved as follows:

```
#HmbFScv 0-255 Feature Normalization
mms = preprocessing.MinMaxScaler(feature_range=(0, 255))
new_feats = mms.fit_transform( original_feats, dataset_classes)
```

Once each feature has been normalized, they can be used to build colors for the heatmap, e.g., features values $F_1 = 242$, $F_2 = 20$ and $F_3 = 35$ will create a *bright red* color (in RGB format), while the values $F_1 = 12$, $F_2 = 5$ and $F_3 = 55$ would create a very *dark blue*. In order to build a generalized

representation, the heatmap is not built with the raw generated colors, but with a quantized version, reduced to 16 basic colors. This can be easily achieved by a distance search to find the basic color that most likely represents the real color. The following code assigns *red* and *black* for the feature-values examples we provided above.

```
#Color quantization, from real to 16 colors
def getBaseColor(self, R, G, B):
 distance=[768]*16
 for x in range(0,16):
  bC = self.baseColors[x]
  distance[x]= ( abs(bC[0]-R) + abs(bC[1]-G) + abs(bC[2]-B) )
 return distance.index(min(distance))
```

Once the color quantization is completed, the new compressed dataset is ready for feature selection analysis which is described in the next step.

Selection

Since the data is already grouped as the new compressed dataset has one third ($\frac{1}{3}$) of the original features, a multivariate analysis is inherent even when only single features (as a single feature now is represented by three of the original features) are analyzed. The main idea is that different classes must be represented by different color patterns, and the probability of a given color pattern must be greater for some class than another in order to be selected as useful. The resultant selection formula is presented below:

$$\begin{gathered} \text{Useful } F_i \text{ iff:} \\ \exists\{C_j, C_k\} \subset C \mid \\ [Pr(C_j \mid Mo(C_j \mid F_i))] > [Pr(C_k \mid Mo(C_j \mid F_i)) * T_h] \end{gathered} \tag{4.1}$$

In order to mark a feature as useful, we first find the *mode* (Mo), i.e., the most common color that a particular feature F_i exhibits for a given class C_j, then we compare if this color pattern belongs more likely to the class C_j than for a class C_k. In order to decide if the difference is significant, we use a threshold Th (default 1.5) that prevents selection of features with minimal difference. The following code shows the selection process:

```
#Search useful features
for Cj in range(0,number_of_classes):
 for Ck in range(0,number_of_classes):
  if Cj != Ck:
   if ( (max(data[Cj])/sum(data[Cj]) ) >  ( Th * (data[Ck][data[Cj].argmax()] / sum(data[Ck])))  ):
```

When the if-condition is met, that feature is marked as useful, and the process continues until all features have been evaluated. After the process is completed, the selected features are mapped to the original feature space, e.g., if features 3, 8 and 56 are selected, that means that in the original space, the selected features will be 7,8,9 (because compressed $F_3 = F_7$, F_8 and F_9), 22,23,24 and 166,167,168.

Automatic Threshold

It is clear that the threshold Th plays a very important role in the selection process, although very intuitive to tune (higher Th = more aggressive reduction), it is still the user responsibility to select a proper value, in order to fix this inconvenient, we propose a sequential stratified-cross validated search, where different threshold are analyzed with automated cross validation in a similar fashion as RFE does, with the main difference that with HmbFS, the steps in selection are not arbitrary reductions as it happens with RFEcv, where we are required to select a number of features to reduce in each step, e.g., we could try reducing 10 features each iteration, or 15, or 50, or 100, while in HmbFScv the steps are more intuitive, e.g., a $Th = 3.0$ is twice as rigorous in the selection than a $Th = 1.5$, however, if the features are good enough, changing the threshold would not lead to different selections which helps to provide more stable results. The Figure 4.1 shows the proposed architecture.

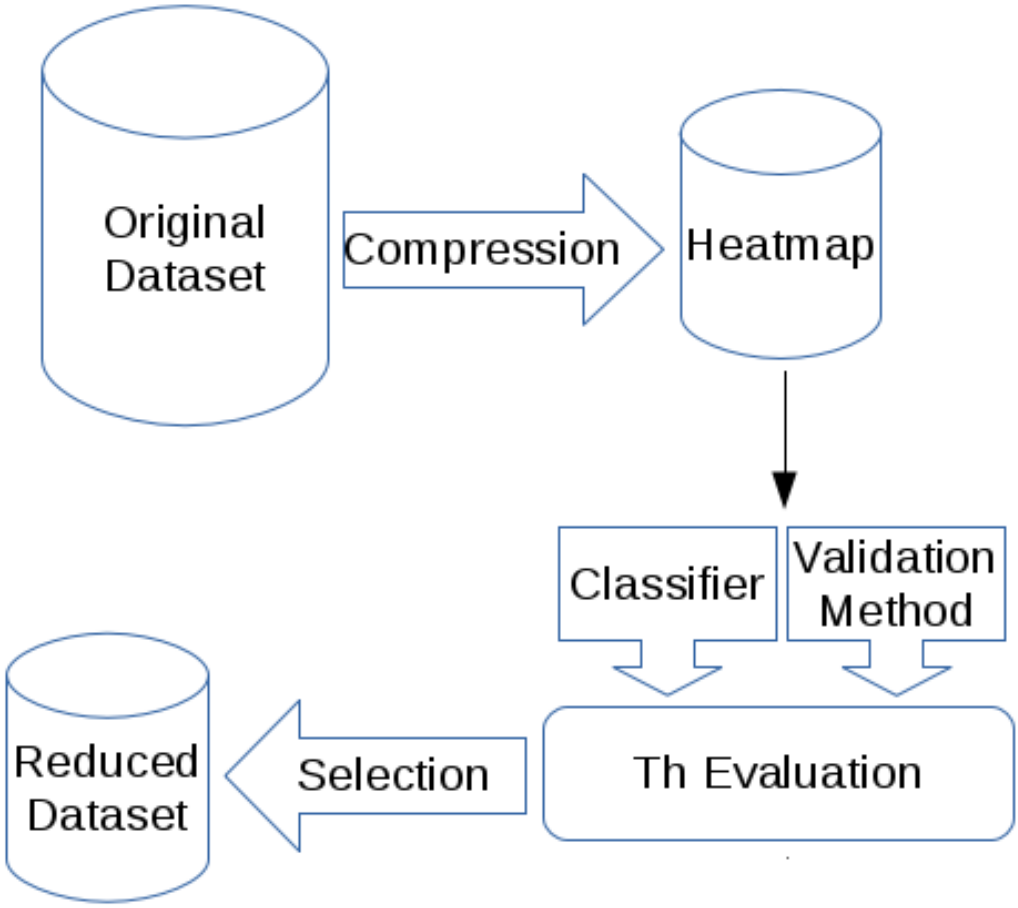

Figure 4.1: Proposed architecture for optimal threshold selection

In order to find an optimal threshold Th, we included two key components in the process, a classifier that its responsible of building models with the subset of features that HmbFS selected, and a validation method that tests the built models to ensure that the selected threshold is the optimal for the dataset. Other techniques such as RFE uses an arbitrary reduction step, such as 10% of features reduced each step, this greatly differs from HmbFS because the number of features is never required. In the following section we present some of the validation tests we performed to see how the proposed methodology improves over the default HmbFS with a fixed threshold.

4.2 Experiments

In order to test our proposal we have selected datasets that covers multiple types of cancer, specifically, we have selected a dataset from Alon [62] for colon cancer, Chowdary [63] for breast cancer, Gravier

[80] for central nervous system tumors and, finally a dataset from Tian [68] for myeloma classification.

We have performed a classification experiment using a stratified 10-fold approach, the idea behind stratified splits is to penalize the algorithms if results are biased towards the majority class. We evaluated an online classifier: Passive Aggressive (PA) [77], a classifier with built-in feature selection: Random Forest (RF) [78], and a classifier commonly used in related literature: Support Vector Machines (SVM) with linear kernel [76]. The feature selection stage was accomplished by HmbFS (the original version, with $Th = 1.5$), HmbFS with automatic threshold selection and cross-validation (this work, HmbFScv) and comparison with a popular approach Recursive Feature Elimination with cross-validation (RFEcv), the cross-validation stage was carried out by Logistic Regression, which is not included in the evaluation results to avoid overfitting, every classifier was executed without feature selection, and later with the 3 different approaches to evaluate the usefulness of each approach, the mean accuracy for every experiment is reported in Table I to IV.

Table 4.1: Alon Dataset Accuracy(%)

Methods	No FS	HmbFS	HmbFScv	RFEcv
Features	2,000	2,000	1,733	1,800
PA	77.1	77.1	77.1	77.1
RF	74.0	74.0	81.9	74.8
SVM	82.1	82.1	82.1	82.1
Average	77.7	77.7	80.4	78.0

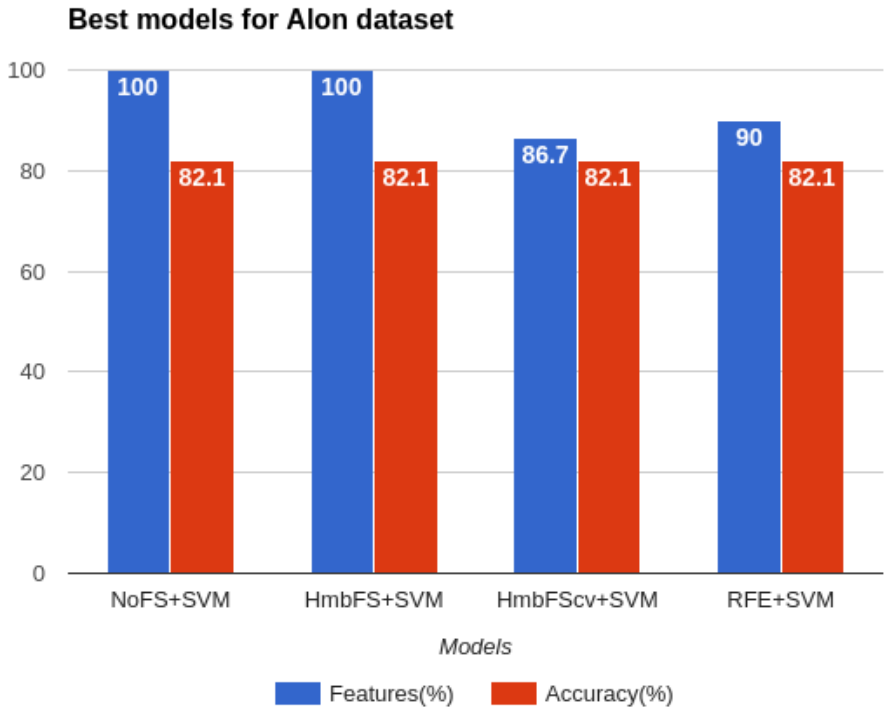

Figure 4.2: Comparison of best models for the Alon dataset

The Alon dataset, being the lowest dimensionality of the datasets reviewed presented a great challenge for the feature selection algorithms, for instance our original proposal HmbFS, was unable to remove any feature, leaving all the original 2,000 features and therefore producing the exact same results as if no feature selection (No FS) would have been employed, however, with the modified version (HmbFScv), it was possible to reduce over 10% of the features, not only without loss of precision, but with a huge improvement in stability, as the Random Forest (RF) performance was increased from 74.0% to 81.9%. In the case of RFEcv, the reduction of features is very similar to HmbFScv but the improvement is less stable as the RF performance did not improve as much. However, as an overall overview as it can be seen in Fig. 4.2, for the Alon dataset, feature selection did not improve the best possible score, which was achieved by Support Vector Machines (SVM) without any feature selection, however, it is worth noting that a reduced set of features helps to understand the data as an added benefit, even if no classification performance is achieved.

Table 4.2: Gravier Dataset Accuracy(%)

Methods	No FS	HmbFS	HmbFScv	RFEcv
Features	2,905	171	171	1
PA	70.8	73.8	73.8	75.0
RF	68.9	70.9	70.9	63.6
SVM	75.1	73.2	73.2	74.5
Average	71.6	72.6	72.6	71.0

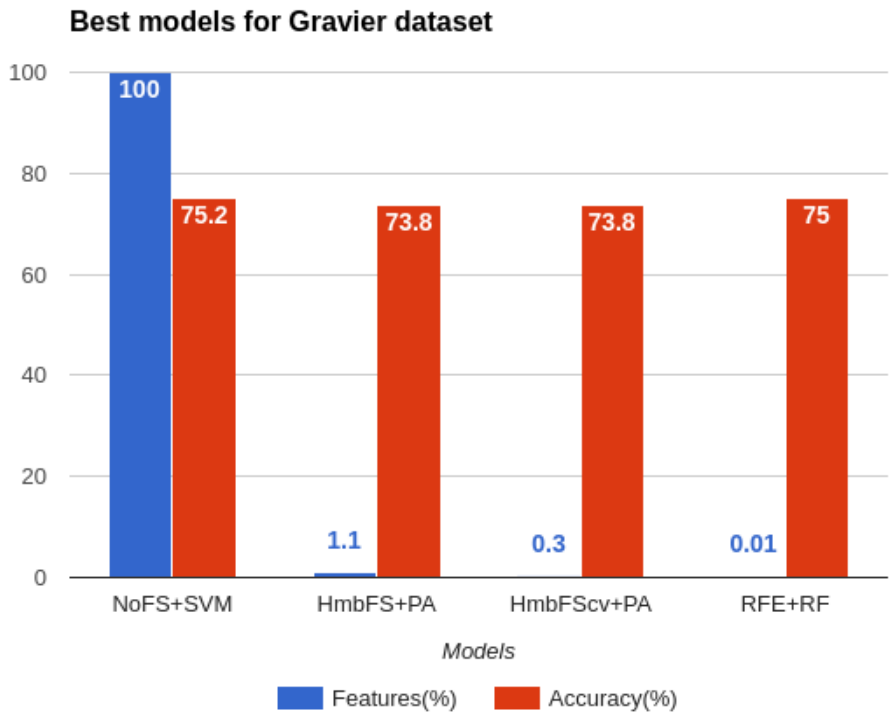

Figure 4.3: Comparison of best models for the Pomeroy dataset

The Gravier dataset, having the highest dimension of the tested group, produced the most remarkable reductions in terms of features, however, it also shows an interesting behavior for RFEcv that causes an excessive reduction of features when the data is too noisy. From Fig. 4.3, we can see that the features bars are barely visible, which is expected as our original HmbFS reduced the over 22,000 features to only 234 while still producing improvements in prediction power for all of the tested classifiers, most notable case was for RF, where there is almost a 10% improvement. In the case of HmbFScv, the reduction goes even further to just 57 features, which confirms that indeed the dataset is extremely noisy, the top model remained at the same prediction power than using HmbFS (97.3%), however the

RF improvement was not as much in this case. For RFEcv, is worth noticing, that with only 3 features, which represents 0.01% of the original space it was possible to achieve an outstanding 93.6% with RF, however the other two algorithms (PA & SVM), were unable to converge to a successful solution lowering the predicting power to as low as 63.6% for SVM.

Table 4.3: Chowdary Dataset Accuracy(%)

Methods	No FS	HmbFS	HmbFScv	RFEcv
Features	22,283	234	57	2222
PA	94.3	97.3	91.3	94.3
RF	88.5	97.2	94.3	97.1
SVM	95.3	96.3	97.3	97.3
Average	92.7	96.9	94.3	96.2

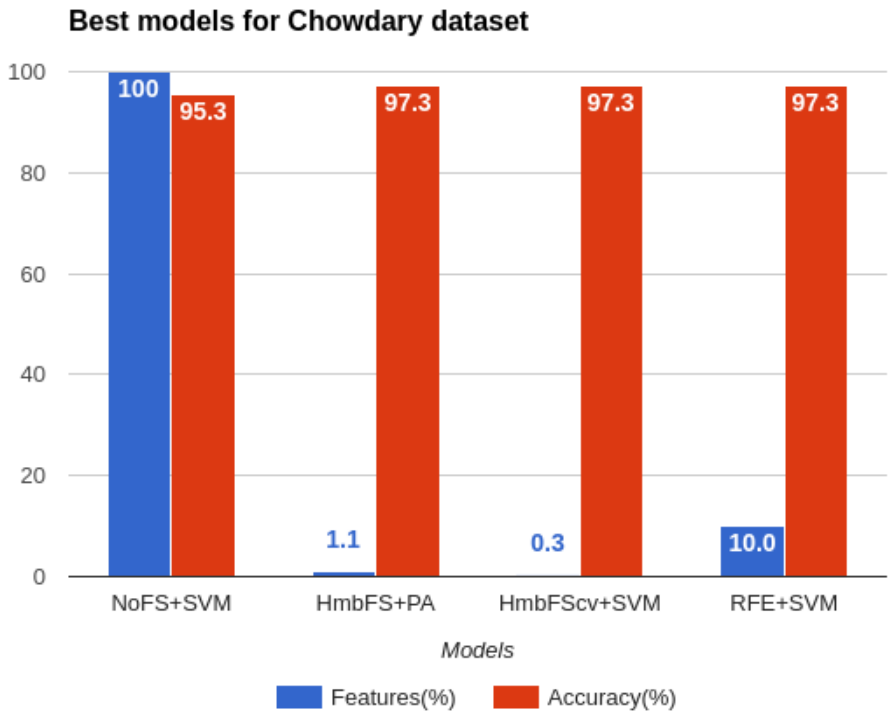

Figure 4.4: Comparison of best models for the Chowdary dataset

The Chowdary dataset, was the hardest to correctly classify, it is worth noting that RFEcv again overly reduce, however in this case, its best model 68.7% was achieved by SVM and only 1 feature, such outstanding result outperforms any of the other models achieved when the full set of features was

used. The best overall model however was possible thanks to HmbFS which achieved 74.3% with SVM while HmbFScv came close with 72.3% with the less complex Passive Aggressive (PA), the loss in accuracy performance is in part caused by the reduced set of features, as HmbFScv models are built using almost half the features (2,217 vs 1,121) than HmbFS. It is important to remember that although below 80% in accuracy performance can be considered low, is a very important improvement over the same data but without feature selection, and the reduced subspace can help with data insights as well. In the Fig. 4.4, we can see how both HmbFS and HmbFScv outperforms the other models in the comparison.

Table 4.4: Tian Dataset Accuracy(%)

Methods	No FS	HmbFS	HmbFScv	RFEcv
Features	12,625	3,609	2,931	1
PA	64.4	71.1	68.6	60.5
RF	75.9	78.7	79.8	68.3
SVM	72.6	77.6	77.0	50.1
Average	71.0	75.8	75.1	59.6

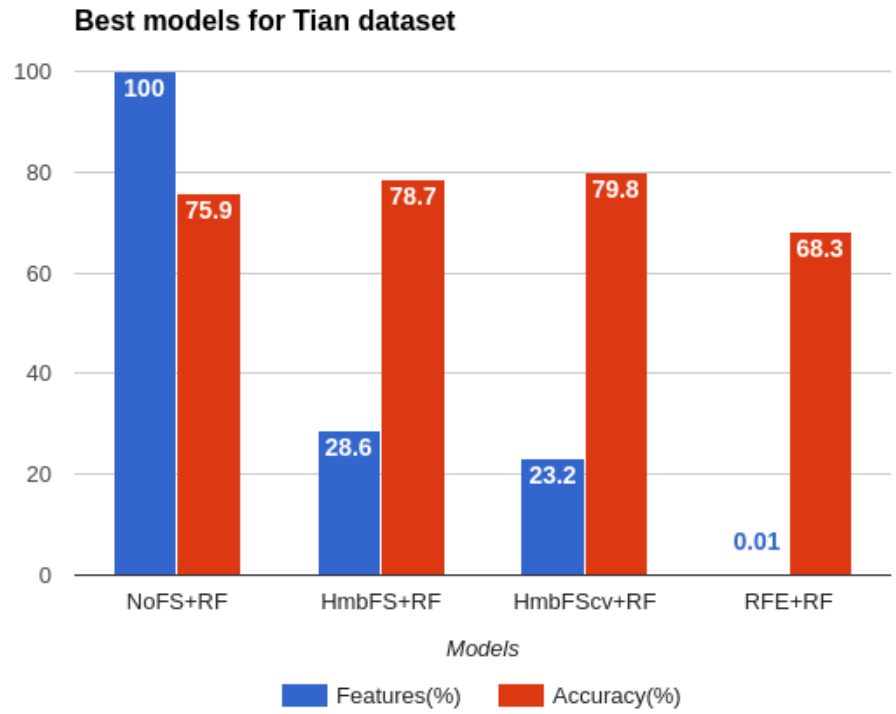

Figure 4.5: Comparison of best models for the Tian dataset

The Tian dataset being the second in terms of dimensionality allowed great reductions to be performed, starting with a reference benchmark of 75.9% accuracy with RF and no feature selection, it presented a considerable challenge, however HmbFS managed to outperformed that benchmark for very algorithm reaching a maximum of 77.5% with SVM with less than 30% of the original features, following the same path, with the improved HmbFScv it was possible to reduce even more to around 23% of the original features and producing an even better model at 79.7% with RF. In the case of RFEcv, the over-reduce effect seems to be an expected behavior in very noisy datasets, in some cases it produce very good results, but for the Tian dataset it impacted too much to the classifiers, getting as low as 60.5% in the case of PA. In the Fig. 4.5, we can see a behavior very similar as the Chowdary dataset, where HmbFScv usually is more aggressive with the reduction while maintaining a similar performance.

We completed a total of 48 experiments to test HmbFS and HmbFScv (automatic Th selection), each dataset was tested under 12 different setups with combinations of feature and learning algorithms. among the results, we can see that as the dimension increases, so does the potential noise, the Alon dataset with 2,000 features was very difficult to reduce, however, Pomeroy dataset with over 22,000 lead us to huge reductions overall. The most significant reductions were for RFEcv in the Chowdary dataset, that with just one feature, it improved over the 7,128 of the original space, and the other impressive reduction goes for HmbFScv in the reduction from 22,283 to only 57 features using the Pomeroy dataset. In terms of predictive power improvement, HmbFScv managed to get the best overall model for 3 of the 4 datasets, only losing to HmbFS (the original) in the Chowdary dataset, improvements can be as big as almost 10% such as in Pomeroy, or as low as no improvement, as was the case with the Alon dataset, where no model improved over the default 82.1%, however, it is important to remember, that even small improvements (1%) can be significant in datasets so noisy, and if such improvement is achieved with a lower number of features, it can lead to more stable models as well as enhance scientist insights about the data.

Even when we can automate the threshold selection, there are situations where it is needed to get the relative feature importance of features, for this scenario the ranker version of the algorithm has been created and evaluated. Details are discussed in the next chapter

Chapter 5

Heat Map Based Feature Ranker

5.1 Introduction to HmbFR

A core part of our algorithm is working with heat maps, which is commonly used in high dimensional spaces such as genomic data to unravel hidden patterns, in recent studies we can see how heatmaps are being used to understand data structure [81] so for this work we use an internal heatmap generation to aid in the search of regions of interest.

The idea behind Heat Map Based Feature Ranker (HmbFR) is to estimate the predictive power of a feature given its association with others, an exhaustive feature combination is avoided under the rationale that high dimensional data usually keeps an intrinsic structure. Even if this order does not apply to all datasets, we consider the evaluation of features in groups of continuous features as a useful approach for high dimensional data.

The algorithm is composed of two main stages: compression and ranking. The core idea is to build groups of features and later evaluate their predictive power as a group to estimate individual feature power.

In compression stage, we build the heat map, which is a color representation of the data, in order to build it, a color model is required, since our design uses groups of 3 features, the RGB model is a perfect fit as it requires three channels, therefore 3 features F are used to build 1 group G of the form $G_i = \{F_r, F_g, F_b\}$, these elements stand for $Feature_{Red}$, $Feature_{Green}$ and $Feature_{Blue}$ and the relation between a particular group of features creates a pattern, such pattern is mapped to a color in a virtual heatmap, and this color is what is further analyzed in order to decide if the group of features should be kept. These groups are built by continuous features, in two stages, forward and backwards, so in the original space F_1, F_2 and F_3 becomes G_1 for the forward stage and, for N features, the backward stage G_1 is represented by F_n, F_{n-1} and F_{n-2}

The first step for compression is the normalization isolated features, each of them is treated independently, to formalize this, let I be the full set of instances and I_x a particular sample from the whole dataset D. Each value F_i that belongs to instance I_x (I_xF_i) is then normalized in a 0 to 255 interval as shown in 5.1:

$$\begin{gathered} \forall I_xF_i \in D : \\ \widehat{I_xF_i} = \left(\frac{I_xF_i - min(FI_x)}{max(FI_x) - min(FI_x)} \right) * 255 \end{gathered} \tag{5.1}$$

Where min(IF_i) and max(IF_i) represents the minimum and maximum value respectively that a feature gets across all the instances of the dataset. This process is repeated for every feature F_i across all instances I_x. Once the features have been normalized, each of the groups containing F_r, F_g and F_b can be mapped to a true color expressed in red/green/blue (RGB) format, together the three features can represent up to 16,777,216 different colors or patterns. However, since the idea is to build a generalization of the original data, we apply a technique called *Color Quantization* that allows the mapping of a true color to a lower depth color scale, in this case, we reduce each true color to a 4-Bit 16-Colors scheme. The idea is to discard small data variations or uniqueness, and produce a new set of data which is more general and consistent, the process of quantization is performed by looping over the 16 reference colors and find the minimum Euclidean distance with the true color as this approach is very popular for the task and it has been found to produce satisfactory results [82]. The RGB values for each of those colors are the standard values defined in the HTML 4.01 specification [1].

To formalize the information quantization, let G be the set of all the built groups and G_i a particular group (which is composed by F_r, F_g and F_b) that belongs to a instance I_x, each of those I_xG_i combinations are then compared against all the reference colors R_j in set R to produce the new single-value compressed feature F_i that will represent the original 3-feature as shown below:

$$\begin{gathered} \forall I_xG_i \in D \; and \; \forall j \in \{1, ..., 16\} : \\ I_xF_i = min \left(\sqrt{\begin{array}{r} (I_xG_iF_r - R_jF_r)^2 + \\ (I_xG_iF_g - R_jF_g)^2 + \\ (I_xG_iF_b - R_jF_b)^2 \end{array}} \right) \end{gathered} \tag{5.2}$$

Once all the distances between true color and reference color have been calculated, we select the reference color R_j with the minimum Euclidean distance, and this process gives origin to a new dataset that is purely built in reference colors, or in other words, it is a compressed lossy version of the original.

Dataset reduction is possible due to the fact that the reference color is represented by a single

[1] https://www.w3.org/TR/html4/types.html#h-6.5

value (e.g.: red, which is composed of RGB values of 255,0,0) instead of three different features, this compression makes possible to reduce the original dimension of any dataset to a third with very minimal loss of information, however although there is indeed a loss, this can be negligible as the data transformation is only used for the feature selection process, and the original information suffers no changes.

After compression is completed, ranking relevant features is based on the rationale that different classes should look different, hence their associated quantized colors should look different as well. The ranking occurs in the new compressed dataset, the process sees regular features F_i although we know they represent a group in the original space.

Since we are using a 16 color scheme to build the heatmap, each feature can have as much as 16 possible values (in the compressed space), to estimate its predictive power, we iterate over the color spectrum searching for different ratios in the conditional probability to find a color given a reference class. The score for each feature is then calculated as follows:

$$HmbFR_{score}(f_i) = \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{r=1}^{C} \Big(p(C_r \mid K_i) - p(C_r \mid K_j) \Big) \tag{5.3}$$

Where K represents the number of classes and C is the color spectrum size, in our proposed design we use 16 base colors. The idea is to iterate all classes looking for the conditional probability of getting a color r given a class i or j, the probabilities are subtracted to account for class imbalance.

After all features have been ranked, we need to restore the ranking to the original space as all the process was performed on the compressed data, however, the mapping process is very efficient as the groups were formed from continuous features, e.g., in the compressed space $F_i = \{2\},\{5\},\{7\}$ and $\{10\}$ are mapped to the original space to $F_i = \{4,5,6\},\{13,14,15\},\{19,20,21\}$ and $\{28,29,30\}$. Below we present the algorithm pseudo-code:

To evaluate the usefulness of HmbFS we have prepared a series of tests through different datasets and compare multiple techniques for feature selection. In the next section, we discuss more in depth the results.

5.2 Experiments and Results

In order to estimate algorithms performance in high dimensional data, it is very important to find their order of complexity as some algorithms might not scale well, this results in useless approaches once the feature space grows out of a manageable threshold. Some algorithms exhibit orders of n^2 or even

Algorithm 2: HmbFR, Heat map based Feature Ranker

Data: Dataset with full number of features
Result: Subset of best predictive features

Normalize Data 0-255;
while *there are more features* **do**
    \\Compression
    for F_i, F_{i+1}, *and* F_{i+2} **do**
        Build group G_i with features as 16-Bit RGB values;
        Save quantized 4-bit version of $G_i \rightarrow QG_i$;
    end
    Increase i by 3 ;
end
for *each group* QG_x **do**
    \\Ranking
    for *each class* K_i **do**
        for *each class* K_j *!=* K_i **do**
            for *each color* C_r **do**
                $QG_x = QG_x + \Big(Pr(C_r \mid K_i) - Pr(C_r \mid K_j)\Big)$
            end
        end
    end
end

n^3, resulting in a totally infeasible option for anything but very low dimensional spaces [83]. We have carried out experiments to find out how the increase of classes, features, and instances have an impact in algorithms performance. In Table 5.1 we present the scenarios we tested.

Table 5.1: Test scenarios for algorithm order behavior

Growth	Instances (n)	Features (p)	Classes (K)
Class	100	100	2, ..., 10
Instance	100, ..., 1000	100	2
Feature	100	100, ..., 1000	2

We first analyzed how an increase in the number of classes affects the scalability of each algorithm, fixing the other parameters (instances and features) we increased only the classes, starting in 2 up to 10 to see the behaviour. The Figure 5.1 shows the results.

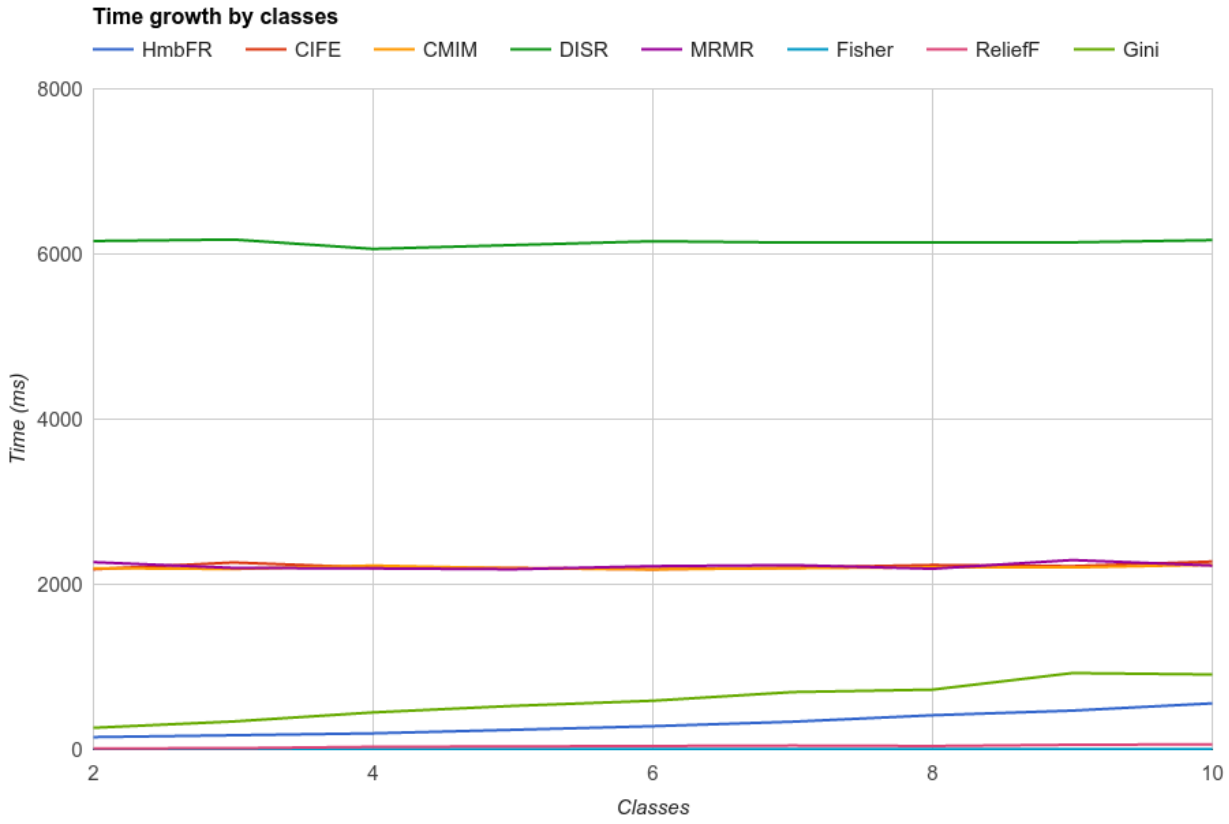

Figure 5.1: Scalability behavior for class growth

As we can see in Figure 5.1, in terms of scalability due to increased number of classes, all reviewed algorithms handle the task very efficiently, at least complexity-order wise, we can identify 4 groups, DISR, which is not affected by class increase, but it's quite slow, at least three times slower than other approaches such as CIFE, CMIM and MRMR all of which are not affected by class increase neither. A third group can be identified, Gini and HmbFR, both being slightly affected by class growth but not enough to fall even to a linear order scenario, finally a fourth group composed by ReliefF and Fisher, both performing very fast with no compromise in class size.

Next we continue with the analysis of sample growth.

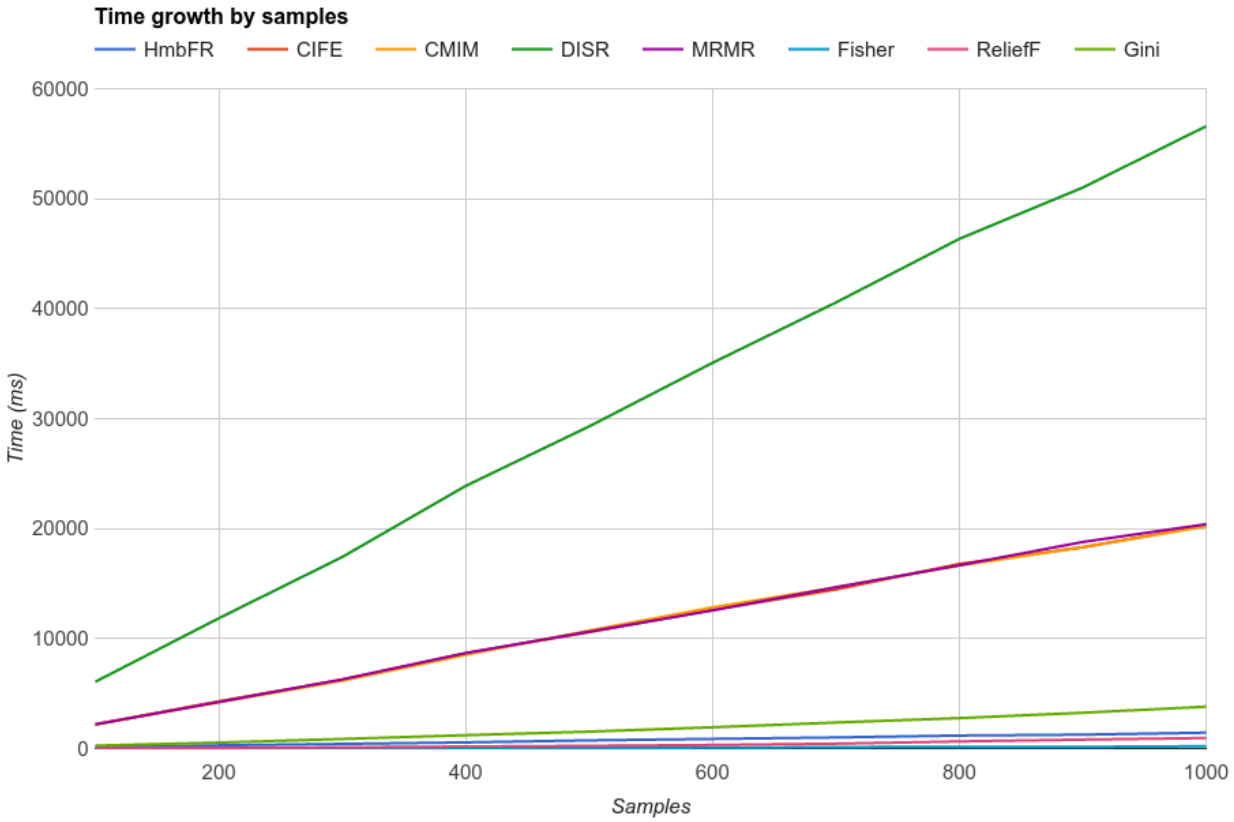

Figure 5.2: Scalability behavior for samples growth

As far as samples increase, we can see in Figure 5.2 most algorithms behave in a linear fashion, although some special cases need attention, once again DISR tops the time chart performing almost 300 hundred times slower than the fastest of the group, even scaling in a linear fashion. In terms of scalability issues, the worst scenario was for ReliefF which scaled at n^2 but since it performs very fast, it will require a very large number of samples to make it unfeasible.

Next, we review the case of features increase.

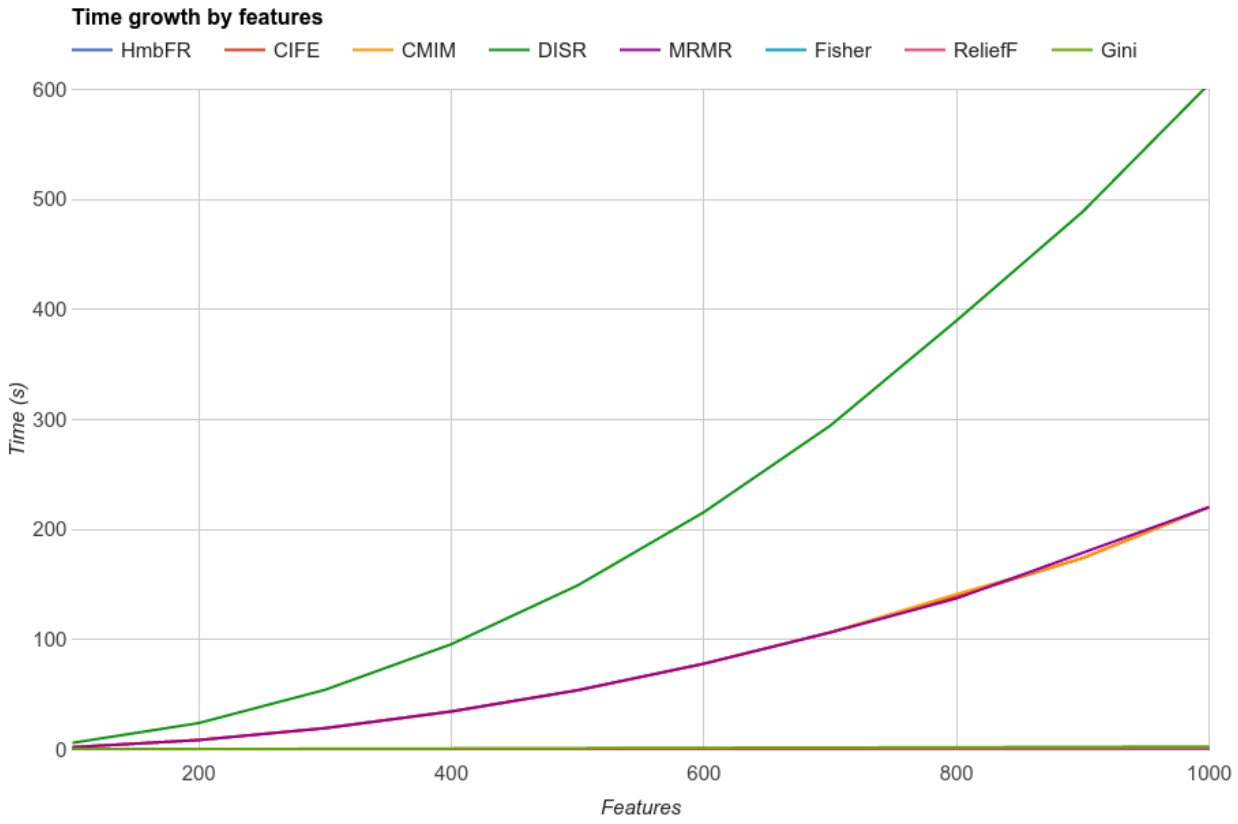

Figure 5.3: Scalability behavior for features growth

In Figure 5.3 we can see the real issue with most algorithms, they scale at n^2 as the number of features increases, from the eight algorithms reviewed, half of them failed to keep a linear run time, only HmbFR, Fisher, ReliefF and Gini achieved to scale properly, hence they represent the only feasible option for real world test scenarios, where the number of features can easily reach 20,000 features or more. Table 2 summarizes this scalability evaluation.

We can see that half of the algorithms (CIFE, CMIM, DISR and MRMR) do not scale well as the number of features increases, with an $O(n^2)$ order, those algorithms are really not suitable for high dimensional data. For our practical comparison we have selected the remaining four (HmbFR, Fisher, ReliefF and Gini), in Table 3 we show the characteristics of the benchmark datasets.

In our experiments, most dataset easily exceed a thousand features, notable exception is the USPS dataset with only 256, but enough samples to cause troubles for algorithms such as ReliefF that scales quadratically with the number of samples. The need for more efficient algorithms is noticed more clearly when we consider high dimensional datasets such as GLI-85, with 22,283 features, trying to perform feature selection with algorithms such as DISR would be an infeasible task. We also include

Table 5.2: Summary of scale behavior

	Order		
Algorithm	Classes	Samples	Features
HmbFR	$O(n)$	$O(n)$	$O(n)$
CIFE	$O(1)$	$O(n)$	$O(n^2)$
CMIM	$O(1)$	$O(n)$	$O(n^2)$
DISR	$O(1)$	$O(n)$	$O(n^2)$
MRMR	$O(1)$	$O(n)$	$O(n^2)$
Fisher	$O(1)$	$O(n\log n)$	$O(n)$
ReliefF	$O(n)$	$O(n^2)$	$O(n)$
Gini	$O(n)$	$O(n\log n)$	$O(n)$

Table 5.3: Benchmark datasets details

Dataset	Classes	Samples	Features	Feat/Smp	Anomaly
Carcinom	11	174	9,182	55.8	0.330
Colon	2	62	2,000	32.3	0.485
GLI-85	2	85	22,283	262.1	0.234
Glioma	4	50	4,434	88.7	0.324
Leukemia	2	72	7,070	98.2	0.666
Lung	5	203	3,312	16.3	0.281
Orl-raws-10p	10	100	10,304	103.1	0.320
Prostate-ge	2	102	5,966	58.5	0.255
USPS	10	9298	256	0.03	0.252
Warp-pie-10p	10	210	2,420	11.5	0.296

the feature to samples ratio to give an idea how disparity those values are, generally we would want more features if and only if those features are useful, otherwise, the less noise the better. To estimate how noisy the dataset is, we include the anomaly score as reported by IsolationForest [84] algorithm.

The benchmarking process consists of performing feature ranking for each dataset followed by a Stratified 10-fold cross validation that ensures class imbalance is considered in every fold. The whole process is repeated 10 times, every time using 10% fewer features (removing the less powerful ranked), this is done for 3 different classifiers as some datasets might favor a given learner. In total 300 setups are evaluated for every dataset using an F1 weighted (i.e, for each class) score to consider precision and recall in a single metric as Equation 5.4 shows:

$$F1 = \frac{2 * (precision * recall)}{(precision + recall)} \tag{5.4}$$

The Scikit-learn classifiers, as well as running parameters are being shown in Table 5.4:

Table 5.4: Classifiers setup

Classifier	Parameters
Naive Bayes	Not Required
Logistic Regression	class_weight='balanced', max_iter=200
Random Forest	n_estimators=Variable, class_weight='balanced'

The only parameter we carefully tuned was the number of estimators (or trees) for RandomForest using the formula in 5.5:

$$n\_estimators = 10 + \lceil \sqrt{\#_{feats}} \rfloor + \lceil \sqrt{\#_{Samples}} \rfloor + \#_{Classes} \tag{5.5}$$

The rationale behind variable number of trees is that bigger and complex data sets usually need a higher number of trees, however, a higher number can be counterproductive for smaller scenarios and could lead to overfit.

In the following section, we present our analysis for each of the benchmarked dataset, as well as statistical significance analysis, this is done by comparing to a progressive linear reduction (PLR) algorithm, which only reduce features with no analysis involved, just linearly select a given amount. A t-test is performed against the PLR algorithm and the p-value is reported.

After feature reduction is completed, we take advantage of the smaller dimensionality dataset to enhance data representation. Given our reduced noise, we now use a 3 PCA vector representation that is being feed to an emergent self organized map (ESOM) [85] which will find a nonlinear representation in an unsupervised way that should provide an extra insight of dataset structure after noise has been reduced using only Top 10% of features as reported by HmbFR.

Carcinom

The Carcinom dataset has the most number of classes of our tests with 11, and with the only exception of the class 0, all others are nonlinearly separable in our 2D PCA representation as it can be seen in Figure 5.4.

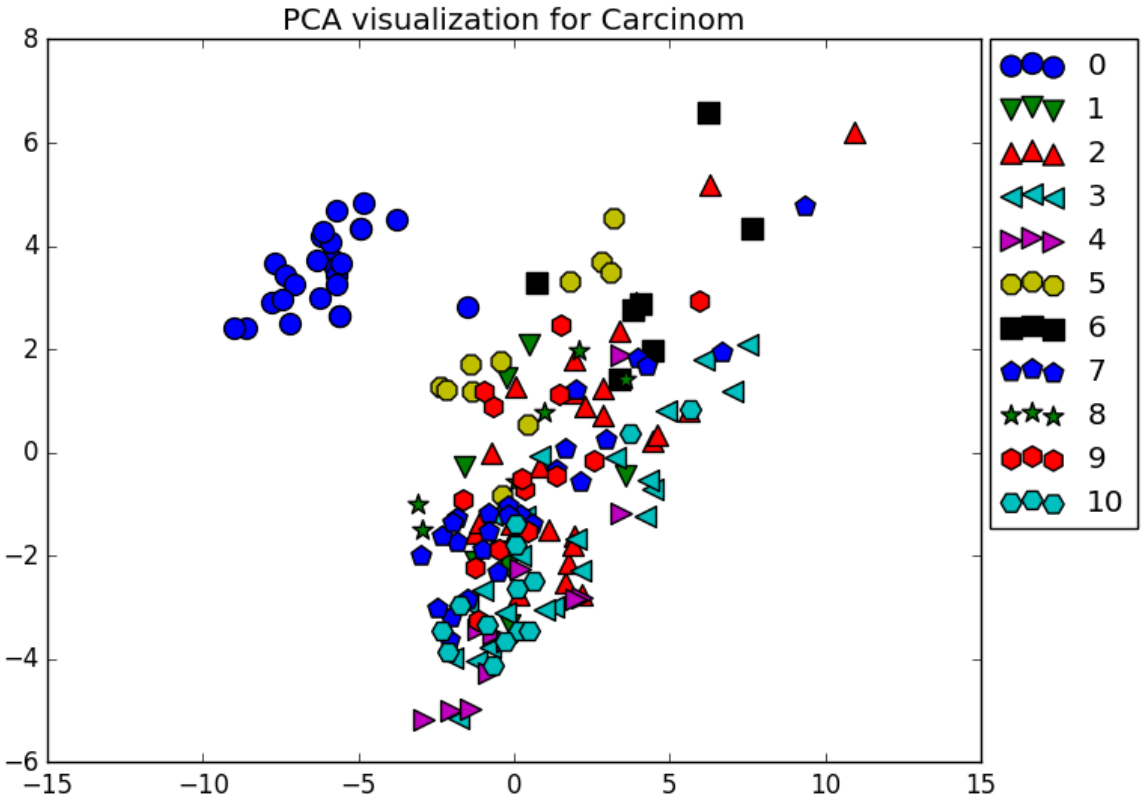

Figure 5.4: Carcinom Visualization

For this particular dataset Logistic Regression performed the best, achieving better performance across all number of selected features. Starting with a baseline score of $F1 = 0.947$ it was possible to increase the performance to 0.98 and 0.968 with Fisher and HmbFR respectively, both algorithms produce their best result with only 10% of the original features. On the other hand, ReliefF got a poor performance while Gini was not able to produce an acceptable p-value, therefore the results could fall under a lucky scenario.

Table 5.5: Logistic Regression for Carcinom

Num. Features	HmbFR	Fisher	ReliefF	Gini	PLR
9182	0.947	0.947	0.947	0.947	0.947
8264	0.947	0.955	0.939	0.929	0.929
7346	0.953	0.960	0.944	0.919	0.939
6427	0.944	0.970	0.944	0.929	0.939
5509	0.955	0.972	0.936	0.938	0.903
4591	0.951	0.977	0.936	0.945	0.918
3673	0.960	0.977	0.941	0.936	0.917
2755	0.960	0.980	0.951	0.929	0.923
1836	0.960	0.980	0.951	0.918	0.910
918	0.968	0.980	0.957	0.916	0.894
Average	0.955	0.970	0.945	0.931	0.922
Peak	0.968	0.980	0.957	0.947	0.947
PLR p-value	0.002	0.000	0.005	0.142	NA

Using only 10% of the original features, it was possible to improve in 2% the baseline model, although at minimal improvement, we can see in Figure 5.5 that the ESOM plane makes much better job at separating the instances while still maintaining a similar structure, for instance, classes No. 5 and 0 are very similar spaced as they were in the original PCA plot but now have better margins vs other classes.

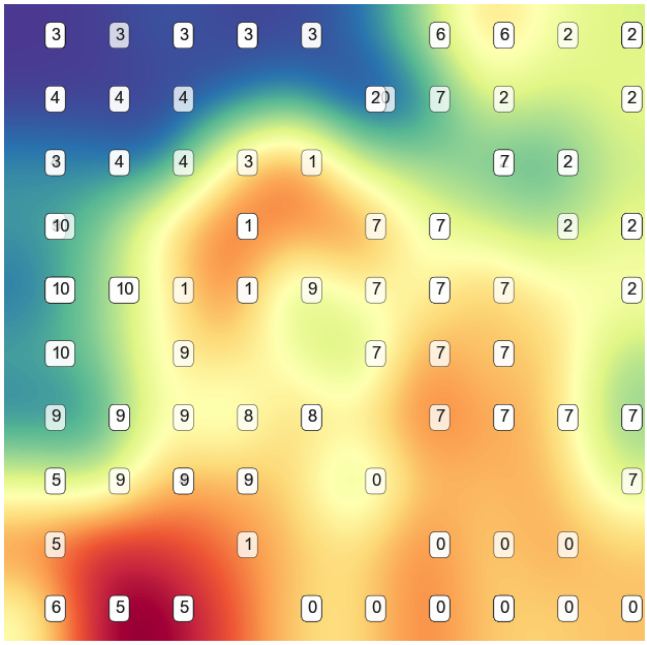

Figure 5.5: Carcinom reduced features SOM plane

Colon

The colon dataset looks simple to separate, a single decision tree might do a very good job on its own, and with only two classes its not surprise that Random Forest performed the best in this scenario.

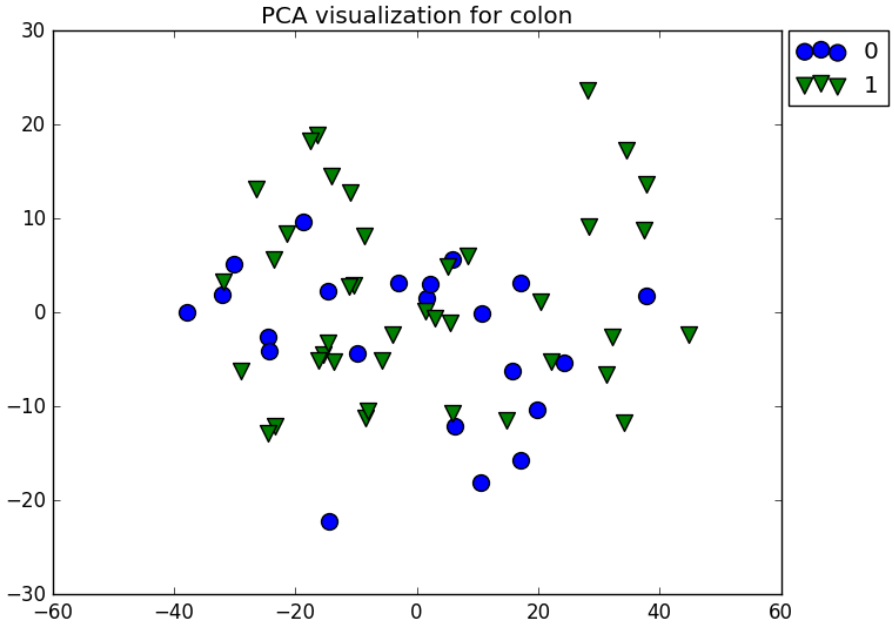

Figure 5.6: Colon Visualization

An interesting result arise from the colon dataset, all algorithms achieved a very high p-value, meaning that for this dataset with so few features (2000), feature reduction might be trivial, ReliefF and HmbFR achieved the best peak result with 0.865 and 0.851 although HmbFR seems a bit more stable with a better average score. Fisher and Gini performed poorly in this task with a score even lower than PLR.

Table 5.6: Random Forest for Colon

Num. Features	HmbFR	Fisher	ReliefF	Gini	PLR
2000	0.817	0.817	0.817	0.817	0.817
1800	0.817	0.790	0.819	0.803	0.786
1600	0.790	0.789	0.774	0.815	0.786
1400	0.768	0.803	0.774	0.776	0.790
1200	0.833	0.803	0.818	0.771	0.848
1000	0.831	0.833	0.814	0.831	0.831
800	0.846	0.836	0.833	0.832	0.831
600	0.851	0.832	0.832	0.814	0.790
400	0.850	0.818	0.865	0.818	0.801
200	0.846	0.821	0.865	0.832	0.859
Average	0.825	0.814	0.821	0.811	0.814
Peak	0.851	0.836	0.865	0.832	0.859
PLR p-value	0.245	0.963	0.474	0.762	NA

With a 3% improvement in model performance, and after removing 90% of the original features, the visualization of the Colon dataset is dramatically improved by the ESOM plane, while the PCA plot hardly separates the classes, the new representation shows a clear separation as seen in Figure 5.7.

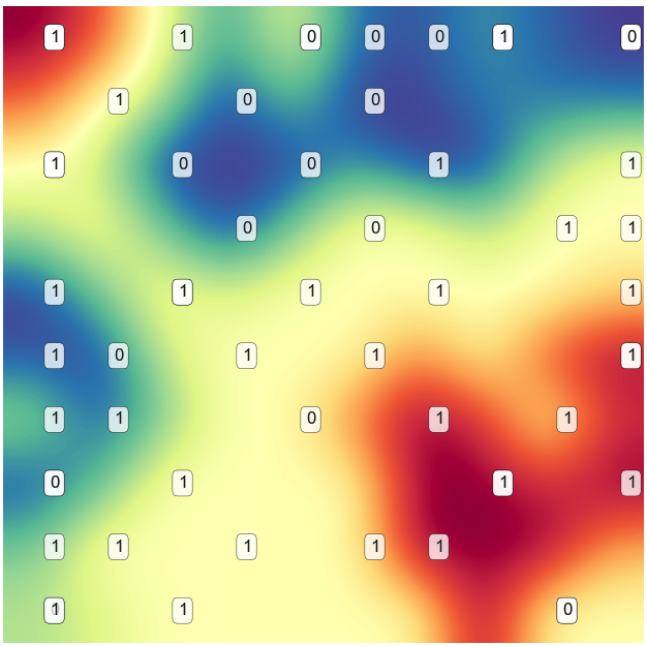

Figure 5.7: Colon reduced features SOM plane

GLI-85

The GLI-85 dataset has the most potential for feature reduction, given the over 22,000 features and the dataset characteristics, it seems some parts of the data are linearly separable as we can see in its PCA visualization.

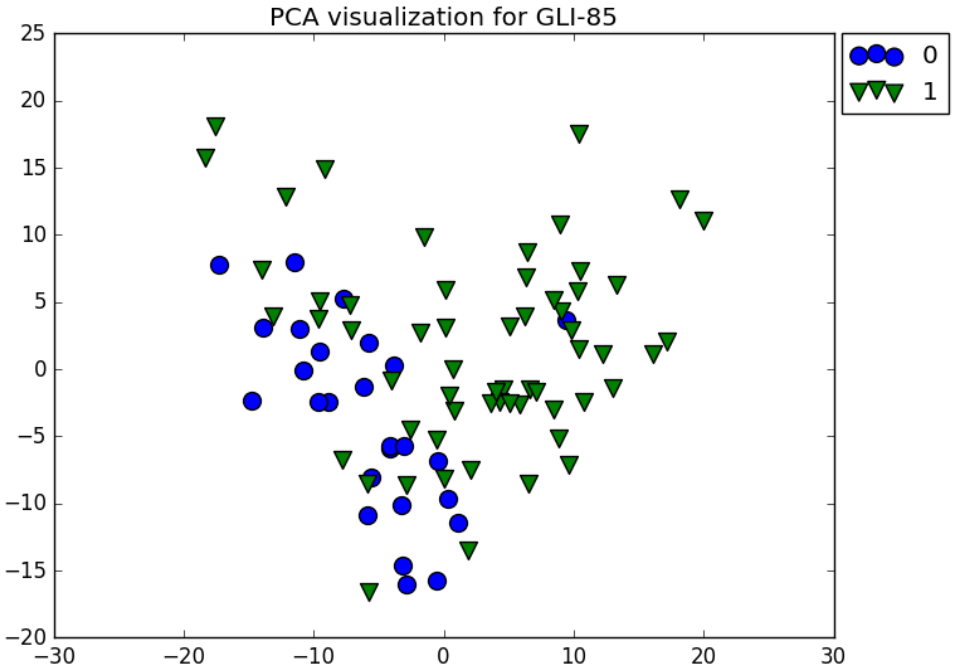

Figure 5.8: GLI-85 Visualization

As expected from the PCA analysis, all algorithms seems to be improving classification performance when Logistic Regression is used, the reductions are very good, using only 10% of the original features ($\approx 2k$ vs $\approx 20k$) seems to improve by as much as 4% with HmbFR or 5% by Fisher and ReliefF. In this particular dataset our approach did not succeed in improving over other techniques, however, it remains very competitive and a low p-value gives confidence on its performance.

Table 5.7: Logistic Regression for GLI-85

Num. Features	HmbFR	Fisher	ReliefF	Gini	PLR
22283	0.886	0.886	0.886	0.886	0.886
20055	0.886	0.900	0.897	0.900	0.886
17826	0.886	0.911	0.900	0.900	0.886
15598	0.897	0.911	0.911	0.900	0.897
13370	0.912	0.912	0.911	0.912	0.886
11142	0.886	0.912	0.911	0.912	0.886
8913	0.912	0.912	0.923	0.912	0.886
6685	0.912	0.936	0.936	0.925	0.897
4457	0.925	0.936	0.936	0.925	0.911
2228	0.925	0.936	0.925	0.925	0.878
Average	0.903	0.915	0.914	0.910	0.890
Peak	0.925	0.936	0.936	0.925	0.911
PLR p-value	0.033	0.001	0.001	0.001	NA

Using only 10% of the original features helped to boost the model performance 4%, the dataset is a bit unbalanced in favor of positive class and so can be noticed for both the PCA and the ESOM plots. However, after reduction, we can see a much clearer representation, although some outliers are still present as seen in Figure 5.9.

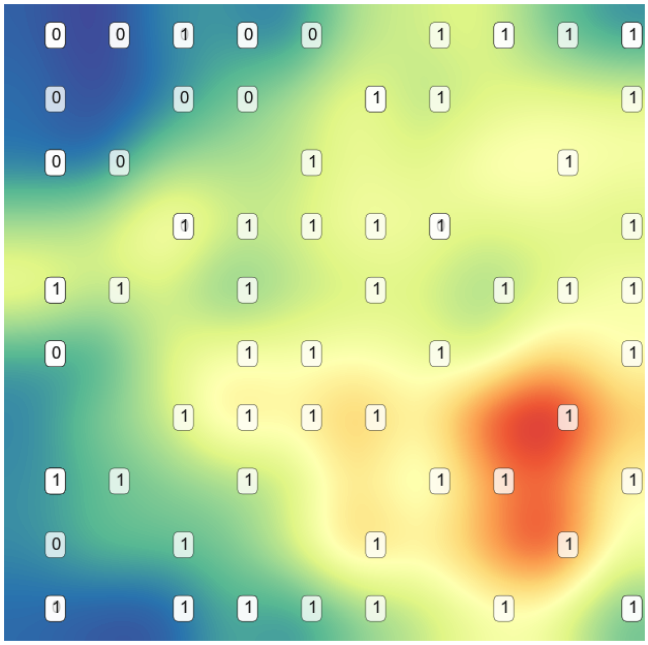

Figure 5.9: GLI-85 reduced features SOM plane

Glioma

The Glioma dataset represents a huge challenge for feature selection, as there are way too many classes (4) for a very small number of samples (50), to make the scenario worse, the classes are all merged with no apparent linear separability.

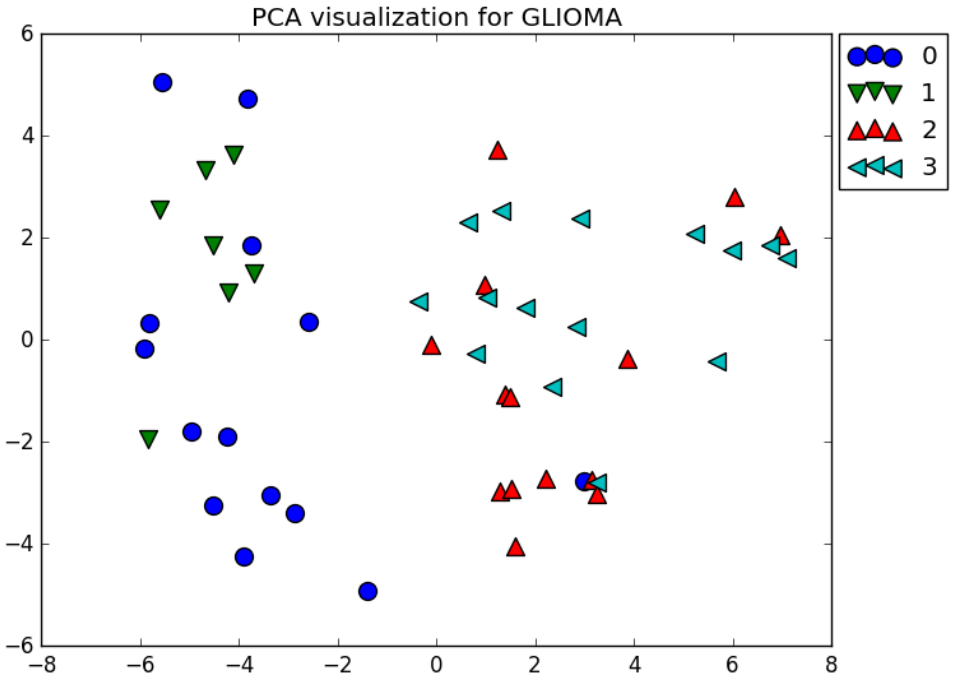

Figure 5.10: Glioma Visualization

According to p-values, Fisher is the more likely to be different than the simple PLR, however it felt short as peak performance regards, running behind Gini and HmbFR with 0.821 and 0.815 however, each algorithm reached that score with a very different set of features, while Gini required 90% of the original space, HmbFR used only 60%. The average score is improved for all algorithms vs PLR.

Table 5.8: Random Forest for Glioma

Num. Features	HmbFR	Fisher	ReliefF	Gini	PLR
4434	0.772	0.772	0.772	0.772	0.772
3991	0.748	0.753	0.785	0.821	0.767
3547	0.747	0.783	0.758	0.787	0.740
3104	0.805	0.764	0.699	0.787	0.751
2660	0.815	0.807	0.759	0.772	0.753
2217	0.755	0.778	0.790	0.773	0.795
1774	0.759	0.770	0.742	0.766	0.760
1330	0.686	0.780	0.775	0.683	0.734
887	0.742	0.717	0.686	0.725	0.675
443	0.777	0.782	0.768	0.735	0.721
Average	0.760	0.771	0.753	0.762	0.747
Peak	0.815	0.807	0.790	0.821	0.795
PLR p-value	0.337	0.028	0.477	0.187	NA

For the Glioma dataset, best reduction threshold seems to be around 70% of original features, however, to help reduce potential noise, we still keep only 10% of the data for build the ESOM planes, in this case, using 443 features which still provides an improvement vs the base model. From the original PCA plot, we can see classes 0 and 1 are overlapped as well as 2 and 3. However, in the ESOM plane, we can a see a more clear separation.

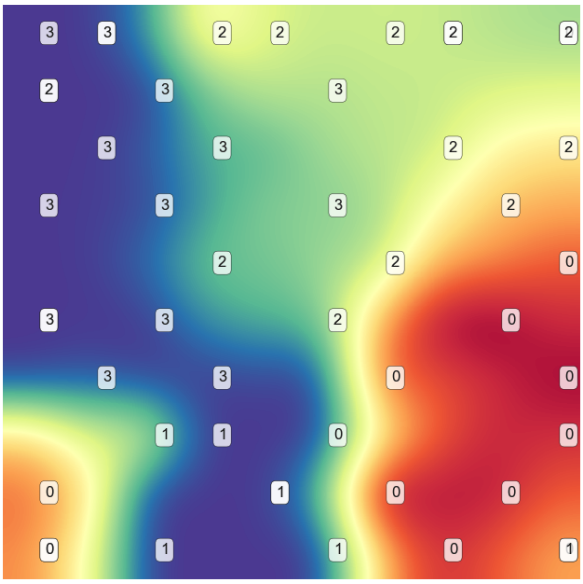

Figure 5.11: Glioma reduced features SOM plane

Leukemia

The Leukemia dataset looks complex for feature selection, with 72 instances and over 7,000 features, the chances of overfitting are huge, however as we can see from the PCA plot, the data does not seem too hard to separate.

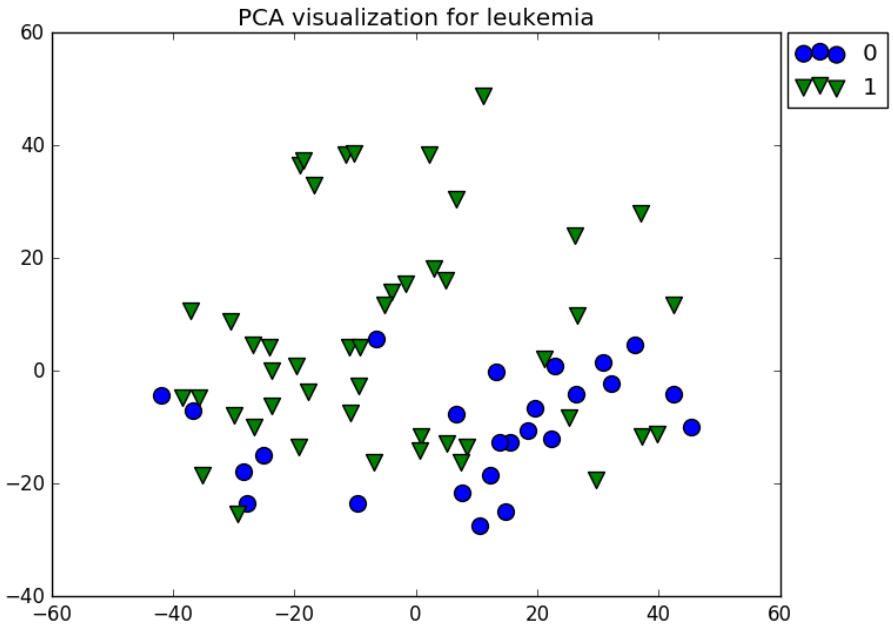

Figure 5.12: Leukemia Visualization

For this particular dataset Random Forest performed the best, with an impressive baseline performance of over 0.93 without any feature selection. It seems that the practical best possible score is 0.981 as all algorithms stuck in that mark, the PLR, in this case, performed way lower than the benchmark algorithms, all of them producing a much higher average score.

Table 5.9: Random Forest for Leukemia

Num. Features	HmbFR	Fisher	ReliefF	Gini	PLR
7070	0.934	0.934	0.934	0.934	0.934
6363	0.949	0.981	0.981	0.934	0.947
5656	0.926	0.952	0.963	0.931	0.918
4949	0.906	0.963	0.981	0.968	0.950
4242	0.981	0.937	0.981	0.968	0.965
3535	0.965	0.981	0.950	0.963	0.952
2828	0.963	0.947	0.968	0.981	0.889
2121	0.981	0.981	0.963	0.981	0.902
1414	0.965	0.981	0.981	0.981	0.885
707	0.963	0.981	0.981	0.981	0.763
Average	0.953	0.964	0.969	0.962	0.911
Peak	0.981	0.981	0.981	0.981	0.965
PLR p-value	0.079	0.036	0.020	0.047	NA

In this dataset the classes are overlapped due to structure complexity, but also imbalance, a 3% increase in model performance is achieved after using only 10% of the original features, this reduction is also beneficial to visualization, as we can see in Figure 5.13, although some overlap still exist, there is a better separation overall.

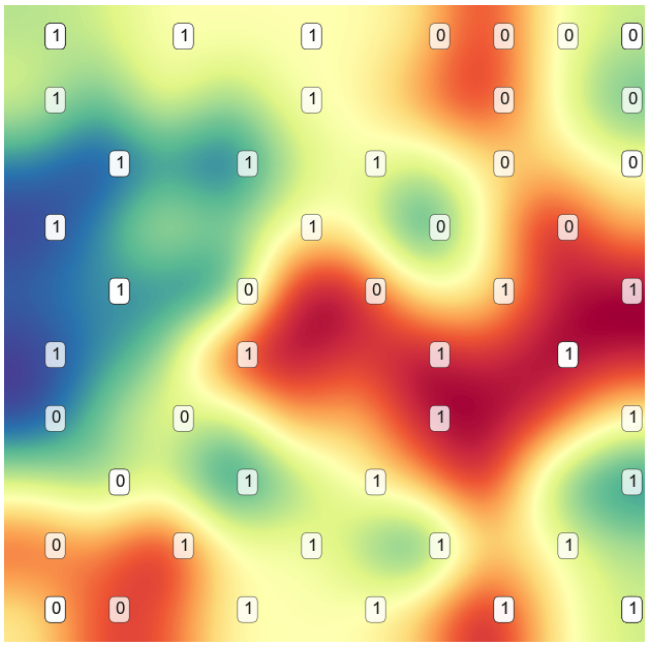

Figure 5.13: Leukemia reduced features SOM plane

Lung

The Lung dataset is a challenge because it has many classes for a relative number of samples and features, in this particular case, more features would have benefited the data in order to improve separability especially for class 0 and 4 as can be seen in the PCA representation.

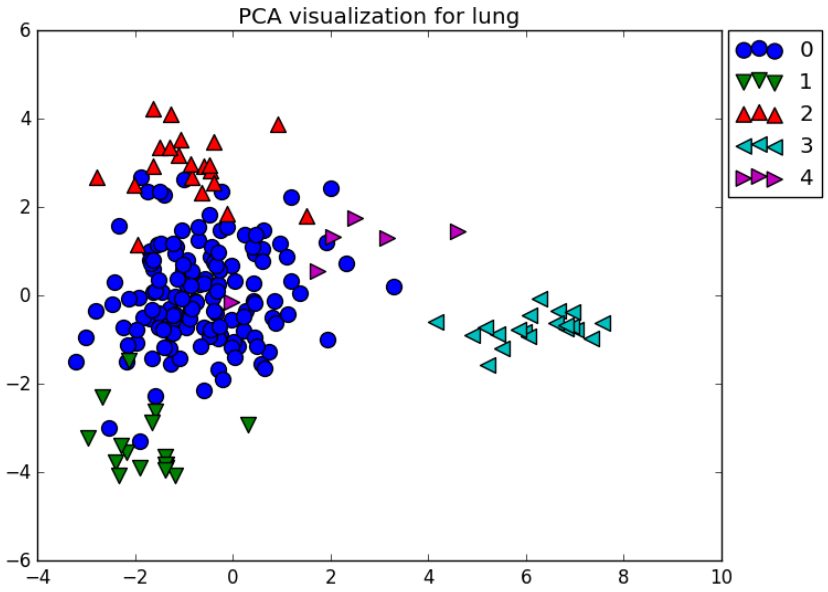

Figure 5.14: Lung Visualization

After analysing results, it seemed that the best practical score could be 0.976 as three algorithms stuck in such mark, however HmbFR managed to find a combination of features capable of 0.98, interesting enough, this combination is also with a smaller set of features of only 10% of the original data, the closest competition comes from Fisher, performing not only slightly less powerful but also requiring 60% of the original data. Averages and p-values are low enough across all methods.

Table 5.10: Logistic Regression for Lung

Num. Features	HmbFR	Fisher	ReliefF	Gini	PLR
3312	0.976	0.976	0.976	0.976	0.976
2981	0.976	0.976	0.976	0.976	0.976
2650	0.976	0.976	0.976	0.976	0.976
2318	0.976	0.976	0.969	0.976	0.965
1987	0.976	0.976	0.969	0.976	0.965
1656	0.976	0.972	0.969	0.976	0.959
1325	0.970	0.972	0.969	0.965	0.964
994	0.976	0.972	0.965	0.965	0.957
662	0.965	0.972	0.965	0.969	0.952
331	0.980	0.959	0.965	0.972	0.921
Average	0.974	0.972	0.970	0.972	0.961
Peak	0.980	0.976	0.976	0.976	0.976
PLR p-value	0.038	0.013	0.063	0.045	NA

In this particular case HmbFS using only 10% of the features achieved the best overall performance, which can be a signal for very high improvements in visualization, starting with the original PCA, we can see class 0 dominating the plot and causing heavy overlap. In the ESOM plane, the class separation is dramatically improved, for instance, class 1 and 0 that were previously overlapped now are fully separated.

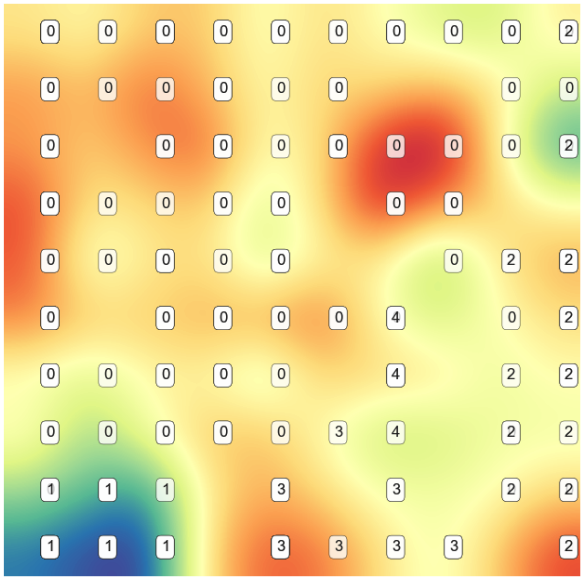

Figure 5.15: Lung reduced features SOM plane

Orl-raws-10p

This dataset, being constructed by face images adds a new scenario for the benchmark, using pixels as features, the ORL face data has over 10,000 features to describe 10 classes (persons). The PCA analysis suggests that there is indeed some linear separability.

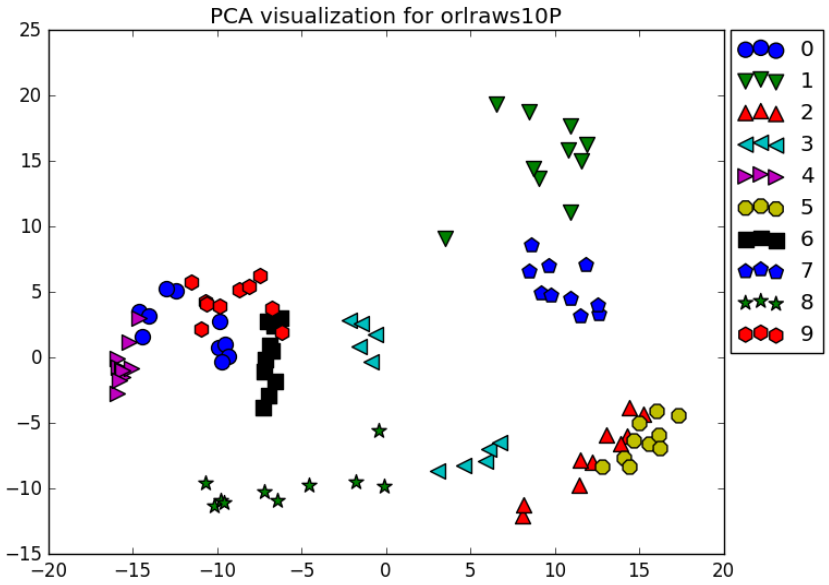

Figure 5.16: ORL Raws 10p Visualization

As expected from the data overview, it can be easily separated even without feature selection for a very powerful 0.967 baseline score. Notable, all algorithms managed to find a subset of features that improved such score, best two were HmbFR and Fisher, improving the recognition by 2% or in some cases a perfect F-Score by Fisher, the p-value, however, is much lower for HmbFR which seems to produce more stable results.

Table 5.11: Logistic Regression for ORL

Num. Features	HmbFR	Fisher	ReliefF	Gini	PLR
10304	0.967	0.967	0.967	0.967	0.967
9274	0.967	0.967	0.967	0.973	0.967
8243	0.967	0.967	0.967	0.973	0.967
7213	0.967	0.967	0.967	0.973	0.967
6182	0.967	0.967	0.973	0.973	0.967
5152	0.973	0.967	0.967	0.973	0.967
4122	0.987	0.967	0.967	0.973	0.973
3091	0.960	0.973	0.973	0.933	0.960
2061	0.973	0.973	0.947	0.887	0.947
1030	0.973	1.000	0.977	0.847	0.877
Average	0.970	0.971	0.967	0.947	0.956
Peak	0.987	1.000	0.977	0.973	0.973
PLR p-value	0.168	0.236	0.286	0.280	NA

Given the very little class overlap for the ORL dataset, improvement in model performance was limited to only 0.6% when using 10% of best features as ranked by HmbFR. The ESOM produced an overall good structure visualization, however, for this specific case, the PCA plot seems to provide a better overview, a signal that feature reduction might not always give an improvement.

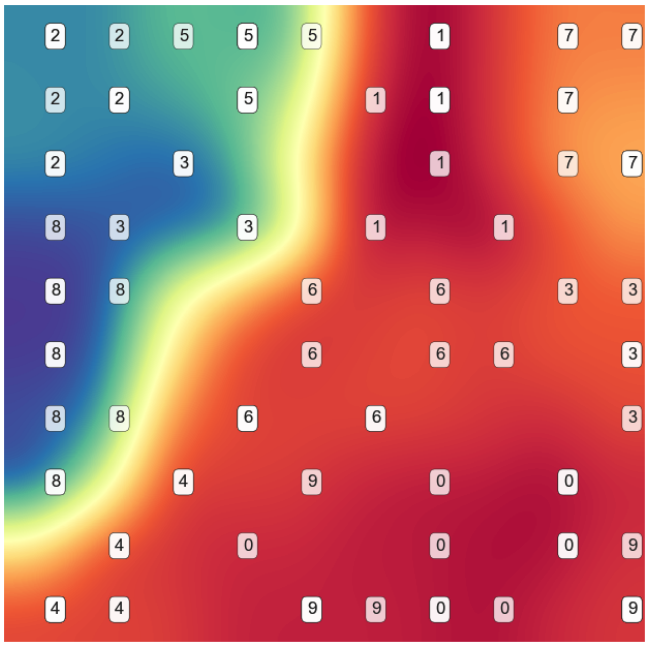

Figure 5.17: ORL reduced features SOM plane

Prostate-ge

The prostate dataset has enough features to separate the binary classes with a very good performance, however, since some samples have heavy overlap, it is possible to reach a point where improvements become very hard.

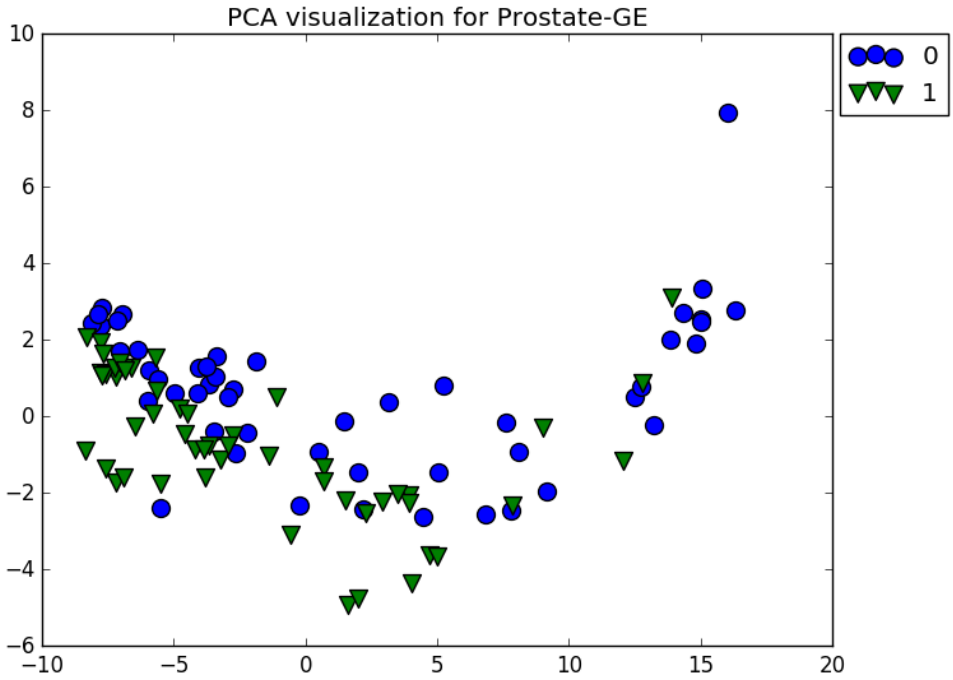

Figure 5.18: Prostate GE Visualization

For this datasets, all algorithms achieved a very low p-value as the baseline PLR performed very poorly doing progression reduction. It is worth noting how from 100% to 60% of features, most algorithms kept the original performance, except ReliefF which produced a good performance improvement. Best two approaches were HmbFR and ReliefF, with 0.939 and 0.95, both of them achieved such results with a very low number of features, 20% and 10% respectively.

Table 5.12: Logistic Regression for Prostate-Ge

Num. Features	HmbFR	Fisher	ReliefF	Gini	PLR
5966	0.911	0.911	0.911	0.911	0.911
5369	0.911	0.911	0.920	0.911	0.911
4773	0.920	0.911	0.920	0.911	0.911
4176	0.910	0.910	0.930	0.911	0.911
3580	0.910	0.910	0.940	0.911	0.911
2983	0.920	0.920	0.940	0.911	0.891
2386	0.920	0.929	0.940	0.920	0.891
1790	0.920	0.929	0.940	0.929	0.891
1193	0.939	0.929	0.950	0.929	0.821
597	0.929	0.929	0.950	0.929	0.826
Average	0.919	0.919	0.934	0.917	0.888
tabst Peak	0.939	0.929	0.950	0.929	0.911
PLR p-value	0.049	0.044	0.011	0.054	NA

The prostate dataset gained around 2% when a 90% of features got removed, based on the original PCA representation, both classes are heavily merged, however in the ESOM plane as can be seen in Figure 5.19, there is a very clear separation of classes with very minimal overlapping.

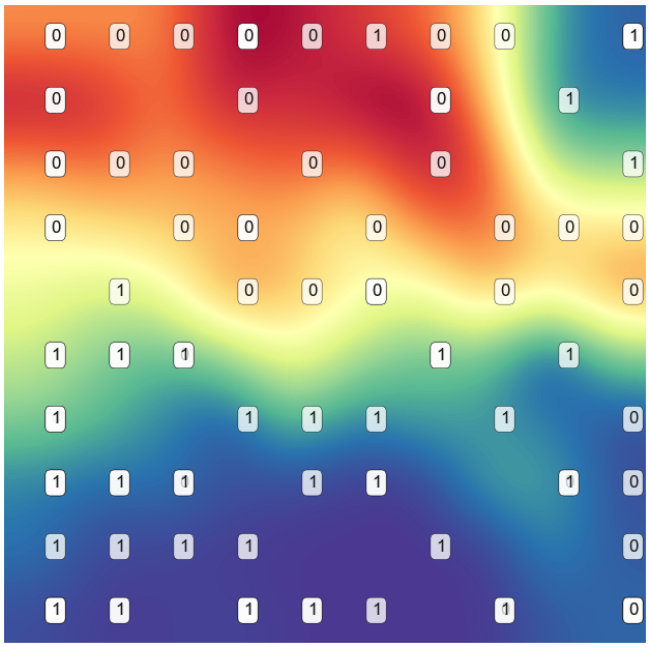

Figure 5.19: Prostate reduced features SOM plane

USPS

Just by looking at the PCA plot, the USPS dataset is clearly a nice add-on to the benchmarks as it is very different, having almost 10,000 samples of hand written digits with a 16x16 resolution (hence 256 features) and 10 different classes, is a nice challenge for feature interactions and reduction. We can see how all classes are sort of merged all over spectrum so linear separability seems very unlikely.

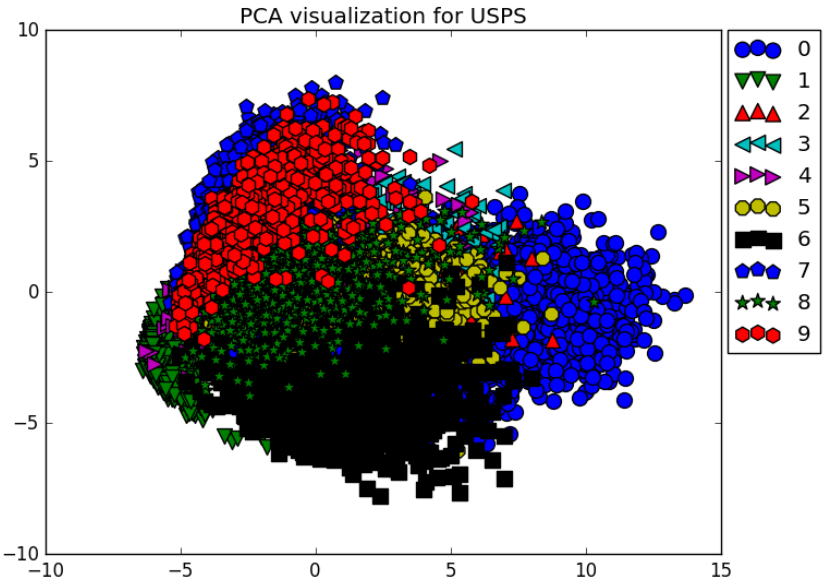

Figure 5.20: USPS Visualization

For this particular dataset, it was possible to achieve 0.945 using Logistic Regression and 0.961 using Random Forest (RF), however, after using feature selection (FS) the results only got worse, although HmbFR achieved as high as 0.851 with RF using only 10% of features. For this analysis, we selected Naive Bayes (NB) as it indeed benefits from FS. From the results table, it is clear how feature reduction is not trivial, PLR achieved scores as low as 0.327 where other techniques such as HmbFR using the same 10% of features achieved 0.685. Gini performed poorly in this case, while the two best were HmbFR and ReliefF, both of them with improvements over 8% and interesting enough, both of them achieved the peak performance with the same number of features at 50%.

Table 5.13: Naive Bayes for USPS

Num. Features	HmbFR	Fisher	ReliefF	Gini	PLR
256	0.783	0.783	0.783	0.783	0.783
230	0.811	0.799	0.811	0.763	0.787
205	0.829	0.820	0.844	0.754	0.795
179	0.859	0.843	0.864	0.709	0.773
154	0.869	0.842	0.874	0.661	0.761
128	0.870	0.835	0.877	0.606	0.703
102	0.856	0.838	0.857	0.620	0.630
77	0.795	0.798	0.836	0.616	0.525
51	0.771	0.766	0.792	0.575	0.389
26	0.685	0.631	0.707	0.494	0.327
Average	0.813	0.795	0.825	0.658	0.647
Peak	0.870	0.843	0.877	0.783	0.795
PLR p-value	0.004	0.007	0.004	0.747	NA

The USPS dataset is our biggest data in term of instances but the smallest in term of features, which means that most instances look very similar, and are all overlapped at least in a 2 dimensions space, after reduction to only 26 features and then building the ESOM plane, the data seems a bit better represented but not enough to overcome the extra processing required.

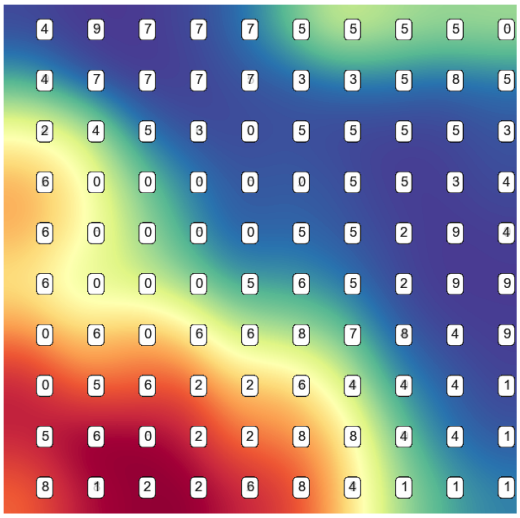

Figure 5.21: USPS reduced features SOM plane

Warp-pie-10p

Looking at the data it's clear that linear separability is challenging, with so many classes and no apparent cluster regions as all classes are mixed together, feature selection algorithms will be required to careful rank the features.

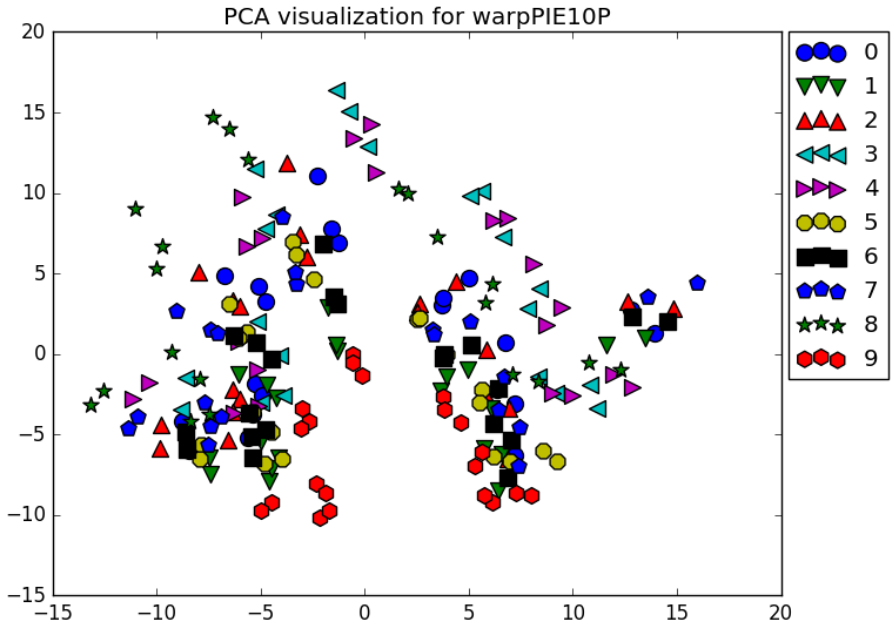

Figure 5.22: Warp Pie 10p Visualization

For this particular dataset, Logistic Regression performed a bit better, however, no useful conclusion could be made as feature selection (FS) did not provide any improvement, however, for Naive Bayes, the improvement is clear and low p-values support this improvement. Best algorithm was HmbFR with a 0.963 score, improving 3% over the baseline without FS using only 20% of the data, another notable result comes from Fisher, with a 0.955 score although using 30% of data.

Table 5.14: Naive Bayes for Warp Pie 10p

Num. Features	HmbFR	Fisher	ReliefF	Gini	PLR
2420	0.934	0.934	0.934	0.934	0.934
2178	0.937	0.940	0.924	0.937	0.910
1936	0.943	0.940	0.922	0.915	0.916
1694	0.946	0.943	0.920	0.891	0.906
1452	0.938	0.947	0.908	0.827	0.854
1210	0.939	0.952	0.916	0.719	0.833
968	0.955	0.951	0.926	0.751	0.792
726	0.956	0.955	0.934	0.761	0.776
484	0.963	0.952	0.948	0.709	0.744
242	0.942	0.946	0.942	0.630	0.755
Average	0.945	0.946	0.927	0.807	0.842
Peak	0.963	0.955	0.948	0.937	0.934
PLR p-value	0.003	0.002	0.008	0.053	NA

In this case, the original 2-dimensional representation is highly overlapped, as a similar scenario as USPS, however in this case, with much more features, noise reduction was actually beneficial, improving 2% with only 10% of the original space. The ESOM plane shows a huge improvement over the original data representation even when it shows some minor overlap.

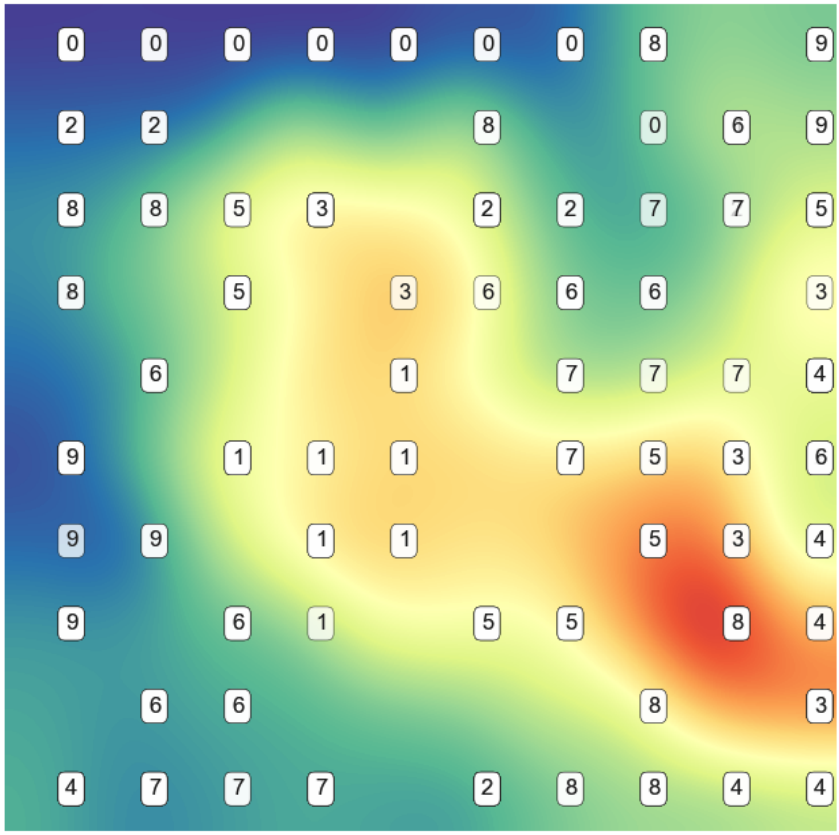

Figure 5.23: Warp reduced features SOM plane

Chapter 6

Conclusions

6.1 Summary

As discussed in thesis statements, the core idea of this thesis is to investigate if its possible to capture important signal from the combination of adjacent features, and therefore, avoiding exhaustive individual analysis for feature selection. In order to support this claim, a series of algorithms have been developed.

The novel idea in this work is the use of a heat map for the task of feature selection. A heatmap is a well known technique in statistics and has been used since decades ago to facilitate the exploration of variables, however, is not trivial how this could be used for feature selection, many challenges arises, such as color space, features interactions and decision equations.

There are many different situations in real-life applications for machine learning, every situation might favor one or other algorithm, to diminish this situation, in this thesis we propose three variants:

Heat Map Based Feature Selection (HmbFS), main original idea, transforms the original space to a virtual heatmap with only 33% of the original variables due to adjacency compression. The statistical analysis of features is performed in the new space and later the results are mapped, this is a very important design element, as many applications require the original space for explainability porpoises. Although it exhibits a good performance, for smaller datasets a more exhaustive search of optimal parameters can be feasible, in this case, the optimal threshold selection, this scenario is considered in the next variant.

Heat Map Based Feature Selection with Cross-validation (HmbFScv), while the original algorithm is a filter approach, this takes the base idea but in a wrapper design, mostly designed for smaller datasets, HmbFScv can automatically tune the selection threshold for best performance. Experiments confirms the improved results producing similar or best performance with a smaller set of features than HmbFS.

There are scenarios where we not only want to know which features are important, but also to find the difference in magnitude of importance, for this case, Heat Map Feature Ranker (HmbFR) address the issue, providing the individual feature importance letting the user decide which features to keep. Worth notice that for this version a two-pass approach needs to be implemented, otherwise all grouped features would have the same score.

Overall across experiments, variants of HmbFS have demonstrated to be very competitive against powerful algorithms while doing only one third of the work due to data compression

6.2 Conclusions

The machine learning (ML) area, in general, deals with projects without exact solution, as if any of such solutions existed, there would be no need at all for ML, given this fact, it means that any solution, for any problem in ML, is always a sub-optimal solution and the search for a better solution is always active.

The case of feature selection is no different, although its possible to argue that using a brute-force approach to find all possible feature combinations, it is imperative to remember that the goal of feature selection is not solely the reduction of variables, but contribute to the overall goal or better model building, which means that even if its possible to find a subset of features that work best for a particular algorithm, it is far from being the global optimal solution, as different parameters or algorithms could have a very different optimal subset.

In this particular Thesis, it has been found that HmbFS and its variants are very competitive and overall contribute to improving model performance in a wide range of applications, however, such results are data dependent, as with any other algorithm in this field. It is possible to create guidelines about which situations may favor one or other algorithm, however, to the best of my knowledge, there is no way to prove an argument about which algorithm is best, until an empirical evaluation is performed.

HmbFS is an algorithm designed for dense, high-dimensional data, it does not mean it would not work on other scenarios but it will clearly be in disadvantage and the use would not be recommended.

6.3 Future Work

The are two main improvements that can be done to this work:

A) Human in the loop (HITL), although this work uses a very visual approach due to the heat map construction, all of such analysis happens in a virtual space, and no actually visual heat map is ever constructed. However, thanks to the powerful human perception, it is possible to include humans in the

selection process by exposing a visual heat map for situations where the algorithm might be in doubt for a pattern selection B) Ensemble of heat maps, this technique (ensembling) has been used with a huge success in other areas of machine learning, the idea is very straightforward, reduce the variance by combining multiple sub-optimal solutions, in this particular problem, it is possible to sub-sample the data and create multiple different heat maps then combine the predictions of all of them to estimate the overall importance of each variable.

Bibliography

[1] N. J. Nilsson, "Introduction to machine learning: An early draft of a proposed textbook. pages 175-188. http://robotics.stanford.edu/people/nilsson/mlbook.ht ml," 1996.

[2] A. J. Smola and S. Vishwanathan, *Introduction to Machine Learning*. Cambridge University Press, 2008. [Online]. Available: http://alex.smola.org/drafts/thebook.pdf,/bib/smola/smola2008ml/thebook.pdf

[3] M. G. Kendall, A. Stuart, and J. K. Ord, Eds., *Kendall's Advanced Theory of Statistics*. New York, NY, USA: Oxford University Press, Inc., 1987.

[4] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Transactions on Information Theory*, vol. 14, no. 1, pp. 55–63, Jan 1968.

[5] A. J. Miller, "Selection of subsets of regression variables," *Journal of the Royal Statistical Society. Series A (General)*, vol. 147, no. 3, pp. 389–425, 1984. [Online]. Available: http://www.jstor.org/stable/2981576

[6] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artif. Intell.*, vol. 97, no. 1-2, pp. 245–271, Dec. 1997. [Online]. Available: http://dx.doi.org/10.1016/S0004-3702(97)00063-5

[7] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, no. 1-2, pp. 273–324, Dec. 1997. [Online]. Available: http://dx.doi.org/10.1016/S0004-3702(97)00043-X

[8] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944968

[9] X. Wang and O. Gotoh, "Accurate molecular classification of cancer using simple rules," *BMC Medical Genomics*, vol. 2, no. 1, pp. 1–23, 2009. [Online]. Available: http://dx.doi.org/10.1186/1755-8794-2-64

[10] S. Xie, "Principal component analysis based feature extraction methods applied to biomedical and communication network data," Ph.D. dissertation, University of Guelph, Ontario, Canada, 2010, aAINR67857.

[11] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, pp. 131–156, 1997.

[12] A. Jain and B. Chandrasekaran, "Dimensionality and sample size considerations," in *Pattern Recognition in Practice*, P. Krishnaiah and L. Kanal, Eds., 1982, pp. 835–855.

[13] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *J. Mach. Learn. Res.*, vol. 3, pp. 1289–1305, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944974

[14] P. Mitra, C. A. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 301–312, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1109/34.990133

[15] Y. Rui and T. S. Huang, "Image retrieval: Current techniques, promising directions and open issues," *Journal of Visual Communication and Image Representation*, vol. 10, pp. 39–62, 1999.

[16] Y. Saeys, I. n. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, Sep. 2007. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btm344

[17] P. Refaeilzadeh, L. Tang, and H. Liu, *On comparison of feature selection algorithms*, 2007, vol. WS-07-05, pp. 34–39.

[18] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957. [Online]. Available: http://books.google.com/books?id=fyVtp3EMxasC&pg=PR5&dq=dynamic+programming+richard+e+bellman&client=firefox-a#v=onepage&q=dynamic%20programming%20richard%20e%20bellman&f=false

[19] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *MACHINE LEARNING: PROCEEDINGS OF THE ELEVENTH INTERNATIONAL*. Morgan Kaufmann, 1994, pp. 121–129.

[20] A. Navot, R. Gilad-Bachrach, Y. Navot, and N. Tishby, "Is feature selection still necessary?" in *SLSFS*, ser. Lecture Notes in Computer Science, C. Saunders, M. Grobelnik, S. R. Gunn,

and J. Shawe-Taylor, Eds., vol. 3940. Springer, 2005, pp. 127–138. [Online]. Available: http://dblp.uni-trier.de/db/conf/slsfs/slsfs2005.html#NavotGNT05

[21] S. Alelyani, "On feature selection stability: A data perspective," Ph.D. dissertation, Tempe, AZ, USA, 2013, aAI3558275.

[22] Q. Gu, Z. Li, and J. Han, "Generalized fisher score for feature selection," *CoRR*, vol. abs/1202.3725, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1202.html#abs-1202-3725

[23] J. Zhou, J. Liu, V. A. Narayan, and J. Ye, "Modeling disease progression via fused sparse group lasso," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 1095–1103. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339702

[24] S. Das, "Filters, wrappers and a boosting-based hybrid for feature selection," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 74–81. [Online]. Available: http://dl.acm.org/citation.cfm?id=645530.658297

[25] J. N. Weinstein, "Biochemistry. a postgenomic visual icon." *Science*, vol. 28, Mar. 2008.

[26] T. Loua, *Atlas statistique de la population de Paris*, 1873.

[27] J. Bertin, *Semiologie Graphique, Paris: Editions Gauthier Villars.*, 1967.

[28] J. Yu and X.-W. Chen, "Bayesian neural network approaches to ovarian cancer identification from high-resolution mass spectrometry data," *Bioinformatics*, vol. 21, no. 1, pp. 487–494, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/bti1030

[29] S. Datta and L. M. DePadilla, "Feature selection and machine learning with mass spectrometry data for distinguishing cancer and non-cancer samples," *Statistical Methodology*, vol. 3, no. 1, pp. 79 – 92, 2006, bioinformatics. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S157231270500064X

[30] C. E. Bonferroni, "Il calcolo delle assicurazioni su gruppi di teste," *In Studi in Onore del Professore Salvatore Ortu Carboni*, pp. 13 – 60, 1935.

[31] S. S. Y. Peter H. Westfall, *Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment*. John Wiley & Sons, 1993.

[32] Y. Liu, "Feature extraction and dimensionality reduction for mass spectrometry data," *Comput. Biol. Med.*, vol. 39, no. 9, pp. 818–823, Sep 2009.

[33] J. Ke, L. zhu, B. Han, Q. Dai, Y. Wang, L. Li, S. Xu, H. Mou, and Z. Zheng, "Sparse representation based feature selection for mass spectrometry data," in *Bioinformatics and Biomedicine Workshops (BIBMW), 2010 IEEE International Conference on*, Dec 2010, pp. 57–62.

[34] B. Wu, T. Abbott, D. Fishman, W. McMurray, G. Mor, K. L. Stone, D. Ward, K. R. Williams, and H. Zhao, "Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data." *Bioinformatics*, vol. 19, no. 13, pp. 1636–1643, 2003. [Online]. Available: http://dblp.uni-trier.de/db/journals/bioinformatics/bioinformatics19.html#WuAFMMSWWZ03

[35] T. Abeel, T. Helleputte, Y. Van de Peer, P. Dupont, and Y. Saeys, "Robust biomarker identification for cancer diagnosis with ensemble feature selection methods," *Bioinformatics*, vol. 26, no. 3, pp. 392–398, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btp630

[36] H. Kim, J. Watkinson, and D. Anastassiou, "Biomarker discovery using statistically significant gene sets." *Journal of Computational Biology*, vol. 18, no. 10, pp. 1329–1338, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/jcb/jcb18.html#KimWA11

[37] S. Ahmed, M. Zhang, and L. Peng, "Feature selection and classification of high dimensional mass spectrometry data: A genetic programming approach," in *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, ser. Lecture Notes in Computer Science, L. Vanneschi, W. Bush, and M. Giacobini, Eds. Springer Berlin Heidelberg, 2013, vol. 7833, pp. 43–55.

[38] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, Mar. 2002. [Online]. Available: http://doi.acm.org/10.1145/505282.505283

[39] Y. Sun and D. Wu, "A relief based feature extraction algorithm." in *SDM*. SIAM, 2008, pp. 188–195.

[40] F. Gonzalez and L. A. B. Munoz, "Feature selection for microarray gene expression data using simulated annealing guided by the multivariate joint entropy," *CoRR*, vol. abs/1302.1733, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1302.html#abs-1302-1733

[41] Z. Zhao, F. Morstatter, S. Sharma, A. Anand, and H. Liu, "Advancing feature selection research asu feature selection repository," 2010.

[42] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical machine learning tools and techniques with java implementations," 1999.

[43] M. A. Hall, "Correlation-based feature selection for machine learning," Tech. Rep., 1998.

[44] S. M. Stigler, "Francis galton's account of the invention of correlation," *Statist. Sci.*, vol. 4, no. 2, pp. 73–79, 05 1989. [Online]. Available: https://doi.org/10.1214/ss/1177012580

[45] R. Kerber, "Chimerge: Discretization of numeric attributes," in *Proceedings of the Tenth National Conference on Artificial Intelligence*, ser. AAAI'92. AAAI Press, 1992, pp. 123–128. [Online]. Available: http://dl.acm.org/citation.cfm?id=1867135.1867154

[46] H. Liu and R. Setiono, "Chi2: Feature selection and discretization of numeric attributes," in *In Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, 1995, pp. 388–391.

[47] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, T. Fawcett and N. Mishra, Eds., 2003, pp. 856–863. [Online]. Available: http://www.aaai.org/Papers/ICML/2003/ICML03-111.pdf

[48] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.

[49] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, P. B. Schölkopf, and J. C. Platt, Eds. MIT Press, 2006, pp. 507–514. [Online]. Available: http://papers.nips.cc/paper/2909-laplacian-score-for-feature-selection.pdf

[50] L. Ceriani and P. Verme, "The origins of the gini index: extracts from variabilità e mutabilità (1912) by corrado gini," *The Journal of Economic Inequality*, vol. 10, no. 3, pp. 421–443, Sep 2012. [Online]. Available: https://doi.org/10.1007/s10888-011-9188-x

[51] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[52] L. J. Wei, "Asymptotic conservativeness and efficiency of kruskal-wallis test for k dependent samples," *Journal of the American Statistical Association*, vol. 76, no. 376, pp. 1006–1009, 1981. [Online]. Available: http://dx.doi.org/10.1080/01621459.1981.10477756

[53] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, Jan. 2000. [Online]. Available: http://dx.doi.org/10.1109/34.824819

[54] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of the Ninth International Workshop on Machine Learning*, ser. ML92. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992, pp. 249–256. [Online]. Available: http://dl.acm.org/citation.cfm?id=141975.142034

[55] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*. John Wiley and Sons, 2003.

[56] C. B. D. Newman and C. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html

[57] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *J. Mach. Learn. Res.*, vol. 5, pp. 1205–1224, Dec. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1005332.1044700

[58] D. Koller and M. Sahami, "Toward optimal feature selection," in *In 13th International Conference on Machine Learning*, 1995, pp. 284–292.

[59] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *MACHINE LEARNING: PROCEEDINGS OF THE ELEVENTH INTERNATIONAL*. Morgan Kaufmann, 1994, pp. 121–129.

[60] X. Zhang, X. Lu, Q. Shi, X.-q. Xu, H.-c. Leung, L. Harris, J. Iglehart, A. Miron, J. Liu, and W. Wong, "Recursive svm feature selection and sample classification for mass-spectrometry and microarray data," *BMC Bioinformatics*, vol. 7, no. 1, p. 197, 2006. [Online]. Available: http://www.biomedcentral.com/1471-2105/7/197

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

[62] U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by

oligonucleotide arrays," *Proceedings of the National Academy of Sciences*, vol. 96, no. 12, pp. 6745–6750, Jun. 1999.

[63] D. Chowdary, J. Lathrop, J. Skelton, K. Curtin, T. Briggs, Y. Zhang, J. Yu, Y. Wang, and A. Mazumder, "Prognostic Gene Expression Signatures Can Be Measured in Tissues Collected in RNAlater Preservative," *The Journal of Molecular Diagnostics*, vol. 8, no. 1, pp. 31–39, Feb. 2006.

[64] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring." *Science*, vol. 286, no. 5439, pp. 531–537, Oct. 1999.

[65] M. A. Shipp, K. N. Ross, P. Tamayo, A. P. Weng, J. L. Kutok, R. C. T. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. S. Pinkus, T. S. Ray, M. A. Koval, K. W. Last, A. Norton, T. A. Lister, J. Mesirov, D. S. Neuberg, E. S. Lander, J. C. Aster, and T. R. Golub, "Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning." *Nature Medicine*, vol. 8, no. 1, pp. 68–74, Jan. 2002.

[66] D. Singh, P. G. Febbo, K. Ross, D. G. Jackson, J. Manola, C. Ladd, P. Tamayo, A. A. Renshaw, A. V. D'Amico, J. P. Richie, E. S. Lander, M. Loda, P. W. Kantoff, T. R. Golub, and W. R. Sellers, "Gene expression correlates of clinical prostate cancer behavior," *Cancer Cell*, vol. 1, no. 2, pp. 203–209, Mar. 2002.

[67] A. I. Su, M. P. Cooke, K. A. Ching, Y. Hakak, J. R. Walker, T. Wiltshire, A. P. Orth, R. G. Vega, L. M. Sapinoso, A. Moqrich, A. Patapoutian, G. M. Hampton, P. G. Schultz, and J. B. Hogenesch, "Large-scale analysis of the human and mouse transcriptomes." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 7, pp. 4465–4470, Apr. 2002.

[68] E. Tian, F. Zhan, R. Walker, E. Rasmussen, Y. Ma, B. Barlogie, and J. D. Shaughnessy, Jr., "The Role of the Wnt-Signaling Antagonist DKK1 in the Development of Osteolytic Lesions in Multiple Myeloma," *New England Journal of Medicine*, vol. 349, no. 26, pp. 2483–2494, Dec. 2003.

[69] H. Liu and R. Setiono, "Chi2: Feature selection and discretization of numeric attributes," in *In Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, 1995, pp. 388–391.

[70] Y. Benjamini and Y. Hochberg, "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995. [Online]. Available: http://dx.doi.org/10.2307/2346101

[71] R. Lomax, *Statistical Concepts: A Second Course*. Lawrence Erlbaum Associates, 2007. [Online]. Available: https://books.google.com.mx/books?id=p17rT373FNAC

[72] M. Suganthy and P. Ramamoorthy, "Principal component analysis based feature extraction, morphological edge detection and localization for fast iris recognition."

[73] V. A. Huynh-Thu, L. Wehenkel, and P. Geurts, "Exploiting tree-based variable importances to selectively identify relevant variables." in *FSDM*, ser. JMLR Proceedings, Y. Saeys, H. Liu, I. Inza, L. Wehenkel, and Y. V. de Peer, Eds., vol. 4. JMLR.org, 2008, pp. 60–73.

[74] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.869

[75] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theor.*, vol. 13, no. 1, pp. 21–27, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1967.1053964

[76] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1390681.1442794

[77] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Dec. 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1248547.1248566

[78] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[79] K. Grabczewski and N. Jankowski, "Feature selection with decision tree criterion." in *HIS*, N. Nedjah, L. de Macedo Mourelle, A. Abraham, and M. Kppen, Eds. IEEE Computer Society, 2005, pp. 212–217.

[80] Gravier, Eleonore, G. Pierron, A. Vincent-Salomon, N. gruel, V. Raynal, A. Savignoni, Y. De Rycke, J.-Y. Pierga, C. Lucchesi, F. Reyal, A. Fourquet, S. Roman-Roman, F. Radvanyi, X. Sastre-Garau, B. Asselain, and O. Delattre, "A prognostic DNA signature for T1T2

node-negative breast cancer patients." *Genes, Chromosomes and Cancer*, vol. 49, no. 12, pp. 1125–1125, Sep. 2010.

[81] Z. Gu, R. Eils, and M. Schlesner, "Complex heatmaps reveal patterns and correlations in multidimensional genomic data," *Bioinformatics*, vol. 32, no. 18, pp. 2847–2849, 2016. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btw313

[82] A. Pujol and L. Chen, "Color quantization for image processing using self information," in *2007 6th International Conference on Information, Communications Signal Processing*, Dec 2007, pp. 1–5.

[83] M. Tan, I. W. Tsang, and L. Wang, "Towards ultrahigh dimensional feature selection for big data," *Journal of Machine Learning Research*, vol. 15, pp. 1371–1429, 2014. [Online]. Available: http://jmlr.org/papers/v15/tan14a.html

[84] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 413–422. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2008.17

[85] P. Wittek, "Somoclu: An efficient distributed library for self-organizing maps," *CoRR*, vol. abs/1305.1422, 2013. [Online]. Available: http://arxiv.org/abs/1305.1422