A Study on the Impact of Feature Selection on Data Analysis

Njoku, Uchechukwu Fortune

Award date:
2021

Awarding institution:
Universitat Politècnica de Catalunya

A Study on the Impact of Feature Selection on Data Analysis

Master Thesis Report

Uchechukwu Fortune, NJOKU

Supervisors:
Dirk FAHLAND (d.fahland@tue.nl)
Alberto ABELLO (aabello@essi.upc.edu)
Besim BILALLI (bbilalli@essi.upc.edu)

15th-August-2021

Contents

List of Figures

List of Tables

Acknowledgements

This master's thesis would not have been possible without the support and unreserved love of many people, to whom I am sincerely grateful.

I would like to express my profound gratitude to my supervisors Dirk FAHLAND, Alberto ABELLO, and Besim BILALLI, for their excellent guidance and support during this work especially for their availability and sacrifice during the summer vacation to give prompt and detailed feedback in the thesis report writing phase.

I would like to thank the BDMA coordinators and all my professors for this incredible opportunity that has indeed changed my life, to say the least. I have had the opportunity to learn at the best universities in Europe; I am grateful. Special thanks to Charlotte; truly not all superheroes wear capes. I acknowledge her excellent administration and care; especially during the pandemic.

Special thanks to my friends, those I have known my whole life and to the amazing new friends I made during my BDMA studies. I sincerely appreciate every time spent together, your kind words, and warm wishes.

Finally, to my family, I am not sure what I would do without them. I have experienced unconditional love and unreserved support during my program. Concluding the last phase of my thesis with you was indeed the best thing. Thanks for taking care of all I needed while I focused on completing my thesis.

Abstract

The high-dimensionality of Big Data poses the problems such as storage, difficulty in understanding and visualizing the data, poor generalization by models, and lengthy model building times in data analysis. This gives rise to the need for feature selection to improve the analysis of data. An understanding of the factors that influence the performance of feature selection methods as well as their impact on various data analysis tasks is key in guiding the choice of a feature selection method. This study aims to review the factors that influence the performance of various feature selection methods as well as how they impact classification in terms of improved training time and accuracy; to propose guidelines for choosing feature selection methods. Eight feature selection methods were analyzed on four classification algorithms using 32 real-world datasets and the outcome was used to propose guidelines for choosing feature selection methods. The results showed that feature selection impacts binary and multiclass classification differently and although no one of the studied feature selection methods is best in all cases, we observed that correlation based and conditional mutual information maximization feature selection methods gave the best results for multiclass classification. The Python scikit-learn libraries and scikit-feature repository which have implemented many learning and feature selection algorithms respectively were used for the experiments.

Chapter 1

Introduction

1.1 Motivation

Data has been termed the new oil, gold, and even currency of the current economy; this tells its value. Data comes in varying sizes; from a few kilobytes stored in the SRAM of an embedded system generated during program execution to hundreds of gigabytes, terabytes, or even petabytes generated from intensive data-producing processes, for example, data warehousing. Using data, many advances have been made in science, technology, and other fields allowing for more informed decision making and automation. Models that power exciting innovations such as self-driving cars and facial recognition systems are made possible because of data. The potential that data holds has led to increased collection whether or not its use is clear at the time of collection.

With cheaper storage available, many organizations store up data without proper problem definition hoping to gain some insight from the gathered data in the future. This 'blind' data collection often leaves practitioners with large-sized noisy data having features that could be irrelevant or redundant. Irrelevant, meaning that they carry no information about a target feature alone or in combination with other variable(s). Redundant, meaning that there exists some other feature(s) that conveys the same information about the target feature [2].

In certain fields like bioinformatics, in addition to data having a large number of features, it often has an insufficient number of samples. Microarray data for example could contain thousands of gene expressions as features but a few tens or hundreds of samples. With this kind of data, it is almost impossible to build a model of sufficiently good quality that generalizes the process which generated the data. The intuitive reasoning with such cases is

to gather more samples. However, in such cases, data collection can be time consuming, expensive, or even impossible. Hence, the practitioner is faced with datasets obtained from insufficient number of samples, having many features that may be irrelevant or redundant and has to identify and filter out such features.

1.2 Problem Statement

Although Big Data has played a significant role in building efficient models for better decision making, it comes with the problems of storage, challenging data understanding and visualization, poor generalization by models due to overfitting, and lengthy model building times. As the dimensionality of a dataset increases, the required number of samples needed to build a model that generalizes it properly grows more than exponentially [2]. The state of having insufficient samples compared to the dimensionality of a dataset is referred to as sparseness and it generally increases as the dimensionality increases. Sparseness negatively affects the quality of a model's performance metrics, for example, its accuracy.

Besides the problem of sparseness, high-dimensional datasets are difficult to understand and visualize. Models built with such datasets are thus difficult to explain, especially in terms of the many features; the time required to build such models could also be very long. Associated with high dimensional data is also the need for relatively more storage space which signifies increased expenses whether on-site or in the cloud. These resultant challenges from the high dimensionality of datasets have been termed the curse of dimensionality. To fight the curse of dimensionality, effective techniques to reduce the dimensionality of datasets by selecting the most informative features of a dataset are necessary.

Many of such techniques have been long proposed and are called Feature Selection (FS) methods. Most FS methods were designed for small-medium sized datasets with polynomial theoretical runtime [3] and so in the face of large-sized datasets, their effectiveness becomes jeopardized and they fall into the curse of dimensionality [4, 2]. Individual FS methods are affected by various characteristics in the dataset like its dimensionality or the number of samples. It is therefore imperative to understand the performance of FS methods with respect to the changing characteristics of datasets; in order to serve as a guide for choosing FS methods.

Depending on the data analysis task at hand, FS methods yield differing impacts in terms of improving the building time or accuracy of a model from a dataset. Therefore, the impact of FS methods differs for classification (bi-

nary, multiclass), clustering and regression tasks. Besides impacting various tasks differently, its effect could also differ per learning algorithm, hence, a context-specific understanding of the impact of FS is necessary.

Research Questions

Focusing on binary and multiclass classification tasks, the impact of eight FS methods on classification problems is studied in this work to answer the following questions:

1. What factors influence the runtime of each FS method?
2. How well do the FS methods scale with increasing numbers of features/instances?
3. How does FS affect the accuracy of binary/multiclass classification models?
4. What is the impact of the feature subset size on classifier accuracy-runtime tradeoff?
5. What are the guidelines for choosing FS methods?

Chapter 2

Dimensionality Reduction

Datasets seldom come in a form fit for analysis, they are often dirty with inaccurate, incomplete, or(and) inconsistent data having features that are often irrelevant or(and) redundant. To prepare the data for analysis, the dataset has to pass through a number of data-preprocessing steps that broadly consists of data cleaning, data transformation, and data reduction.

Where the dataset has numerous features, data reduction is a necessary pre-processing step. Dimensionality Reduction (DR) is an umbrella term for techniques used in data reduction; the aim is to decrease the number of features of a dataset. DR consists of Feature Extraction (FE) and Feature Selection (FS) and these are discussed below.

2.1 Feature Extraction

FE is the process of reducing the dimensionality of a dataset by projecting its features into a lower-dimensional space that is a combination (linear or nonlinear) of the original features. FE produces fewer new features from the original features still carrying the same information. Singular Value Decomposition (SVD), Latent Semantic Indexing (LSI), and Principal Component Analysis (PCA) are examples of feature extraction techniques

2.2 Feature Selection

FS reduces the dimensionality of a dataset by selecting a subset of datasets' features that are considered to be most relevant based on defined criteria. Depending on the type of FS method, FS involves only the first two or all four of the following steps [5]:

- A generation procedure that guides the selection of a subset of the original features,
- An evaluation procedure in which an evaluation function is used to quantify the relevance of the features,
- A stopping procedure in the case of iterative FS methods using some predefined halt criterion, and
- A validation procedure that evaluates the validity of the selected feature subset.

FS methods can be categorized into *supervision*, *selection strategy*, and *evaluation criteria* perspectives [6] elaborated below.

2.2.1 Feature Selection Categorizations

Supervision perspective

Datasets can either be labeled or unlabelled and FS can be applied on either types of datasets. Based on the reliance of FS methods on the label in a dataset, they are classified into *supervised* and *unsupervised* FS methods.

Supervised FS methods rely on the label of the dataset to evaluate the relevance of features in the dataset. The goal of these methods is to select features that rightly discriminate the instances between distinct label values. Supervised FS methods are mainly applied to supervised problems such as classification and regression

Unsupervised FS methods on the contrary, do not rely on the label of the data to evaluate and select features from a dataset. Rather, alternative methods like distance measures between instances are used to select features aiming to select features that preserve the intrinsic structure of the dataset. Although these alternative methods can be applied to both supervised and unsupervised problems, unsupervised FS methods are originally designed for unsupervised tasks like clustering.
A large number of supervised methods have been proposed than unsupervised methods for FS.

Selection strategy perspective

Irrespective of their supervision perspective, FS methods can be classified by their reliance on a learning algorithm to select the most relevant features into *filter*, *wrapper*, and *embedded* FS methods.

Filter methods evaluate and select features independent of any learning algorithm; solely relying on the characteristics of the dataset to determine the relevance of a feature. This approach gives the advantage of selecting a subset of features once and using it with multiple learning algorithms. Albeit, since these methods do not select features for any particular learning algorithm, the selected features may not be optimal for any chosen learning algorithm.

Wrapper methods select features by going through an iterative process of selecting a subset of features and evaluating its quality by the performance metric (e.g., accuracy) of a particular learning algorithm. This is done until arriving at a subset of features that optimize the performance of that learning algorithm or till a pre-set halt condition is satisfied. Since wrapper methods select features using a learning algorithm, it leads to selecting a subset of features that optimize the performance of the learning algorithm. However, wrapper methods can be time inefficient as the size of the dataset increases since each iteration requires training the learning algorithm.

Embedded methods refer to learning algorithms that have FS in them. Embedded methods have the advantages of time efficiency since FS is not done iteratively; and optimal subset selection since FS is done with the target learning algorithm. Examples of embedded methods are tree-based learning algorithms which prune irrelevant features while building the tree model, using criteria like *Gini* or *Entropy*.

Evaluation criteria perspective

In defining a criterion to quantify the relevance of features, FS methods generally use *sparse learning*, *similarity*, *statistics measures*, or *information theory*; in forming another categorization of FS methods.

Sparse learning-based methods measure the relevance of a feature by its ability to minimize the fitting errors along with some sparse regularization terms like the l1 (LASSO) or l2 (Ridge) norm. Studies show that these methods have good performance and increase model interpretability [6]. This is because they are typically embedded in the learning algorithm which leads to selecting a subset of features that is optimal for the particular learning algorithm with a model that can be explained due to the sparsity of feature weights these methods produce. Although results from these methods are optimal for the particular learning algorithm, they are not generalizable as they might be inefficient when used in another algorithm. Methods like these often involve expensive matrix operations such as inverse and multiplication and can thus be computationally inefficient. Examples of these methods include lasso and ridge regressions and support vector machines.

Similarity-based methods are the group of FS methods that define the relevance of a feature based on its ability to preserve data similarity. Data similarity is derived from the label of the dataset or by some distance measure, for supervised and unsupervised FS methods respectively. This implies that there are similarity-based FS methods for both supervised and unsupervised problems.

These methods are all filter methods (i.e independent of target learning algorithm), hence, the selected subset of features can be used in many learning algorithms. However, similarity-based FS methods do not handle feature redundancy because features are evaluated and selected individually leading to high ranking redundant features being selected.

Statistical-based methods are a group of FS methods that use well-known statistical measures like variance, chi-square, and correlation to quantify the relevance of a feature. These methods often evaluate features individually, not relying on any particular learning algorithm. The selected subset of features might thus contain redundant features but can be generalized (i.e can be used with multiple learning algorithms).

This category of FS methods is often computationally inexpensive but works with only discrete features; continuous features would need to be discretized before being used in these methods.

Information theoretical-based methods consist of FS methods that measure the relevance of a feature using various heuristic filter criteria like entropy or mutual information. Most proposed criteria in this family are designed to select features that maximize relevance and minimize redundancy; this yields a subset of less redundant features. These methods rely on the label of the dataset to evaluate and select features, making them only suitable for supervised tasks. In addition, due to their method filtering attribute, the resulting feature subset can be applied to any supervised learning algorithm. Finally, these methods work with discrete features and thus require continuous features to first be discretized before applying them.

Chapter 3

Background

3.1 Literature Review

The subject of FS has been studied by researchers, especially in recent decades, due to its relevance in Big Data. The objective of FS is to build simpler and more understandable models, improve model performance, reduce their training time, and prepare data that are clean, understandable, and easier to visualize [6]. FS has been used in diverse fields including text mining, image processing, computer vision, industrial applications, bioinformatics, etc [7]; as a tool to fight the problems that come with the size and volume of Big Data.

However, these challenges of Big Data have caught up with some FS methods, affecting their effectiveness and posing new challenges for FS [8]. The challenges identified in the literature are scalability, data formats, distribution, and stability [8, 9]. Due to FS methods being mainly designed for centralized data that are small to medium in size, it is difficult to apply them to Big Data that are often distributed for better management, thereby compromising their scalability with increasing data size. Lastly, a lack of robustness in the results from FS methods with changes in training data (i.e. poor stability) dents the reliability of the results. In [10], the scalability of 3 filter methods on 2 datasets was studied and the results confirmed the inadequacy of these methods to scale with increasing sizes of data. Although progress has been made in redesigning some FS methods in a distributed manner to match the current convention of data management and improve scalability [11, 12],this has not been done on many more methods.

The choice of FS methods to use depends on the search technique, evaluation criteria, and data mining task [3]. In addition to these, the characteristics of the dataset also influence this choice. The study of the effects

of dataset characteristics; classified into standard measures, data sparsity measures, statistical measures, information theoretic measures, and noise measures [13], on FS in [14] showed the inherent relationship between the FS method performance (in terms of improved learning algorithm runtime and accuracy) and the characteristics of the dataset. However, the quality of features selected by FS methods may not be optimal as many of these methods are designed under the assumption that features are not correlated which often is not the case [15].

Hundreds of FS methods have been proposed and several benchmarks to compare their benefits to other analysis, tasks and algorithms have been conducted. In [16], 22 filter methods were benchmarked on 16 real datasets with respect to run time and accuracy when combined with three classification methods. Of these filter methods, three groups were identified to return similar feature rankings, however, no group outperformed the rest on all datasets. Based on the review in [7], filter based FS methods with information theory evaluation criteria and wrapper based methods using greedy stepwise selection approaches appear to give the best results. Most of the reviews and benchmarks were conducted on real data from open repositories like OpenML[1] and UCI[2]. In [17] however, 11 FS methods using 11 synthetic datasets allowing for more control on the characteristics of the dataset were benchmarked and the results obtained were confirmed on two real datasets. From the results, ReliefF turned out to be the best choice as it selected better subsets independent of the characteristics of the dataset.

There is a need for more research in FS to fight the challenges, understand the limits of current methods and propose guidelines around their usage. The focus of previous studies has been on binary classification due to the availability of data and ease of evaluation. However, many real world problems are indeed multiclass in nature and also require FS. Hence in this work, the impact of eight FS methods is studied not just on binary classification but also on multiclass classification to understand the factors that influence their performance, their impact on classification and propose a guideline for using them.

3.2 Preliminaries

In what follows, some fundamental information and theoretical measures important to understand the FS methods studied in this work are defined. **Entropy** quantifies the uncertainty of a discrete random variable X and is

[1] https://www.openml.org

[2] https://archive.ics.uci.edu/ml/index.php

defined as follows:

$$H(X) = \sum_{x_i \epsilon X} P(x_i) \log(P(x_i)), \tag{3.1}$$

where x_i is the i^{th} value of X and $P(x_i)$ is the probability of x_i over all values of X.

Conditional entropy measures the remaining uncertainty of X given another discrete variable Y and defined as:

$$H(X|Y) = \sum_{y_i \epsilon Y} P(y_j) \sum_{x_i \epsilon X} P(x_i|y_j) \log(P(x_i|y_j)), \tag{3.2}$$

where y_i is the i^{th} value of Y, $P(y_j)$ is the prior probability of y_j over all values of Y and $P(x_i|y_j)$ is the conditional probability of x_i given y_j.

Mutual information is the measure of information shared between two discrete random variables X, Y and is defined as:

$$I(X;Y) = H(X) - H(X|Y). \tag{3.3}$$

Conditional mutual information quantifies the amount of information shared between two discrete random variables X, Y when a third discrete random variable Z is known. It is defined as:

$$I(X;Y|Z) = H(X|Z) - H(X|Z,Y). \tag{3.4}$$

Bayes' theorem states that given two events A and B, with $P(B) \neq 0$, then:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{3.5}$$

where $P(A)$ and $P(B)$ are the prior probabilities of events A and B respectively, $P(A|B)$ is the conditional probability of A given B and $P(B|A)$ vice versa.

3.3 Feature Selection Algorithms

The generalizability of filter FS methods makes it easier to compare their impact on multiple learning algorithms since the resulting subset of features are derived independently of any particular learning algorithm unlike wrapper and embedded FS methods. Filter methods are also less time consuming compared to wrapper methods because they do not require an iterative training of a learning algorithm to select a subset of features of a dataset. For these reasons, eight filter methods (Table 3.1) were studied in this work and are presented below.

FS method	Evaluation criteria	Supervision perspective	Selection strategy
RFS	Sparse learning-based	Supervised	Filter
ReliefF	Similarity-based	Supervised	Filter
SPEC	Similarity-based	Supervised, Unsupervised	Filter
Gini index	Statistical-based	Supervised	Filter
CFS	Statistical-based	Supervised	Filter
CMIM	Information theoretical-based	Supervised	Filter
JMI	Information theoretical-based	Supervised	Filter
MRMR	Information-based	Supervised	Filter

Table 3.1: FS methods studied and their respective perspectives.

Conditional Mutual Information Maximization (CMIM) is an information based FS method which iteratively picks features that maximize mutual information to the class feature conditioned on the already selected features [18, 19]. Where S is the subset of already selected features, Y the class feature, and X_k an unselected feature [6, 16], the feature score of X_k can be defined as:

$$J_{CMIM}(X_k) = \min_{X_j \epsilon S} I(Y; X_k | X_j). \tag{3.6}$$

This approach ensures that the selected features are good predictors of the class label but minimally redundant.

Joint Mutual Information (JMI) is an information based FS method which iteratively selects features that complement already selected features with respect to the class feature. Its feature score is defined as follows:

$$J_{JMI}(X_k) = \sum_{X_j \epsilon S} \{I(Y; X_k | (X_j)\}. \tag{3.7}$$

Minimum Redundancy Maximum Relevance (MRMR) is also an information based FS method proposed to select features that balance relevance and redundancy. A feature X_k is scored as:

$$J_{MRMR}(X_k) = I(Y; X_k) - \frac{1}{|S|} \sum_{X_j \epsilon S} I(X_k; X_j), \tag{3.8}$$

where the mutual information between the feature and target feature $I(Y; X_k)$ is the measure of relevance while $\frac{1}{|S|} \sum_{X_j \epsilon S} I(X_k; X_j)$ tells the features redundancy compared to already selected features [16].

Efficient and Robust Feature Selection (RFS) is a sparse learning based FS method that uses a joint $l_{2,1}$-norm minimization on both the loss

function and the regularization. This approach is more robust to noise and achieves group feature sparsity [6]. The objective function for RFS is:

$$\min_{W} ||XW - Y||_{2,1} + ||W||_{2,1}. \tag{3.9}$$

ReliefF is a similarity based FS method which extends the classic relief FS method [20] for not just binary, but also multi-class classification [21]. The idea of this method is to select features that maximally distinguish between instances that are near to each other [22]. The score of a feature X_k is defined as:

$$\begin{aligned} J_{ReliefF}(X_k) = \frac{1}{c}\sum_{j=1}^{l}(-\frac{1}{m_j}\sum_{x_r \epsilon NH(j)} d(X(j,i) - X(r,i)) \\ + \sum_{y \neq y_j} \frac{1}{h_{yj}} \frac{P(y)}{1-P(y)} \sum_{x_r \epsilon NM(j)} d(X(j,i) - X(r,i))), \end{aligned} \tag{3.10}$$

where l is the number of randomly selected instances ($\leq$ n total instances), $NH(j)$ of size m_j is the set of instances closest to instance x_j and in the same class,
$NM(j,y)$ with size h_{jy} is the set of instances closest to x_j and in class y, and $P(y)$ is the probability of instances belonging to class y.

Spectral feature selection (SPEC) is a similarity based FS method which works for both supervised and unsupervised tasks [6]. Given a dataset X with features $F_1, F_2 \ldots, F_m$, feature vectors $F_1, F_2 \ldots, F_m$, normalized feature vectors $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_m$ and target feature Y in the case of supervised tasks with class labels, the pairwise similarity of its instances can be encoded into a similarity matrix S using a similarity measure like the Radial Basis Function (RBF) kernel. With G being the representative graph of S, SPEC uses the structural information of G gotten from its spectrum[3] to determine the relevance of a feature. The idea of SPEC is to select features which preserve the intrinsic structure of the data by assigning similar values to instances that are close to each other [6]. The relevance of a feature can be

[3]The spectrum of a graph refers to its eigenvalues and their multiplicities.

scored by either of the following:

$$SPEC_{score1}(f_i) = \hat{\mathbf{f}}_i{}'\gamma(\mathbf{L}_{norm}), \hat{\mathbf{f}}_i = \sum_{j=1}^{n} \alpha_j^2 \gamma(\lambda_j)$$

$$SPEC_{score2}(f_i) = \frac{\hat{\mathbf{f}}_i{}'\gamma(\mathbf{L}_{norm})}{1-(\hat{\mathbf{f}}_i{}'\varepsilon_1)^2}, \hat{\mathbf{f}}_i = \frac{\sum_{j=1}^{n} \alpha_j^2 \gamma(\lambda_j)}{\sum_{j=1}^{n} \alpha_j^2} \tag{3.11}$$

$$SPEC_{score3}(f_i) = \sum_{j=1}^{m} (\gamma(2) - \gamma(\lambda_j))\alpha_j^2.$$

The SPEC scores are derived from the spectral decomposition of the normalized laplacian matrix L of G; in particular its trivial eigenpair ($eigenvalue = 0$, $eigenvector = D^{\frac{1}{2}}\mathbf{e}$). SPEC score2 is used in cases where $\hat{f}_i$ is very close to eigenvector 0 and score3 is used when the number of leading eigenvalues of L k that optimally separates G into K parts is known.

The selection of features is done in three steps [23]: firstly a similarity set S from the dataset is built followed by its representative graph. Afterwards, the relevance of the features are evaluated using the spectrum of the graph i.e. any of the scores in equation 2.10. Finally, the features are ranked in descending (score 3) or ascending (score 1 and 2) order in terms of feature relevance.

Gini index is a very common statistical measure used to quantify the ability of a feature to separate instances between classes [6] . A feature f_i with r distinct values is scored as:

$$J_{Gini}(f_i) = \min_{W}(p(W)(1-\sum_{s=1}^{c} p(C_s|W)^2) + p(\overline{W})(1-\sum_{s=1}^{c} p(C_s|\overline{W})^2)), \tag{3.12}$$

with W_{f_i} and $\overline{W_{f_i}}$ denoting sets of instances with f_i value $\leq \phi$ and $> \phi$ respectively. Where ϕ is some j^{th} value of f_i. This implies that ϕ separates the dataset into W_{f_i} and $\overline{W_{f_i}}$. The maximum gini score in binary classification is 0.5 and lower values imply higher relevance.

Correlation-based Feature Selection (CFS) is a statistical based FS method with the basic idea of selecting a feature subset S in which features are highly correlated with the class and uncorrelated with other selected features. This implies that redundant features are less likely to be selected. The CFS score of a feature subset dubbed 'merit'. Is defined as:

$$J_{CFS} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}, \tag{3.13}$$

where k is the number of features in S, $\overline{r_{cf}}$ is the average feature-class correlation and $\overline{r_{ff}}$ is the average feature-feature inter-correlation.

3.4 Classification Algorithms

The classification task is a common problem in practice; for instance, the bank wants to know if a loan applicant is likely to default or not before issuing a loan, the grocery store wants to automatically detect items that span multiple categories. Classification is broadly categorized into binary classification where subjects are assigned to one of two classes and multiclass classification where subjects are assigned to one of more than two classes.

In this work, four classification algorithms for binary and multiclass problems that do not have FS embedded were used to study the impact of FS on binary and multiclass classification; they are discussed below.

Naive Bayes (NB) is a collection of classification algorithms that are based on Bayes' theorem and the assumption that every pair of features X_i, X_j in a dataset are conditionally independent given the class feature Y i.e.:

$$P(X_i \cap X_j|Y) = P(X_i|Y)P(X_j|Y). \tag{3.14}$$

NB classifies an instance to the most probable class value y given its feature vector $\langle x_1, x_2, \ldots, x_n\rangle$ i.e.

$$y = argmax P(y_j|x_1, x2, \ldots, x_n). \tag{3.15}$$

Applying bayes' theorem,

$$y = argmax \frac{P(x_1, x_2, \ldots, x_n|y_j)P(y_j)}{P(x_1, x_2, \ldots, x_n)}. \tag{3.16}$$

Since $P(x_1, x_2, \ldots, x_n)$ is constant for all y_js,

$$y = argmax P(x_1, x_2, \ldots, x_n|y_j)P(y_j). \tag{3.17}$$

With the assumption of independence of features, $P(x_1, x_2, \ldots, x_n|y_j)P(y_j) = \prod_{i=1}^{n} P(x_i|y_j)$, hence,

$$y = \prod_{i=1}^{n} P(x_i|y_j)p(y_j). \tag{3.18}$$

NB is commonly applied in fields like text mining because of its efficiency on large datasets [24].

K-Nearest Neighbours (KNN) is a commonly used classification algorithm that works on the bird of the same feather flock together principle. The class of an unknown instance is determined by that of its K closest neighbors whose instances are known. KNN is considered a lazy algorithm because it does not build a classifier once for reuse rather it scans through

```
for all unknown instance (S_i)
    for all known instance (S_j)
        Compute the distance between S_i and S_j
    end for
    Sort the distances and pick the instances of the K smallest distances.
    Assign S_i to the class of the mode of the classes of the selected instances.
end for
```

Figure 3.1: K Nearest Neighbour classification pseudo-code.

the training data to determine the class if an unknown instance occurs each time it is called [25]; this makes it computationally expensive. Figure 5.4b shows a pseudo-code of the KNN algorithm.

Linear Discriminant Analysis (LDA) is a classic classification algorithm that uses linear decision boundary(ies) to build a classifier for unknown instances. Its closed-form solution[4] with no need for hyperparameter tuning makes it easier to compute and it has demonstrated good performance in practice [26]. Given an unknown instance x, the probability of x belonging to a class k of the class feature Y is estimated by:

$$P(Y = k|X = x) = \frac{P(k)P(x|k)}{\sum_{l=1}^{k} P(l)P(x|l)}. \tag{3.19}$$

However, $p(x|y)$ can be estimated with a Gaussian distribution function which when substituted leads to a simplified discriminant function for class k given x defined as:

$$D_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \ln(P(k)), \tag{3.20}$$

where μ_k is the mean value of x for the class k and σ^2 is the variance across all inputs x [27]. LDA works with the assumption that the data is normally distributed and that the variance of all features is equal. It can also be used for DR by projecting the training data into a linear subspace such that the separability between classes is maximized [26].

Multilayer Perceptron (MLP) is a type of feedforward artificial neural network consisting of three layers of nodes; an input layer, output layer[5], and an arbitrary number of hidden layers (Figure 3.2). A full pass of data (epoch) through the MLP comprises forward and backward passes. In the forward pass, data is passed from the input layer through the hidden layer(s) to the

[4]A closed-form solution (expression) is any formula that can be evaluated in a finite number of standard operations.

[5]The number of classes in the target feature is the number of nodes in the output layer.

output layer adjusting the weights of the nodes based on the ground truth. The backward pass goes in the opposite direction from the output layer to the input layers to minimize the error of the MLP. The classification model is built by several epochs of the training data through the MLP after which it is used to classify unknown instances. MLP is applied in various fields

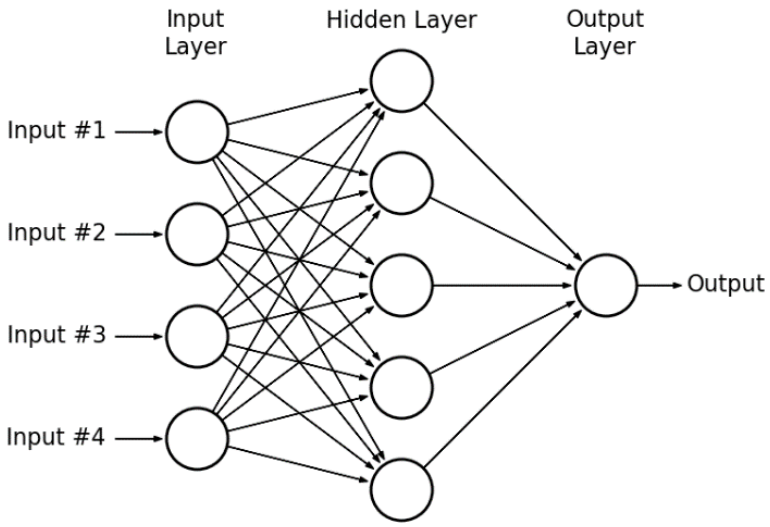


Figure 3.2: A hypothetical MLP with one input layer, a single hidden layer, and an output layer (source: [1]).

including speech and image recognition, and natural language processing. [28] This classifier has the option of regularization which can be seen as embedded FS however this feature was not used in this work.

Chapter 4

Methods

To evaluate the performance of FS on classification, we followed a systematic approach in defining the components of the experiments to be executed as well as selecting the datasets to be used. These methods are presented in the following sections.

4.1 Performance Evaluation

In studying the impact of FS on classification, more factors than can be studied in this work can be analyzed for the FS methods, classifiers, and datasets. It is therefore important to clearly and methodically define the components of the experiments as failing to do so can lead to mistakes like unrepresentative workload, erroneous analysis, too complex analysis, etc. The systemic approach for performance evaluation (Figure 4.1) proposed in [29] was followed to avoid common mistakes.
In what follows, we present the details of the ten steps in the systematic approach to performance evaluation.

1. **State Goals and Define the System**
   The system under performance evaluation consists of the datasets (before and after FS), the FS methods, and the learning algorithms (Figure 4.2). The **target** defined by the subsystem $\langle$*Full Dataset* $\rightarrow$ *FS Method* $\rightarrow$ *Filtered Dataset* $\rightarrow$ *Learning Algorithm*$\rangle$ is benchmarked against the **baseline** of the system defined as $\langle$*Full Dataset* $\rightarrow$ *Learning Algorithms*$\rangle$.
   In order to answer the research questions stated in section 1.2, the goals of the experiments conducted are to:

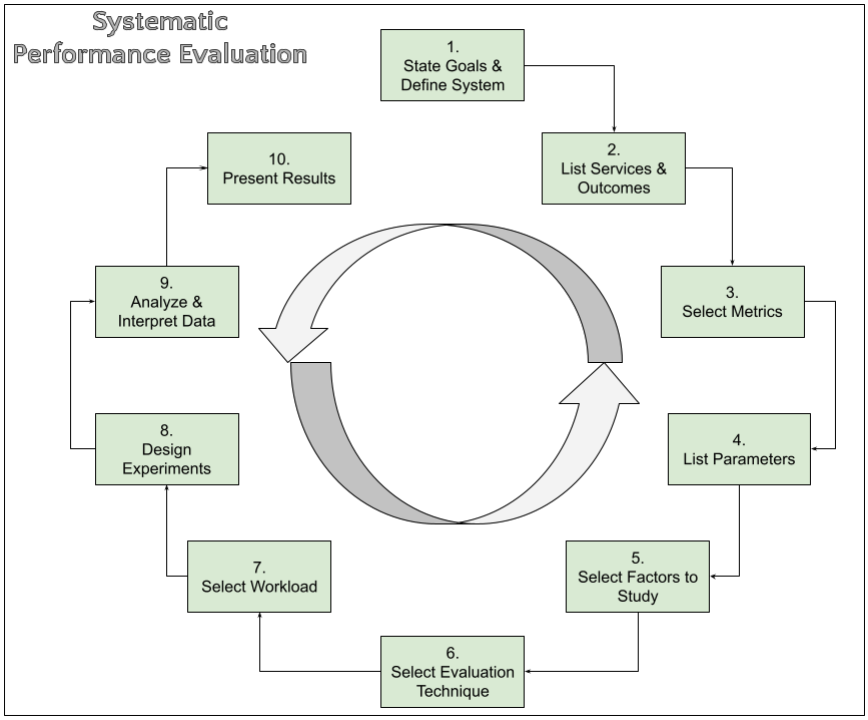


Figure 4.1: Ten steps of a systematic approach to performance evaluation of a system.

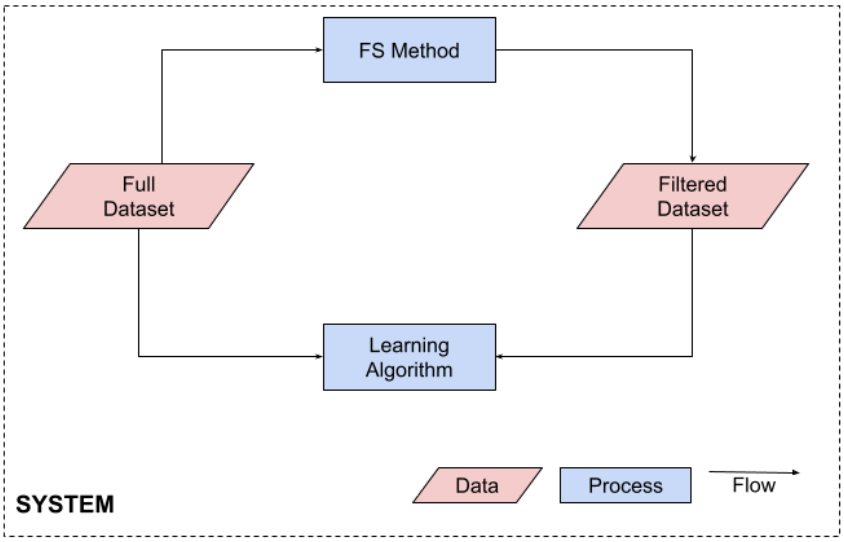


Figure 4.2: System definition showing components and flow.

- Measure the performance of the classifiers built in the baseline and target.
- Measure the execution time (in seconds) of FS on each dataset.
- Measure the model training time (in seconds) of the classifiers in the baseline and target.

2. **Services and Outcomes**
   The services of the system are delivered by the two processes; the FS method and the learning algorithm. Depending on the FS method, one of the first two services are provided:

   - Selection of a fixed size of feature subset.
   - A ranking of all features by relevance.
   - The learning algorithm provides the service of building a classifier from the dataset.

   The outcome from these services are feature subset/ranked features and a classifier.

3. **Metrics**
   Metrics are the criteria used to evaluate the performance of the system. The metrics used to evaluate the performance of the defined system (Figure 4.2) are:

   - **FS runtime**, which is the time taken to derive the feature subset/ranked features.
   - **Model training runtime change** is the ratio of the difference in time between training the baseline classifier and target classifier to the training time of the baseline classifier defined as:

$$Runtime\ change^{1} = \frac{Base\ runtime - Target\ runtime}{Base\ runtime}. \quad (4.1)$$

   - **Classifier accuracy change** is the ratio of the difference between the accuracy of the target classifier and the baseline classifier to the accuracy of the baseline classifier expressed as:

$$Accuracy\ change^{2} = \frac{Target\ accuracy - Base\ accuracy}{Base\ accuracy}. \quad (4.2)$$

[1]There is an expected decrease in runtime i.e., Base runtime > Target runtime.
[2]There is an expected increase in accuracy i.e., Target accuracy > Base accuracy.

Where accuracy is the fraction of all predictions that were correct; defined as:

$$Accuracy = \frac{Correct\ predictions}{All\ predictions}. \tag{4.3}$$

4. **Parameters**
   Several characteristics of the system components affect the performance of the system. These are called parameters and are categorized into workload and system parameters. The workload parameters are associated with the dataset and algorithms and include:

   - Dataset parameters
     - Number of feature
     - Number of instances
     - Number of classes
     - Dimensionality, which is the feature-instance ratio of a dataset defined as:

$$Dimentionality = \frac{Number\ of\ features}{Number\ of\ instances}. \tag{4.4}$$

     - Class balance, this is the distribution of instances amongst the classes of the target variable.It is measured by the Shannon entropy which is defined as:

$$Class\ balance = -\frac{\sum_{i=1}^{k} \frac{c_i}{n} \log(\frac{c_i}{n})}{\log k}, \tag{4.5}$$

       where the dataset has n instances with k classes in the target variable of size c_i. A perfectly balanced dataset with equal number of instances in each class has a class balance of one while an extemely unbalanced dataset has a class balance of zero.
     - Class Entropy, this measures the entropy (3.1) of the classes in the target feature.
     - Average feature correlation, is the measure of interdependence between all features and is quantified by

$$p = \frac{1}{T} \sum_{i=1}^{k} \sum_{j=1}^{m-1} \sum_{l=j+1}^{m} |P_{jl}|, \tag{4.6}$$

       where P_{jl} is the Pearson's correlation coefficient of features j and l, T is the total number of P_{jl}'s summed, k is the number

of classes in the target feature, and m is the number of features in the dataset. A p value of zero indicates independence of all features while a value of one indicates high correlation between features and thus feature redundancy.

- Algorithm parameters
  - Number of algorithms
  - Features subset size
  - Learning algorithm hyper-parameters [30]

While the system parameters are the characteristics of the hardware used for the system and include among others:

- Processor speed
- RAM/Disk size
- Operating system context switching overhead

5. **Factors to Study**
   The number of parameters controlled in an experiment determines the workload and the time of the experiment. Considering the limited time for experimenting, a subset of the parameters must be chosen to be studied. The five factors; *Number of features, Number of instances, Number of classes, Class balance, and Feature subset size* were considered more important and chosen to be studied in this work because repositories can readily be filtered by them, making data selection straightforward since the goal is to have representative datasets for the experiments.

   Also, the other workload parameters either depend on the studied factors (e.g., dimensionality is simply the feature-instance ratio) or were considered less important; e.g., most FS methods do not factor in the average correlation of features during FS but rather access features individually thus it not included in the study.

   The system factors were kept fixed as one machine was used for the whole experiment. The context switch overhead was acknowledged but not studied. Hence 10 fold cross-validation was employed to ensure results are as close to accurate as possible. The default learning algorithm hyper-parameters were used; since the same chosen hyper-parameters will be used for both baseline and target classifiers, changing the hyper-parameters is unlikely to affect the relative improvements in training time and accuracy between the baseline and target classifier derived

from FS. Also, controlling all parameters for all algorithms would explode the design space and we would not be able to complete them in the limited time frame.

6. **Evaluation Technique**
   The studied FS methods and learning algorithms were implemented in the scikit-feature[3] repository and scikit-learn[4] library respectively. The scikit-learn library also provides a metrics library with which the accuracy of the classification models is measured. The experiments were done in Python and the time library was used to measure the runtime of code execution.

7. **Workload**
   This refers to the datasets used in the evaluation of the system. With the four factors studied being dataset factors and each having two categories (see Section 4.2), two real-world datasets from OpenML were selected per 16 possible combinations of these factors, leading to a total of 32 datasets. Only datasets with no missing values were considered as candidates to save time in the pre-processing stage in addition to the potential effect of these missing values on the performance of the classifier especially in the case that insstances with missing values are dropped. Lastly, datasets with only numeric values were considered due to the input constraint by the FS methods studies.

8. **Design Experiments**
   Out of the eight FS methods studied, Gini, RFS, ReliefF, SPEC, and CFS methods return a ranking of the features in order of relevance while JMI, MRMR, and CMIM return a subset of the features according to the specified size. Five possible subset sizes were studied; specifically $(number\ of\ features)^i$, for i $\in$ [0.5, 0.6, 0.7, 0.8, 0.9]. This means that a total of 20 runs (Table 4.1) are required for the eight FS methods. The four classification algorithms considered in this work require a single run. Therefore with all 32 datasets, a full factorial experimental design with 32*20*4 = 2,560 experiments will be used.

9. **Analyze and Interpret Data**
   To derive insights from the results of the experiments Z-test and ANOVA statistical methods were used to analyse the results (i.e., measured metrics). For each of these tests, a null and alternative hypothesis were

[3]https://github.com/jundongl/scikit-feature
[4]https://scikit-learn.org/stable

Feature selection Method	Required runs
Gini, RFS, ReliefF, SPEC,CFS	1
JMI, MRMR, CMIM	5
Total	**20**

Table 4.1: Required number of runs for each feature selection method.

formulated and based on the test result, we either reject or fail to reject the null hypothesis.

10. **Present Results**
    A combination of tables, bar charts, scatter plots, box plots, and other visualizations was used to present the conclusions from the analysis of the results of the experiments.

4.2 Data Selection

Several data repositories provide open data which enables research; OpenML is one and was used in this work. Considering the limitations of time and the input constraint of some of the algorithms studied, not all datasets in OpenML (over 21,000) could be used.
Available datasets were pruned based on the following eligibility criteria:

- Number of features $\leq$ 1.000
- Number of instances $\leq$ 10.000
- Number of classes [2,19]
- No missing values
- Numerical types (excluding timestamps)
- Single target variable

After pruning, there were 380 candidate datasets available and the metadata of these datasets showed a wide range of values for the factors to be studied; for example, the number of features of the datasets ranged from 2 to 971. Since not all datasets can be used in the study, it is important to pick a subset that is a proper representation of these datasets. To achieve this, each factor in the metadata of the candidate datasets was discretized into two bins. Although alternatively the factors can be discretized to more than

two bins, in order to have an experimental design that is achievable within the available time, we discritized to only two since increasing the number of bins exponentially increases the full factorial of the experimental design. With two bins and 4 dataset factors, we had $16(2^4)$ possible factor combinations all of which we need representative datasets. However, some of these combinations of had as few as three datasets only; increasing the number of bins will result in more factor combinations some of which will likely have no representative datasets.

Quantile, uniformed, and clustering discretization methods were explored on the datasets metadata as is and on the log values (for factors *number of features* and *number of instances* due to the large range of values) while number of classes were simply classified into binary (two classes) and multiclass (more than two classes). In what follows, the results from each discretization method is discussed.

4.2.1 Discretization

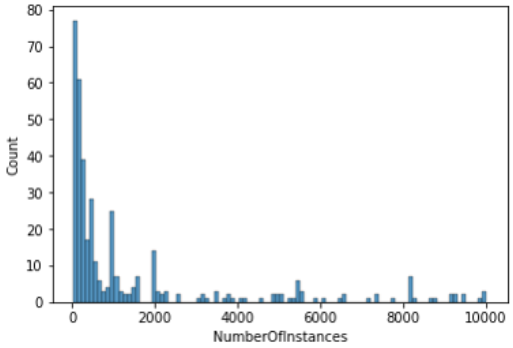
Quantile method discretized the input into equally sized bins. The results of discretization on the values as is and the log values were the same as shown in Table 4.2.

Factor	Bin	Number of datasets
Number of instances	0: [27, 383]	187
	1: [400, 9.989]	193
Number of features	0: [2, 12]	187
	1: [13, 971]	193
Class balance	0: [0,0385, 0,9877]	188
	1: [0,9881, 1,0]	192

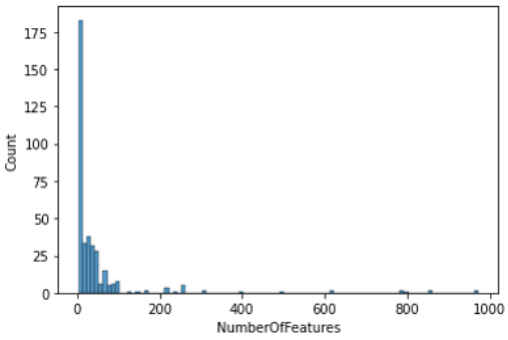
Table 4.2: Discretization of values of studied factors using the quantile method.

The result, although balanced in terms of the number of datasets in each bin, does not reflect the actual distribution of the datasets which is indeed not uniform as shown in the Figure 4.3.

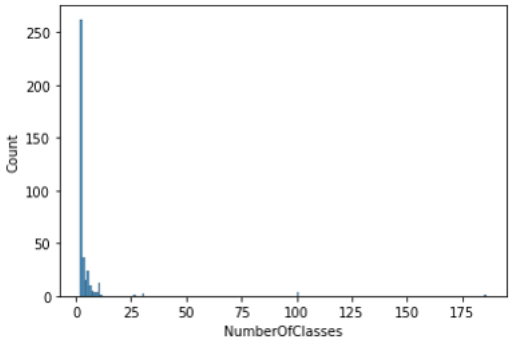
The uniformed approach returns each bin of the same length. In the case of discretization of the values as is the result seen in Table 4.3 while discretizing the log values results in Table 4.4. As with the case of quantile discretization, the actual distribution of the datasets is not reflected in the results; leading to factor combinations with no candidate datasets.



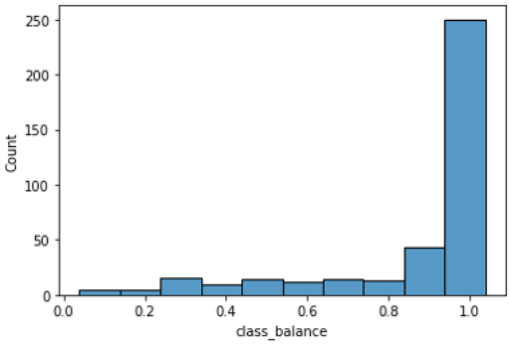
(a) Distribution of number of instances.



(b) Distribution of number of features.



(c) Distribution of number of classes.



(d) Distribution of class balance.

Figure 4.3: Distribution of values of factors studied in datasets.

Factor	Bin	Number of datasets
Number of instances	0: [27, 5.000]	338
	1: [5.100, 9.989]	42
Number of features	0: [2, 401]	370
	1: [501, 971]	10
Class balance	0: [0,0385, 04899]	45
	1: [0,5294, 1,0]	335

Table 4.3: Discretization of values of studied factors using the uniformed method.

Lastly, the clustering discretization method takes into account the intrinsic distribution of the data values in the creation of bins. Discretizing the values as is and after log resulted in the bins shown in tables 4.5 and 4.6 respectively.

Factor	Bin	Number of datasets
Number of instances	0: [27, 508]	221
	1: [45, 971]	159
Number of features	0: [2, 42]	287
	1: [501, 971]	93
Class balance	0: [0,0385, 04899]	45
	1: [0,5294, 1,0]	335

Table 4.4: Discretization of log values of studied factors using the uniformed method.

The discretization results from applying clustering on the log values was used to guide the selection of the datasets from OpenML because it best describes the distribution of the datasets in the repository and contained candidates for all 16 (2^4) combinations of the studied factors.

Factor	Bin	Number of datasets
Number of instances	0: [27, 2.600]	320
	1: [3.107, 9.989]	60
Number of features	0: [2, 618]	373
	1: [785, 971]	7
Class balance	0: [0,0385, 0,8083]	84
	1: [0,8232, 1,0]	296

Table 4.5: Discretization of values of studied factors using the clustering method.

After discretization, two datasets were selected randomly for each of the 16 factor combinations. To make the datasets as dissimilar as possible, data sets were chosen from each side of the diagonal after visualizing the feature - instance in the combination as shown in Figure 4.4. The final 32 datasets used in this study are outlined in Table 4.7.

Factor	Bin	Number of datasets
Number of instances	0: [27, 846] 1: [937, 9.989]	246 134
Number of features	0: [2, 16] 1: [19, 971]	205 175
Class balance	0: [0,0385, 0,8083] 1: [0,8232, 1,0]	84 296

Table 4.6: Discretization of log values of studied factors using the clustering method.

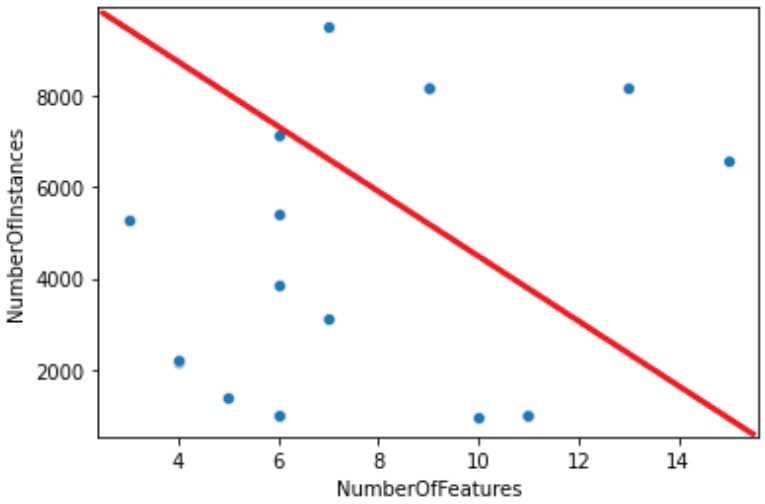


Figure 4.4: Plot of datasets with Class(0), Features(0), Instances(1), and Class balance(1)

Factors[1]	Dataset	Classes	Features	Instances	Class Balance
0000	confidence	2	4	72	0.055
0000	blood-transfusion-service-center	2	5	748	0.0067
0001	fri_c3_250_10	2	11	250	0.9954
0001	disclosure_z	2	4	662	0.9981
0010	page-blocks	2	11	5473	0.4763
0010	wilt	2	6	4839	0.3029
0011	delta_elevators	2	7	9517	1
0011	stock	2	10	950	0.9995
0100	synthetic_control	2	61	600	0.6500
0100	ar4	2	30	107	0.6950
0101	isolet	2	618	600	1
0101	fri_c4_250_100	2	101	250	0.9896
0110	mfeat-zernike	2	48	2000	0.4690
0110	clean2	2	169	6598	0.6201
0111	gina	2	971	3153	0.9998
0111	philippine	2	309	5832	1
1000	Engine1	3	6	383	0.5905
1000	heart-h	5	14	294	0.7065
1001	LED-display-domain-7digit	10	8	500	0.9971
1001	heart-long-beach	5	14	200	0.9365
1010	wine-quality-white	7	12	4898	0.6632
1010	volcanoes-d3	5	4	9285	0.1761
1011	JapaneseVowels	9	15	9961	0.9881
1011	wall-robot-navigation	4	5	5456	0.8573
1100	meta_all.arff	6	63	71	0.6913
1100	meta_ensembles.arff	4	63	74	0.7719
1101	synthetic_control	6	61	600	1
1101	robot-failures-lp3	4	91	47	0.9102
1110	Indian_pines	8	221	9144	0.6941
1110	cardiotocography	3	36	2126	0.614
1111	cnae-9	9	857	1080	1
1111	texture	11	41	5500	1

[1] ***Factors*** is a 4 digit representation of the studied parameters which stands for Class, Feature, Instances, and Class balance sequentially.

Table 4.7: Experimental datasets and their properties.

Chapter 5

Evaluation

To evaluate the FS runtime, model training runtime change, and classifier accuracy change of the system (Figure 4.2), the runtime for FS on each dataset was measured during the experiments as well as the training time and accuracy for the base and target classifiers built. We describe the experimental setup and execution in sections 5.1 and 5.2 respectively followed by the results of the experiment and a discussion in sections 5.3 and 5.4 respectively.

5.1 Experimental Setup

All experiments were executed on a Dell XPS laptop with an Intel Core i7-1065G7 processor, 16GB RAM, and 512 GB storage. The experiments were executed serially while also limiting other processes on the machine.
Although 32 datasets were selected for the experiments, 31 datasets were used to evaluate the CFS method. Due to its very high computational cost, it was unable to finish FS on the largest dataset which had 971 features, 3153 instances, and 2 classes after 9 days.

5.2 Execution

To begin, all features of the dataset were discretized using the uniformed discretization method; setting the number of bins to $q = \max\{\min\{\frac{n}{3}, 10\}, 2\}$ where n is the number of instances in the dataset [16]. Discretizing the features is done to meet the input constraint of most FS methods which work with only discrete features.
Next, the dataset was split into ten partitions; with the training data comprising nine partitions and the test data, one partition. After this split, FS

was done on the training data; the entire training data was used to build the base classifier while the training data containing only the selected subset of features were used to build the target classifier. Both classifiers were tested on the test data and the accuracy was measured. The FS, model building, and testing are repeated ten times for each dataset (i.e., 10-fold cross-validation); each time a new partition is used as the test data while the remaining nine are used as the training data.
Using 10-fold validation leads to results that are more robust against overfitting, bias, and variation in runtime. At the end of each of the 10-fold execution, the median runtime and average accuracy were recorded. The median was used in the runtime for robustness against possible variations due to context switching or other sources.

Due to the poor stability of some FS methods, different features might be selected with a change of data samples. Hence, FS and model building were executed together for each fold to ensure that the performance evaluation is done on the same data for both FS and classification. Figure 5.1 gives an overview of the execution described from discretization to measure recording. The Python source code for the experiments are publicly available on https://github.com/F-U-Njoku/Thesis-BDMA2021-Njoku.

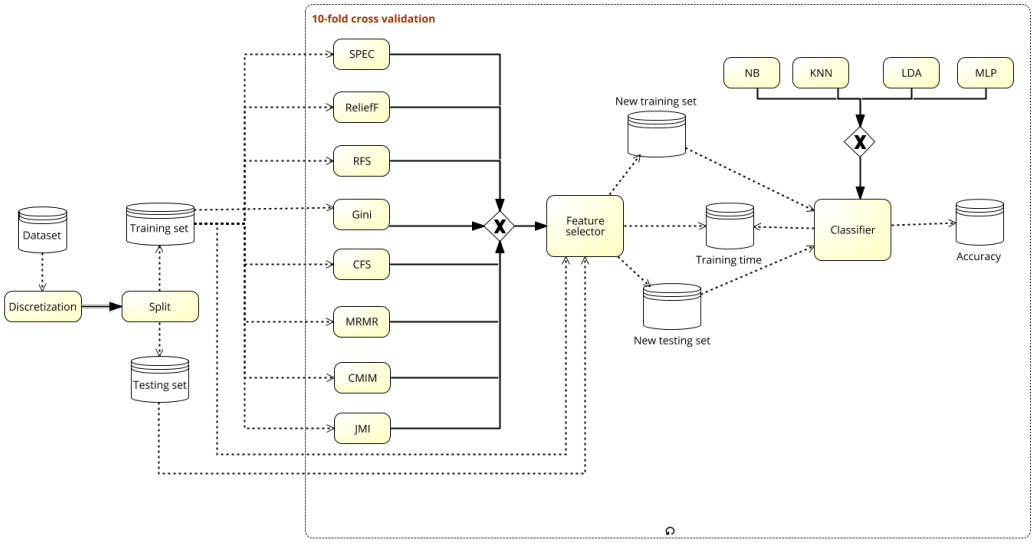


Figure 5.1: Experiment design, components, and flow.

5.3 Results

5.3.1 Runtime

The runtime of an FS method is affected by various factors and at differing proportions. Therefore, some may be influenced more by the number of features while others will be affected more by the number of instances and so on. The theoretical runtime of the studied FS methods are presented in Table 5.1, this shows the factors which affect runtime for each FS method, where S, m, c, n, and t are the feature subset size, number of featues, number of classes, number of instances, and $\min(m, n)$ respectively.

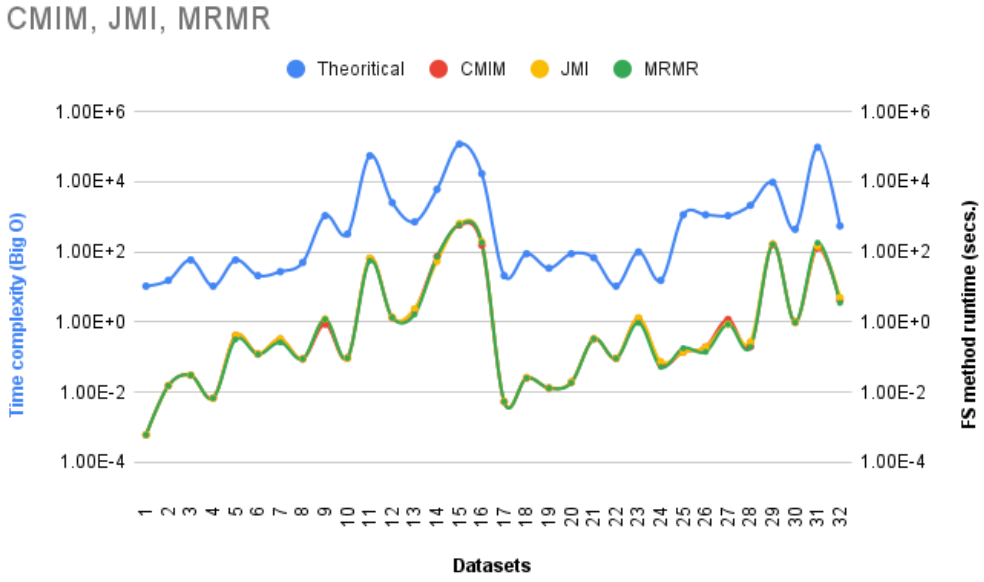
Algorithm	Time complexity (theoretical)	Source
ReliefF	$O(n^2.m)$	[31]
MRMR	$O(\|S\|.m)$	[32]
CMIM	$O(\|S\|.m)$	[19]
SPEC	$O(n^2.m)$	[23]
Gini	O(m.c)	[6]
RFS	$O(t.(3n^2m + m^2n + mnc + m))$	[6]
JMI	$O(\|S\|.m)$	[6]
CFS	$O(n((m^2 - m)/2)+$ $(m^2 - m)/2 + \|S\| + (\|S\|^2 - \|S\|)/2)$	[33]

Table 5.1: Theoretical runtime complexities of FS methods.

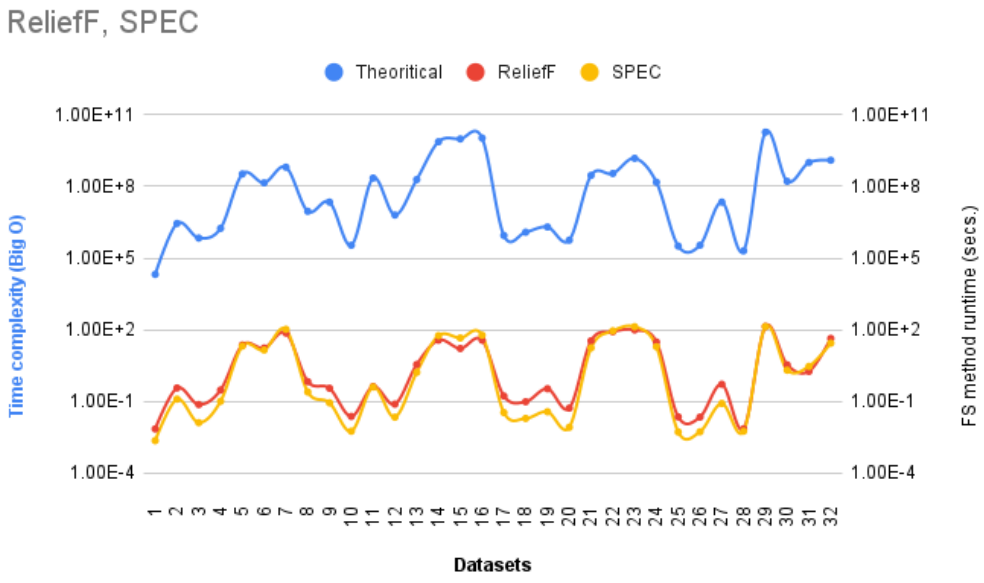
CMIM, JMI, and MRMR all share the same theoretical runtime with the number of features and the subset size being the factors that predominantly influence their runtime. The runtime of SPEC and ReliefF is controlled by the number of instances and features with the number of instances having a higher impact on the runtime. RFS and CSF have longer runtimes due to the matrix multiplication operations that are required while Gini has the lowest runtime and is influenced mainly by the number of features then classes.

Figure 5.4 shows the recorded runtime of CMIM, JMI, MRMR, SPEC, ReliefF, and SPEC recorded from the experiments against the theoretical runtime presented in Table 5.1. This is to confirm that the implementations are aligned to the theory and ensure that results of the experiment are as expected. The theoretical-actual comparison of the runtime of other methods can be found in the appendix.

For learning algorithms like LDA that build classifiers after training, FS reduces the model training runtime as expected. There is over 50% training



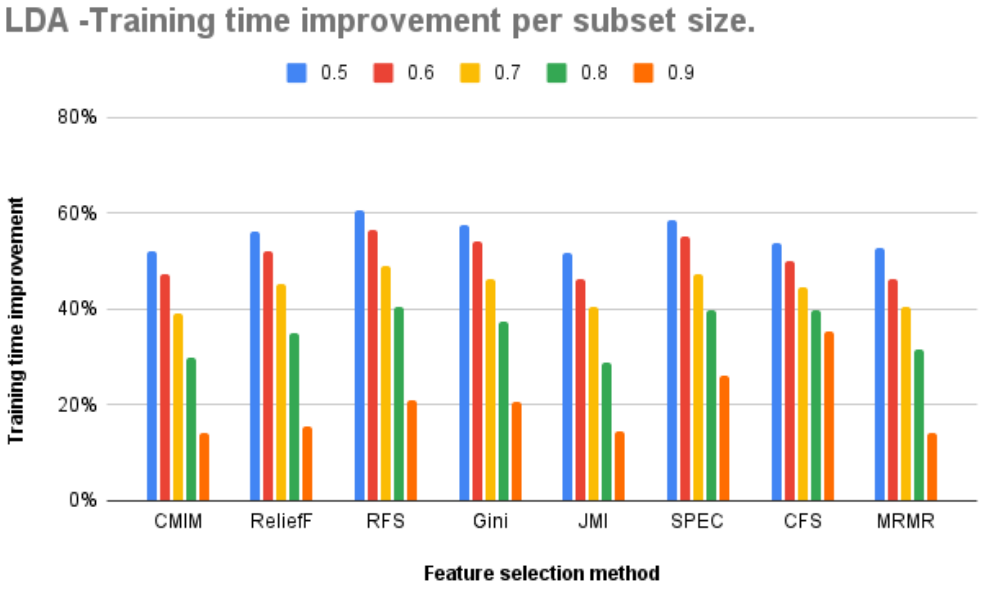
(a) CMIM, JMI, MRMR runtimes.



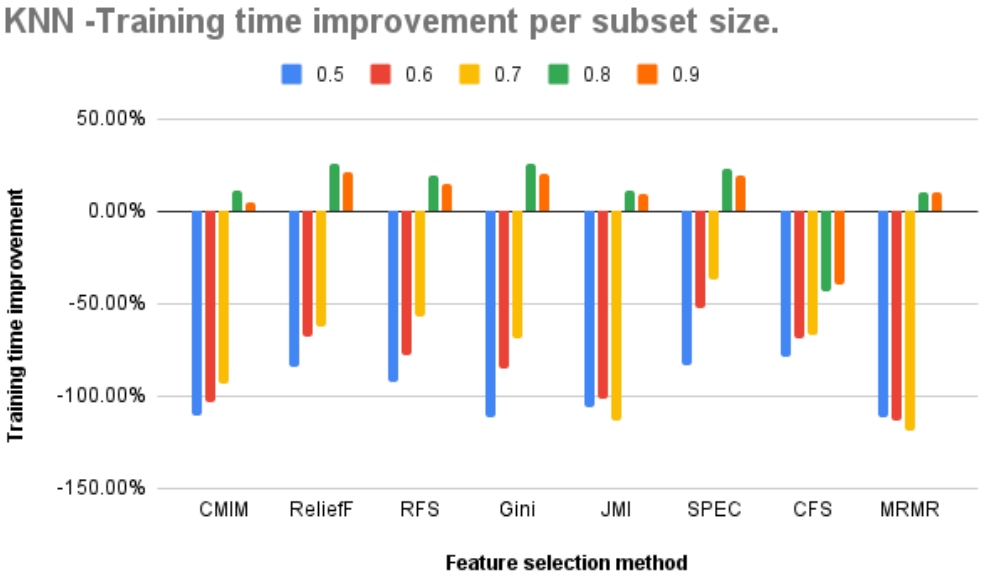
(b) ReliefF, SPEC runtimes.

Figure 5.2: Theoritical VS actual FS runtime.

time reduction for LDA when the subset size is set to $(number\ of\ features)^{0.5}$ (Figure 5.4a). However, KNN as reported is a lazy algorithm meaning it does



(a) LDA training time improvement.



(b) KNN training time improvement.

Figure 5.3: The improvement in the training time of LDA and KNN classifers for varrying sizes of selected feature subset.

not build a classifier at the point of training, rather it stores useful metadata that is used at the point of classification. Not following the pattern of other algorithms, KNN appeared to give the best training time improvement with a subset size of $(number\ of\ features)^{0.8}$ (Figure 5.4b).

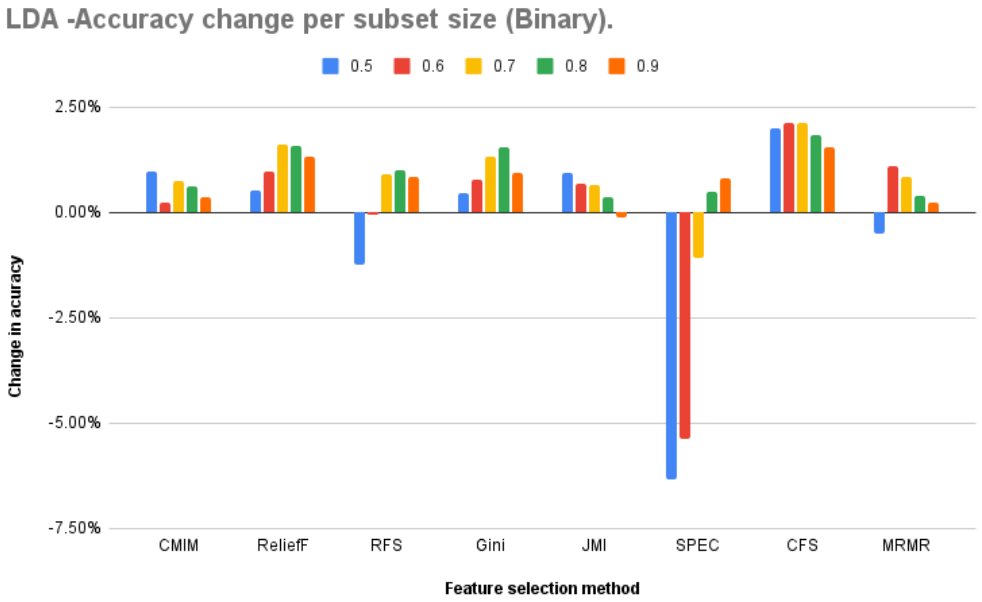
5.3.2 Statistical Analysis

To derive insights from the measures recorded during the experiments, a number of statistical tests were conducted which are presented in this section. All statistical tests were performed at an alpha level of 0.05.

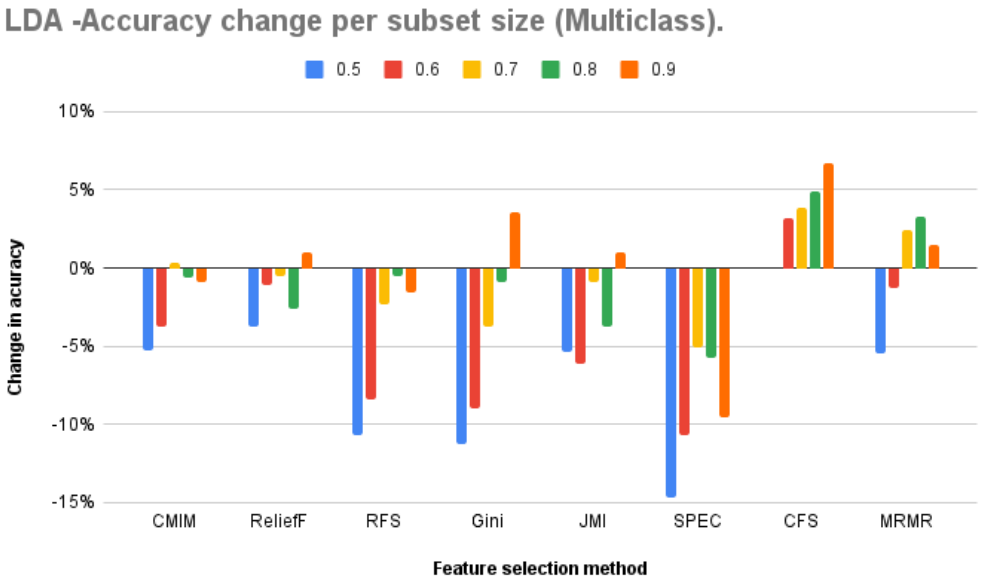
One of the objectives for FS is to build models with improved performance that generalize the data better. Since FS is applicable to both binary and multiclass classifications, we expect an improvement in the accuracy of the classifiers after FS for both binary and multiclass classifications. Hence, we will be investigating the effect of classification type on the accuracy change obtained after FS. This leads us to the following hypothesis:
H_0: The classification type does not influence the change in accuracy derived from applying FS.
H_1: The classification type influences the change in accuracy derived from applying FS.
By visualization, we see a difference in the accuracy change obtained for binary and multiclass classifications (Figure 5.4) and this is irrespective of the FS method used (Figure 5.5).

To test the aforementioned hypothesis, we consider two independent groups. The binary and multiclass groups, each made up of the change in accuracy for 16 datasets applying the eight FS methods and building four base and target classifiers (512 result points). Testing the priori hypotheses using Z-test, the results show the impact of FS on binary classification in terms of accuracy change ($\mu = 0.0058, \sigma = 0.0506, N = 508$) was significantly higher than on multiclass classification ($\mu = -0.0627, \sigma = 0.1483, N = 512$), $Z = 9.2589$, $p < 0.001$. Hence, going forward, the results from binary and multiclass classification were analysed separately.

Besides the type of classification, the classification algorithms differ and thus could interact with the FS methods to give different results in terms of change in accuracy. Hence, the changes in accuracy for binary classification were subjected to a two-way analysis of variance with FS methods having eight (CFS, CMIM, Gini, JMI, MRMR, ReliefF, RFS, SPEC) levels and classifiers having four (NB, KNN, LDA, MLP). Both effects were found to be statistically significant. The main effect of the FS method yielded an F ratio of $F(7, 354) = 3.5833$, $p < 0.001$ indicating that the mean change in accuracy achieved by applying each FS method differed significantly. The



(a) Binary accuracy improvement.



(b) Multiclass accuracy improvement.

Figure 5.4: The improvement in the accuracy of LDA classifer for varrying sizes of selected feature subset.

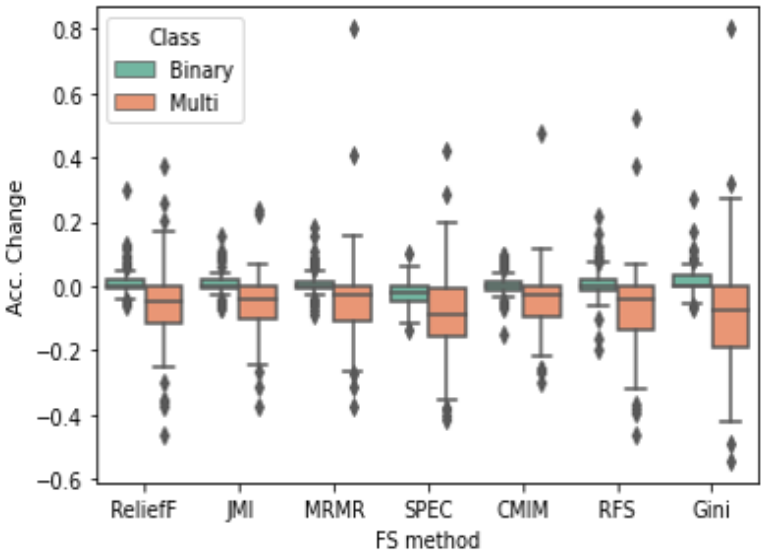
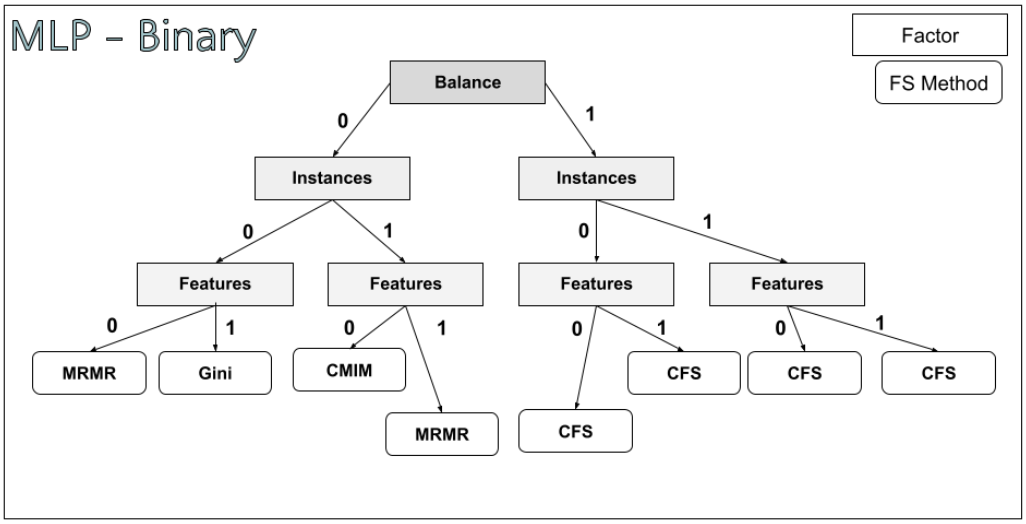


Figure 5.5: The improvement in the accuracy of all classifers for varrying FS methods.

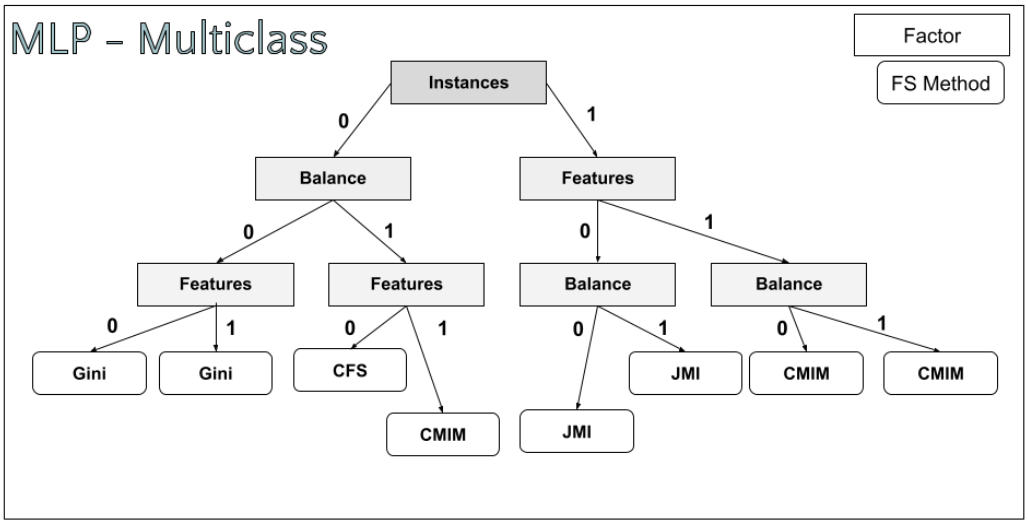
main effect of the classification algorithm was statistically significant yielding an F ratio of F(2, 354) = 3.33, p = 0.04. The interaction (FS method and classifier) effect was non-significant, F(14, 354) = 0.2802, $p > 0.05$.

The same two-way analysis of variance was done for multiclass classification, with FS methods having eight (CFS, CMIM, Gini, JMI, MRMR, ReliefF, RFS, SPEC) levels and classifiers having four (NB, KNN, LDA, MLP). The main effect of the classification algorithm was statistically significant with an F ratio of F(2, 357) = 7.69 and $p < 0.001$ while the main effect of the FS method was non-significant with an F ratio of F(7, 357) = 1.0402 and $p > 0.05$. The interaction (FS method and classifier) effect was also non-significant with F(14, 354) = 0.2802, $p > 0.05$.

In order to propose a guideline for choosing FS methods, given certain characteristics of a dataset, the FS method which yielded the highest accuracy change for each dataset-classifier pair was assigned as the preferred FS method for that pair. From this, we build a decision tree (Figure 5.6) to guide the choice of a FS method given the number of features, instances, classes, and class balance.



(a) Tree for MLP binary classifier.



(b) Tree for MLP multiclass classifier.

Figure 5.6: Decision trees showing the proposed FS method given the number of instances, features, and class balance.

5.4 Discussion

There is an expectation for improved runtime and accuracy after applying FS on a dataset. However, the runtime of most FS methods become an overhead causing the total time needed for FS and model building to increase beyond the time it takes to build the model without FS. Of the eight FS methods

studied, CFS had the highest runtime for feature selection taking an average of 4810 seconds for all datasets; this is due to the multiple computations of pairwise correlations between features required by this method. Second to CFS was RSF which took an average of 867 seconds. The other methods took less than 50 seconds on average with Gini being the most efficient in runtime taking less than a second on average for FS. The advantage however is that when FS is done once on a dataset the selected features can be used across various learning algorithms one of which can then be selected for use. The runtime of these FS methods is influenced by various factors. CFS and Gini are primarily influenced by the number of features, CMIM, JMI, and MRMR by the size of the feature subset and number of features, ReliefF and SPEC are more influenced by the number of instances in the dataset, while RFS by both the number of features and instances.

After FS, the improvement of accuracy of the target classifiers differs for binary and multiclass classifications. With binary classification, there is an average improvement of the accuracy while for multiclass classification, reducing the number of features generally leads to a decrease in accuracy. CMIM and CFS methods, however, yielded the best result for multiclass classification; giving the least decrease. Therefore, although FS generally results in a decrease in multiclass classification accuracy, in the case that the dataset is too large and FS is indeed needed, CMIM and CFS methods are most recommended.

With FS selection, there is no one method that is optimal for all cases as for various factors of a dataset, a different method might be used. From the results of the statistical tests, the classification type and algorithm significantly affect the change in accuracy obtained after FS. Therefore, for each classification algorithm, a guideline is proposed based on the number of features, instances, and class balance for binary and multiclass classifications. The SPEC method was not recommended for any case. Although it is applicable to both supervised and unsupervised learning tasks, because this method of FS is not done using the target feature, the selected features are less optimal than those selected by other methods. CFS on the other hand yielded good results in many cases because it uses the correlation of features to the target feature as well as the pairwise correlation of features for FS.

From the decision trees built, we propose the guidelines for selecting FS methods for binary classification (Table 5.2) and multiclass classification (Table 5.3) with the objective of maximizing change in classification accuracy.

Classifier	Features	Instances	Class Balance	FS_method
	0	0	0	JMI
	0	0	1	CFS
	0	1	0	CFS
	0	1	1	JMI
	1	0	0	GINI
	1	0	1	CFS
	1	1	0	CFS
NB	1	1	1	MRMR
	0	0	0	CMIM
	0	0	1	RFS
	0	1	0	RFS
	0	1	1	CMIM
	1	0	0	ReliefF
	1	0	1	CFS
	1	1	0	CFS
KNN	1	1	1	JMI
	0	0	0	CFS
	0	0	1	CFS
	0	1	0	Gini
	0	1	1	CMIM
	1	0	0	CFS
	1	0	1	Gini
	1	1	0	CMIM
LDA	1	1	1	Gini
	0	0	0	MRMR
	0	0	1	CFS
	0	1	0	CMIM
	0	1	1	CFS
	1	0	0	Gini
	1	0	1	CFS
	1	1	0	MRMR
MLP	1	1	1	CFS

Table 5.2: Proposed FS method selection guideline for binary classification.

Classifier	Features	Instances	Class Balance	FS_method
	0	0	0	CMIM
	0	0	1	CFS
	0	1	0	CMIM
	0	1	1	CMIM
	1	0	0	MRMR
	1	0	1	CFS
	1	1	0	MRMR
NB	1	1	1	CMIM
	0	0	0	CFS
	0	0	1	CFS
	0	1	0	CMIM
	0	1	1	JMI
	1	0	0	CMIM
	1	0	1	CFS
	1	1	0	MRMR
KNN	1	1	1	CMIM
	0	0	0	Gini
	0	0	1	CFS
	0	1	0	RFS
	0	1	1	CMIM
	1	0	0	CFS
	1	0	1	CMIM
	1	1	0	ReliefF
LDA	1	1	1	CMIM
	0	0	0	Gini
	0	0	1	CFS
	0	1	0	JMI
	0	1	1	JMI
	1	0	0	Gini
	1	0	1	CMIM
	1	1	0	CMIM
MLP	1	1	1	CMIM

Table 5.3: Proposed FS method selection guideline for multiclass classification.

Chapter 6

Conclusion

In this thesis, we studied the impact of eight filter FS methods on four classification algorithms using 32 real-world datasets. We found that on the average, FS improved the accuracy for classifiers of binary classification, however, for multiclass classification, FS decreased the performance of the classifiers.

With regards to FS runtime, the Gini FS method was the most efficient with an average runtime of less than a second making it the recommended FS method when minimizing runtime is the objective. However, when the goal is to improve classifier performance, none of the studied FS methods gives the best performance improvement for all classifiers and datasets. Hence, from the results of the experiments, we proposed guidelines for selecting FS methods for binary and multiclass classification for the four classification algorithms studied given the number of features, instances, and class balance.

Although FS on multiclass classification on the average led to a degradation in classifier performance, we observed that CFS and CMIM methods gave the best result in that they yielded the least degradation in accuracy and led to improvement in some cases.

6.1 Future Work

FS can be applied to supervised and unsupervised tasks which implies that this work can be extended to regression and clustering tasks. It can also be extended to study other dataset factors with more bins using multiple repositories to facilitate finding representative datasets. For the benchmarking of FS methods, there is a lack of standardization and this poses a challenge for comparing already existing benchmarks of FS methods. Having a standard FS benchmark will be useful for expanding previous studies which at the

moment are not unified as there are numerous defined benchmarks. New FS methods are still being proposed and since the existing methods do not perform well for multiclass classification, proposing methods that close this gap will be highly beneficial. Lastly, several existing methods have polynomial runtimes and do not scale efficiently, more efficient implementation of these methods aiming for linear or sub-linear time complexity is highly needed especially with Big Data being ubiquitous.

Bibliography

[1] H. Hassan, A. Negm, M. Zahran, and O. Saavedra, "Assessment of artificial neural network for bathymetry estimation using high resolution satellite imagery in shallow lakes: Case study el burullus lake.," *International Water Technology Journal*, vol. 5, Dec. 2015.

[2] G. Bontempi, *Statistical foundations of machine learning (2nd edition) handbook*. Feb. 2021.

[3] V. Kumar and S. Minz, "Feature selection: a literature review," *SmartCR*, vol. 4, no. 3, pp. 211–229, 2014.

[4] B. Venkatesh and J. Anuradha, "A review of feature selection and its methods," *Cybernetics and Information Technologies*, vol. 19, no. 1, pp. 3–26, 2019.

[5] M. Dash and H. Liu, "Feature selection for classification," *Intelligent data analysis*, vol. 1, no. 1-4, pp. 131–156, 1997.

[6] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.

[7] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pp. 1200–1205, IEEE, 2015.

[8] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Recent advances and emerging challenges of feature selection in the context of big data," *Knowledge-based systems*, vol. 86, pp. 33–45, 2015.

[9] J. Li and H. Liu, "Challenges of feature selection for big data analytics," *IEEE Intelligent Systems*, vol. 32, no. 2, pp. 9–15, 2017.

[10] D. Peteiro-Barral, V. Bolon-Canedo, A. Alonso-Betanzos, B. Guijarro-Berdinas, and N. Sanchez-Maroono, "Scalability analysis of filter-based methods for feature selection," *Advances in Smart Systems Research*, vol. 2, no. 1, p. 21, 2012.

[11] C. Reggiani, Y.-A. Le Borgne, and G. Bontempi, "Feature selection in high-dimensional dataset using mapreduce," in *Benelux Conference on Artificial Intelligence*, pp. 101–115, Springer, 2017.

[12] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Distributed feature selection: An application to microarray data classification," *Applied Soft Computing*, vol. 30, pp. 136–150, 2015.

[13] C. M. Van der Walt, *Data measures that characterise classification problems*. PhD thesis, University of Pretoria, 2008.

[14] D. Oreski, S. Oreski, and B. Klicek, "Effects of dataset characteristics on the performance of feature selection techniques," *Applied Soft Computing*, vol. 52, pp. 109–119, 2017.

[15] J. Li and H. Liu, "Challenges of feature selection for big data analytics," *IEEE Intelligent Systems*, vol. 32, no. 2, pp. 9–15, 2017.

[16] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Computational Statistics & Data Analysis*, vol. 143, p. 106839, 2020.

[17] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowledge and information systems*, vol. 34, no. 3, pp. 483–519, 2013.

[18] M. Vidal-Naquet and S. Ullman, "Object recognition with informative features and linear classification.," in *International Conference on Computer Vision*, vol. 3, p. 281, 2003.

[19] F. Fleuret, "Fast binary feature selection with conditional mutual information.," *Journal of Machine learning research*, vol. 5, no. 9, 2004.

[20] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Machine learning proceedings*, pp. 249–256, Elsevier, 1992.

[21] I. Kononenko, "Estimating attributes: Analysis and extensions of relief," in *European conference on machine learning*, pp. 171–182, Springer, 1994.

[22] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1, pp. 23–69, 2003.

[23] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proceedings of the 24th international conference on Machine learning*, pp. 1151–1157, 2007.

[24] Y. Huang and L. Li, "Naive bayes classification algorithm based on small sample set," in *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pp. 34–39, 2011.

[25] A. Mucherino, P. J. Papajorgji, and P. M. Pardalos, *k-Nearest Neighbor Classification*, pp. 83–106. New York, NY: Springer New York, 2009.

[26] 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.

[27] J. Brownlee, *Master Machine Learning Algorithms: discover how they work and implement them from scratch.* Machine Learning Mastery, 2016.

[28] P. D. Wasserman and T. Schwartz, "Neural networks. II. what are they and why is everybody so interested in them now?," *IEEE expert*, vol. 3, no. 1, pp. 10–15, 1988.

[29] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling.* John Wiley & Sons, 1990.

[30] H. J. Weerts, A. C. Mueller, and J. Vanschoren, "Importance of tuning hyperparameters of machine learning algorithms," *arXiv preprint arXiv:2007.07588*, 2020.

[31] R. J. Urbanowicz, M. Meeker, W. La Cava, R. S. Olson, and J. H. Moore, "Relief-based feature selection: Introduction and review," *Journal of biomedical informatics*, vol. 85, pp. 189–203, 2018.

[32] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[33] M. A. Hall, "Correlation-based feature selection for machine learning," 1999.

Appendix A

Theoritical-Actual Runtime Comparison

The comparison of expected runtime of the FS methods to the actual runtime derived from experimenting was done for all studied FS methods. In Figure A.1, the results of the comparison not reported in Section 5.3 are presented. The conclusion is the same i.e., the actuals results follow the expected runtime.

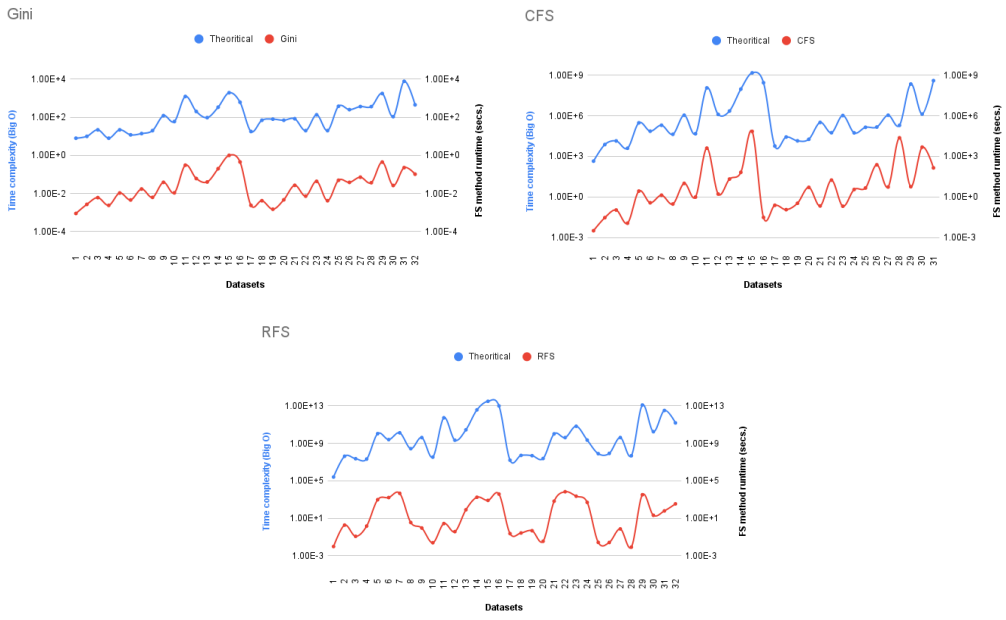


Figure A.1: Theoritical-actual runtime comparison for Gini, CFS, and RFS FS methods.

Appendix B

Decision Trees for Proposed Guidelines

To propose guidelines for choosing FS methods, decision trees maximizing accuracy change after FS were built for each classifier. Figure B.1 shows the decision trees for KNN, LDA, and NB classifiers.

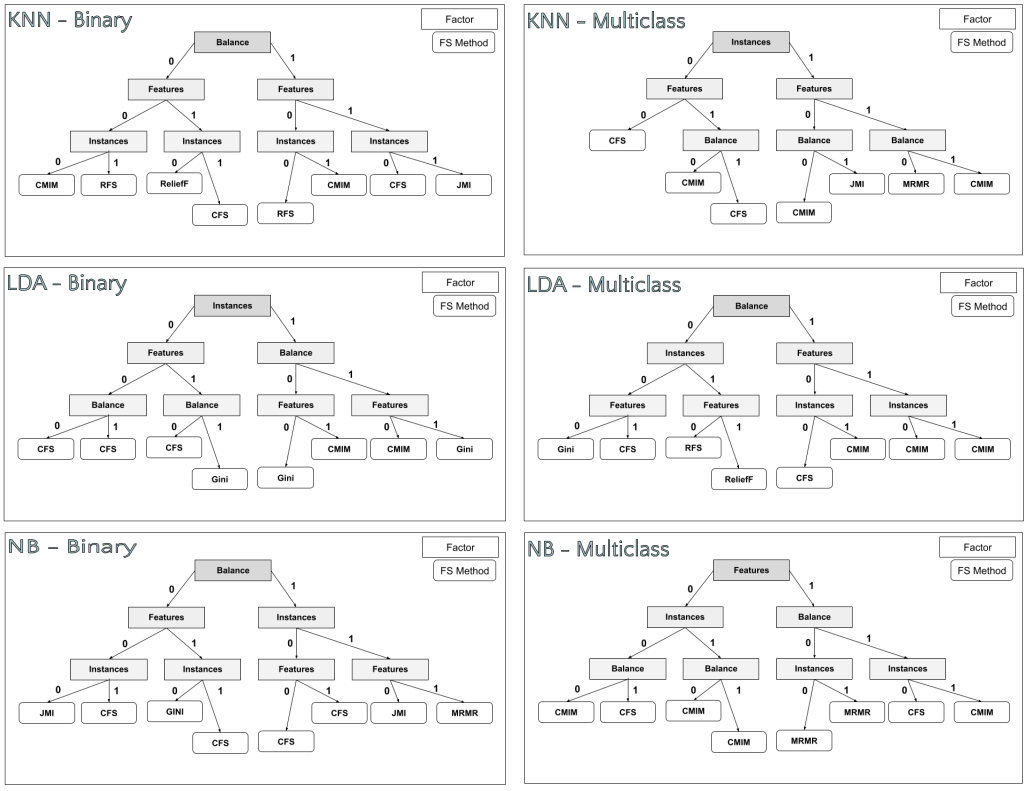


Figure B.1: Decision Trees for binary and multiclass cases.

Appendix C

Training Time Improvement

Applying FS to a dataset reduces the time it takes to train a model on it. Figure C.1 shows the average training time improvement for NB and MLP classifiers after applying the eight FS methods studied on 32 datasets.

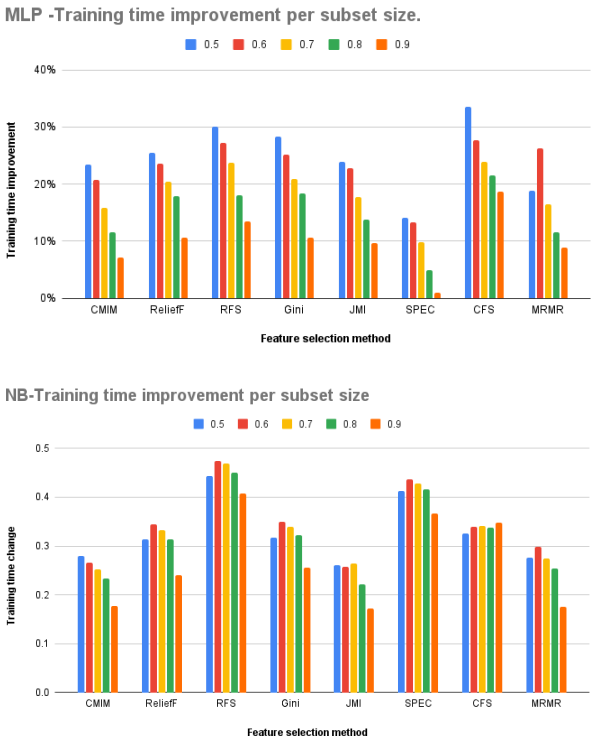


Figure C.1: Training time improvement for NB and MLP classifiers.

Appendix D

Accuracy Change

The accuracy of classifiers change depending on the classification task and FS methods used. Figures D.1 and D.2 show the average change in accuracy for NB, KNN, and MLP classifiers after applying the eight FS methods studied on 32 datasets.

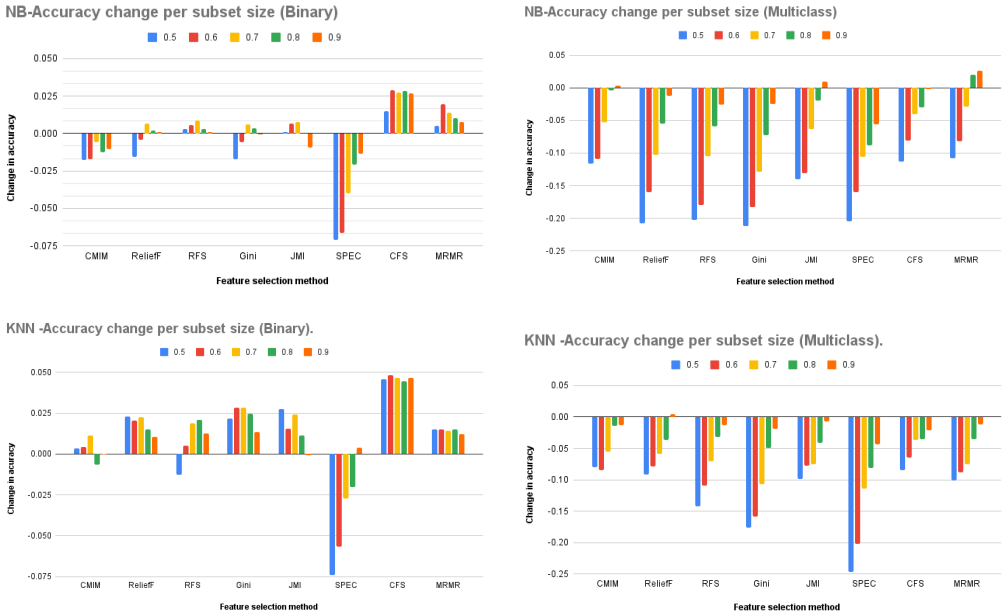


Figure D.1: Accuracy change for NB and KNN classifiers.

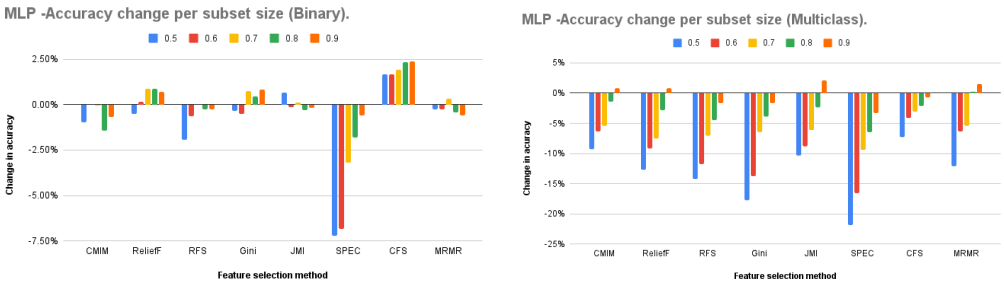


Figure D.2: Accuracy change for MLP classifier.

Appendix E

Scalability

In addition to the experiments reported in Section 5.3, the scalability of CMIM, Gini, JMI, MRMR, ReliefF and SPEC were tested on a synthetic dataset which had 1,000 features and 10,000 instances. The number of features and instances were fixed at 1,000 and 10,000 for the feature and instance scalability tests respectively.

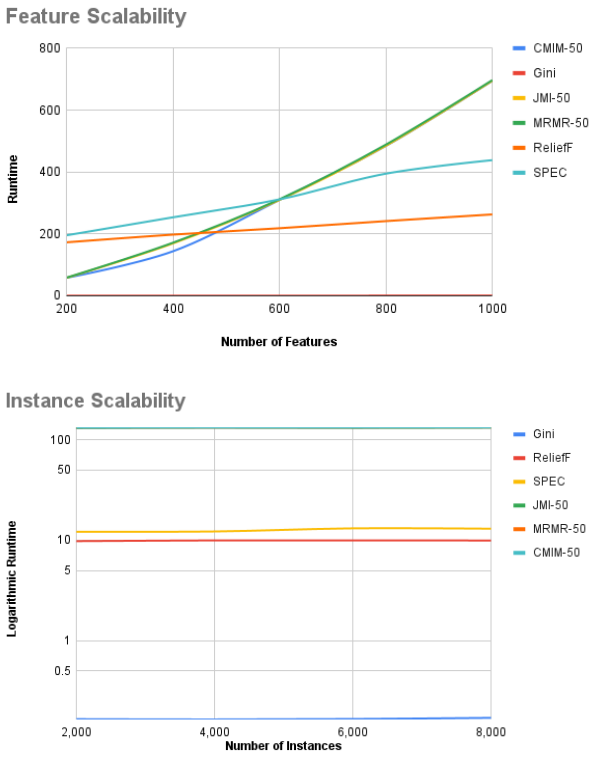


Figure E.1: Feature and instance scalabilty of FS methods.