CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Feature Selection for Microarray Data via Stochastic Approximation

Master's thesis in Computer science and engineering

Erik Rosvall

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Feature Selection for Microarray Data via Stochastic Approximation

Erik Rosvall

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Feature Selection for Microarray Data via Stochastic Approximation
Erik Rosvall

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2024

Feature Selection for Microarray Data via Stochastic Approximation
Erik Rosvall
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis explores the challenge of feature selection (FS) in machine learning, which involves reducing the dimensionality of data. The selection of a relevant subset of features from a larger pool has demonstrated its effectiveness in enhancing the performance of various machine learning algorithms. By reducing noise, improving model interpretability, and minimizing computational costs, FS plays a crucial role in optimizing algorithm outcomes.

This research specifically focuses on FS in the domain of machine learning for microarray data, which frequently involves large and high-dimensional datasets. Microarray data is widely utilized in biological research and holds significant value. While filter-based methods, which employ statistical properties to rank features, are commonly used to address this challenge, they often overlook the connections with the classification algorithm, resulting in suboptimal classification accuracy.

To address this limitation, this study analyses the performance of a novel wrapper-based feature selection approach known as SPFSR, as proposed in Akman et al. (2022) [1]. Unlike filter-based methods, SPFSR considers classification accuracy and demonstrates its capability to handle large datasets. By incorporating the classification algorithm in the feature selection process, this approach aims to improve the overall performance and effectiveness of machine learning models in microarray data analysis.

Keywords: Feature selection, feature ranking, microarray data, stochastic approximation, Barzilai and Borwein method, Machine Learning, AI.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Milad Malekipirbazari, for his invaluable guidance, unwavering support, and exceptional mentorship throughout the course of this project. His expertise, encouragement, and dedication have been instrumental in shaping the direction and success of this work.

I am also deeply thankful to Dr. David Akman for his generous assistance with the code, which significantly expedited the development process and enhanced the quality of the results. His expertise and willingness to help were crucial to overcoming various technical challenges.

I extend my appreciation to Jakob Sjölén for his creative input and contributions to the design of the PCR model.

This project would not have been possible without the collective effort, expertise, and encouragement of these individuals, and I am sincerely grateful for their contributions.

Erik Rosvall, Gothenburg, 2024-02-01

Contents

List of Figures

List of Tables

1

Introduction

The first chapter of this thesis provides an overview of the microarray experimental setup and the problem of feature selection in the context of microarray data analysis. Feature selection is an important preprocessing step that aims to identify a subset of relevant features from a large set of potential candidates. In the context of microarray data analysis, this task is particularly challenging due to the high dimensionality, sparsity, and noise inherent in the data. This chapter begins by stating the motivation of this study, the details of the microarray experimental setup, and introducing the concept of feature selection. We then describe a novel feature selection algorithm, called SPFSR, which is based on a stochastic approximation framework. Finally, we review the related literature on feature selection for microarray data and outline the contribution of this thesis to this field.

1.1 Motivation

This section intends to give a broad overview of the problems and obstacles faced in microarray data analysis, with a specific emphasis on high data dimensionality. It also emphasizes the need of using effective feature selection methods as a solution to these problems. These methods help enhance interpretability and performance of downstream studies by reducing data dimensionality and selecting the most relevant genes.

Over the last two decades, microarray technology has revolutionized the investigation of gene expression levels in diverse biological systems [2]. The examination of microarray data, on the other hand, presents substantial obstacles. The existence of noise, technological variability, and batch effects, which can produce false positives and false negatives and weaken the reliability and reproducibility of the results, is one of the key issues [2]. Another important issue, which is the focus of this thesis, is the data's high dimensionality, in which the number of genes on the microarray vastly outnumbers the number of samples. Overfitting and poor performance of machine learning models can result from this imbalance [3]. Addressing this challenge is critical to ensuring accurate interpretation and useful insights from microarray data analysis.

To address the challenges associated with microarray data analysis and to identify the most relevant genes, a key focus is on employing effective feature selection methods. Please note that in this thesis, the terms *gene expression* and *feature* are used

interchangeably. These methods play a critical role in reducing the dimensionality of the data and identifying the most relevant genes for further analysis. Specifically, the focus is on investigating and evaluating the performance of a novel wrapper-based feature selection approach called SPFSR, proposed in Akman et al. (2022) [1]. This method takes into account classification accuracy and is designed to handle large datasets. By selecting the most relevant genes using SPFSR, this project aims to enhance the identification of gene patterns and relationships and enhance the interpretability and performance of subsequent analyses, aiding in the understanding of underlying molecular mechanisms of diseases and facilitating the development of new therapies and drugs [4].

1.2 Microarray Data

Microarray technology is a powerful tool for analysing gene expression in various biological samples, including tissues, cells, and bodily fluids. It enables researchers to investigate the expression levels of thousands of genes simultaneously and identify molecular signatures associated with various biological conditions. Microarray technology has revolutionized the field of genomics, providing a cost-effective and high-throughput method for gene expression profiling. The development of microarray technology has led to many breakthroughs in basic and clinical research, including the discovery of new biomarkers for disease diagnosis and the development of targeted therapies for cancer [5], [6].

Microarray technology involves printing thousands of small spots of DNA or RNA probes onto a glass slide or a microchip. These probes are designed to hybridize with specific messenger RNA (mRNA) molecules, which are extracted from the biological sample being analysed. The mRNA molecules are fluorescently labelled and hybridized to the probes on the microarray, allowing the expression levels of thousands of genes to be measured simultaneously. This process generates a large amount of data, which requires sophisticated computational tools for analysis and interpretation. The data generated by microarray experiments can be preprocessed to remove noise, normalize the data, and correct for systematic biases [7], [8].

Microarray data can be analyzed using various statistical and machine-learning techniques to identify differentially expressed genes, gene regulatory networks, and other patterns of gene expression associated with various biological conditions. These analyses can provide valuable insights into the molecular mechanisms underlying complex biological processes, such as disease development and drug response. Statistical techniques such as t-tests, ANOVA, and regression analysis can be used to identify genes that are significantly differentially expressed between two or more conditions. Machine learning techniques such as clustering, classification, and FS can be used to identify patterns of gene expression associated with different biological conditions [9]–[11].

Microarray data has been widely used in various domains, including cancer research, drug development, and personalized medicine. In cancer research, microarray data has been used to identify gene expression signatures associated with disease progres-

sion, prognosis, and treatment response [12]. For instance, Cappuzzo et al. (2010) used microarray data to identify a gene expression signature associated with response to specific cancer therapy in patients with non-small cell lung cancer. The identified signature was then validated in an independent cohort of patients, demonstrating the potential of microarray data for identifying biomarkers of treatment response [13].

Despite its many advantages, microarray technology also has several limitations. One major limitation is the potential for cross-hybridization, where non-specific binding between probes and non-targeted molecules can lead to false-positive results. Microarray experiments can be expensive and time-consuming in some instances. Furthermore, in recent years, RNA sequencing (RNA-seq) has emerged as a viable alternative to microarray technology [14], [15].

In summary, microarray technology has revolutionized the field of genomics and provided unprecedented insights into the complex molecular mechanisms underlying various biological processes. Despite its limitations, it remains an important tool for researchers in many domains, particularly for the analysis of archived samples and for identifying candidate genes for further study. However, as newer technologies such as RNA-seq continue to improve and become more accessible, the use of microarray technology may continue to decline [16], [17].

1.2.1 Microarray Experimental Setup

Biologists can create gene expression profiles that differentiate the functionality of each gene in the genome based on the transcription levels of genes measured under different biological conditions. These profiles are organized into a matrix called the gene expression matrix, which biologists use to find new targets in their experimental data. Microarrays can range in size from microscope slides to square silicon chips, with each area carrying a large number of identical DNA molecules. Sample collection, RNA extraction, amplification, labeling, hybridization, and data processing are all critical steps in creating a microarray dataset. This permits single-stranded DNA molecules to be fixedly bound to a solid substrate with a grid of dots on which genetic material from known sequences is structured in a logical manner. Microarray data analysis strives to answer concerns concerning gene functions, regulation, expression, and their significance in cellular processes and disorders through effective analysis and display of the data [18]. These steps that are performed to generate a microarray dataset are shown in Figure 1.1 and are explained in more detail in Appendix A.

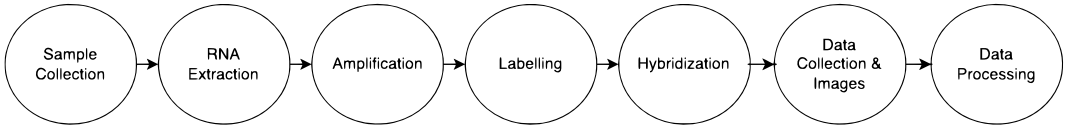

Figure 1.1: Steps involved for conducting microarray experiment

1.3 Feature Selection

Feature selection (FS) is a critical step in machine learning and data mining, where the aim is to identify the most relevant subset of features from a larger set of features in a dataset. This process is essential to improve model accuracy and efficiency, reduce overfitting, and simplify interpretation. The impact of FS extends far beyond the field of microarray data analysis and is increasingly recognized as a critical step in many areas of research and industry. To illustrate the wide-ranging applications of FS, we will discuss several important domains that have benefited from its use, including image recognition, text classification, bioinformatics, and finance [19], [20].

In bioinformatics, FS is used to identify relevant genes for disease diagnosis and prognosis. In finance, it is used to identify relevant financial ratios for predicting bankruptcy. In image recognition and text classification, FS is used to identify the most informative visual or textual features that can distinguish between different categories [21]. In natural language processing, FS is used to identify the most informative linguistic features that can distinguish between different text categories. This can include features such as word frequency, part-of-speech tags, and semantic information. This has important implications for text classification tasks such as sentiment analysis, topic modelling, and named entity recognition, where accurate and efficient classification depends on the selection of relevant features [21], [22].

Moreover, to better understand the methods used for feature selection, it is helpful to first define and contrast the main approaches: filters, wrappers, embedded methods, and hybrids. In the following paragraphs, we will describe the main characteristics and limitations of each of these methods, beginning with filters.

Filter methods are one of the most popular FS approaches that rely on statistical techniques to assess the intrinsic properties of the data and identify the most important features. These methods use a scoring function to rank features based on their individual relevance to the target variable, without requiring a machine learning model. The most common scoring functions used in filter methods are correlation-based measures, mutual information, and chi-squared test [23] (see Section 1.4 for more in-depth description).

Wrapper methods are another type of FS technique that uses a machine learning model to evaluate subsets of features and select the optimal subset based on model performance. These methods select a subset of features, train a model on that subset, and evaluate the model on a validation set. This process is repeated for all or some of the possible subsets of features, and the subset that yields the best performance is selected as the final set of features. Wrapper methods are computationally expensive, but they can identify highly informative features that may be missed by filter methods [24], [25] (see Section 1.5 for further description).

Embedded methods combine FS with the model training process, selecting the most informative features during model training. These methods are commonly used in linear models and decision trees, where the importance of features can be measured

directly. In embedded methods, FS is integrated into the model-building process, so the model can learn which features are most relevant to the target variable [19], [26].

Hybrid feature selection methods, on the other hand, combine the strengths of both filter and wrapper approaches by using filter methods in combination with other FS methods, such as wrapper methods. As discussed earlier, wrapper methods use machine learning models to evaluate the importance of features by considering the interactions between them and evaluating the performance of a model trained on different subsets of features. In contrast, filter methods select features based on their intrinsic properties, such as their statistical measures of relevance or redundancy. By combining filter and wrapper methods, hybrid approaches can improve the efficiency of filter methods while leveraging the effectiveness of wrapper methods in handling feature interactions. Brownlee (2019) provides an overview of hybrid FS methods, highlighting their potential benefits in handling both real and categorical data. For instance, these methods can reduce the computational cost of wrapper methods while still achieving high prediction accuracy [27]. Liu and Motoda (1998) discuss FS for knowledge discovery and data mining, including the use of filter and wrapper methods, as well as their combinations [20].

While we have briefly touched on the two main approaches to feature selection - filters and wrappers - these methods are fundamental to our discussion and will be described in more detail in subsequent sections.

1.4 Filters

Filter methods are a popular approach for FS that uses statistical techniques to identify the most important features in a dataset. These methods do not rely on a machine learning model but instead evaluate the intrinsic properties of the data to determine feature relevance. Filter methods are particularly useful when the number of features in the dataset is high, as they can help reduce dimensionality and improve the accuracy and efficiency of machine learning models [19], [20].

Filter methods typically use a scoring function to rank the features based on their individual relevance to the target variable. Features with high scores are considered more relevant and are retained, while features with low scores are discarded. Common scoring functions used in filter methods include correlation coefficients, mutual information, and statistical tests such as t-tests and ANOVA [19], [28]–[30].

Filter methods have several advantages over other FS methods, including their simplicity and computational efficiency. They are also less prone to overfitting than wrapper methods, as they do not rely on a machine learning model. However, they may not always select the optimal subset of features, as they do not consider the interactions between features [19], [23], [31].

One common type of filter method is the chi-squared test, which is used to evaluate the independence of categorical variables. The chi-squared test is used to determine whether there is a significant relationship between two variables, and is often used

in FS for text classification tasks [32].

There are many variations of filter methods, including univariate, multivariate, and hybrid approaches. In univariate filter methods, each feature is evaluated independently of the others, while in multivariate filter methods, the interactions between features are taken into account. Hybrid methods combine filter methods with other FS techniques, such as wrapper methods, to overcome the limitations of each approach. For instance, Liu et al. (2015) used a hybrid method to select features for predicting heart disease. The study used a filter method to pre-select the most informative features and then used a wrapper method to select the final subset of features [20], [27], [33].

In conclusion, filter methods are a popular approach for FS that uses statistical techniques to rank the importance of features based on their intrinsic properties. They are simple and computationally efficient, making them an attractive option for large datasets. However, they may not always select the optimal subset of features, as they do not consider the interactions between features. Nonetheless, filter methods have been successfully used in many applications and continue to be an important tool in FS [19], [34]–[36].

1.5 Wrappers

Wrapper methods are another popular approach for FS that uses machine learning models to evaluate the importance of features. Unlike filter methods, wrapper methods consider the interactions between features by evaluating the performance of machine learning models trained on different subsets of features [19], [20], [24], [25].

Wrapper methods use a search algorithm to explore the space of possible feature subsets and select the subset that maximizes the performance of the machine learning model. One common search algorithm used in wrapper methods is the recursive feature elimination (RFE) algorithm, which iteratively removes the least important feature from the subset until the desired number of features is reached [37].

Wrapper methods can be computationally expensive, as they require training and evaluating multiple machine learning models on different subsets of features. However, they can often select a better subset of features rather than filter methods, as they consider the interactions between features and the performance of the machine learning model [38].

Somon et al. (2005) compared the effectiveness of filter- versus wrapper-based FS methods for credit scoring, demonstrating the relevance and potential applications of wrapper methods in the finance industry [39]. Another study by Afolabi et al. (2020) used a wrapper method to select financial ratios for predicting financial distress in the finance industry [40]. In image analysis, wrapper methods have been used to select the most informative features for image segmentation and classification, as shown in a review by Zhang et al. (2018) [41].

In conclusion, wrapper methods are a powerful approach for FS that consider the

interactions between features and the performance of a machine learning model. They can often select a better subset of features rather than filter methods but at the cost of increased computational complexity. Wrapper methods have been successfully used in many applications and continue to be an important tool in FS [19], [24], [42].

1.6 SPFSR

As discussed, FS is a crucial step in machine learning pipelines that aims to reduce the dimensionality of datasets by identifying the most informative and relevant features, and a range of filter and wrapper methods have been proposed to achieve this goal. In this work, we focus on SPFSR, a wrapper-based FS technique that has demonstrated promising results in various applications [43]. Unlike univariate filter methods, SPFSR takes into account the interactions between features and considers the performance of a given model on a subset of features to evaluate their relevance. Moreover, as a flexible and generalizable approach, SPFSR can be used with a range of classifiers or regressors of choice. In what follows, we describe the main elements of the SPFSR method, including the Stochastic Perturbation Stochastic Approximation (SPSA) algorithm, the Barzilai-Borwein (BB) approach, and the feature selection procedure. We also discuss the advantages and limitations of this approach. We begin with an overview of the SPSA algorithm, which serves as the optimization building block for SPFSR.

Simultaneous Perturbation Stochastic Approximation (SPSA) is a multivariate optimization algorithm developed by J.C. Spall (1992) [44]. This optimization procedure makes use of gradient approximation, and in order to approximate the solution, the components of the solution vector are subjected to random perturbations produced by a precisely defined probability distribution. This strategy of simultaneously modifying the components of the solution vector results in a gradient approximation [45]. As an alternative, we can indirectly calculate the Hessian matrix [1]. The secant equation serves as the basis of quasi-Newton techniques, and the Barzilai-Borwein (BB) approach, developed by Barzilai and Borwein in 1988, offers an estimate of the Hessian matrix using a two-point approximation of the secant equation. The BB methodology generates a succession of step size values, which may not decrease in a predictable way, in contrast to other approaches that directly calculate the Hessian matrix. A series of step size numbers that are not necessarily monotonically decreasing are produced by this approximation [43].

To further elaborate on the SPFSR technique, it is important to note that the method for FS was proposed in two recent studies of [43] and [1] by Akman et al. In this thesis, we will discuss these two papers that proposed the SPFSR method and used it for different applications. The first paper [43] introduced the BB gain sequence with gradient averaging and gain smoothing into the SPSA algorithm for feature selection tasks. The second paper [1] proposed the SPFSR technique as a constrained optimization problem for feature selection and ranking that seeks to identify and rank the k best features. They evaluated the performance of SPFSR against other feature selection methods. In the following sections, we will provide

a detailed discussion of these papers and their contributions to the development of the SPFSR method.

1.6.1 Feature selection using stochastic approximation with Barzilai and Borwein non-monotone gains [43]

This paper proposes a new method for feature selection using stochastic approximation. The authors suggest a non-monotone stochastic approximation algorithm that is based on Barzilai and Borwein gains to solve the optimization problem that arises in feature selection.

The proposed method selects the best subset of features by optimizing an objective function that balances the accuracy and complexity of the feature subset. The objective function is a weighted sum of two terms: the first term is the classification accuracy of the feature subset, and the second term is a measure of the complexity of the feature subset. The authors use a stochastic gradient ascent algorithm to optimize this objective function.

The novelty of this method lies in the use of the Barzilai and Borwein gains, which enable the algorithm to converge faster to the optimal solution compared to other stochastic approximation methods. Moreover, the non-monotonicity of the gains allows the algorithm to escape local optima and explore the search space more efficiently.

The authors validate the proposed method on several benchmark datasets from the UCI machine learning repository and compare it with several state-of-the-art feature selection methods. The experimental results show that the proposed method achieves better classification accuracy and selects smaller feature subsets compared to other methods.

1.6.2 k-best feature selection and ranking via stochastic approximation [1]

The k-best feature selection method is a popular approach to selecting the most informative features from a high-dimensional dataset. This method involves ranking the features based on their performance in a classification or regression model, and selecting the k-best features with the highest ranking scores.

Stochastic approximation is a method used for the iterative optimization of a function. In the context of feature selection, the stochastic approximation can be used to iteratively improve the ranking of the features based on their contribution to the classification or regression model.

The k-best feature and ranking via the stochastic approximation method starts with an initial feature ranking based on their individual performance in a classification or regression model. Then, the algorithm iteratively updates the feature rankings using stochastic approximation with the non-monotone gain function. At each iteration, the algorithm selects the k-best features based on the updated rankings and trains

a classification or regression model using only these features. The performance of the model is evaluated, and the process is repeated until the algorithm converges to a stable set of k-best features.

In summary, the k-best feature and ranking via the stochastic approximation method is a powerful approach to feature selection that combines the ranking of individual features with iterative optimization using stochastic approximation. This method can be applied to a wide range of classification and regression problems and has been shown to outperform other popular feature selection methods in some cases.

1.7 Related Works

In this section, we review and compare various feature selection strategies that are routinely employed in high-dimensional datasets for cancer classification, such as microarray gene expression data. Due to the increasing availability of high-throughput gene expression data, there has been substantial attention on microarray data analysis in biomedical informatics. One of the main challenges is identifying significant features within these datasets, which is crucial for understanding biological pathways and developing diagnostic and prognostic tools for diseases such as cancer. To address this difficulty, several feature selection methods have been developed and proposed in the literature.

Table 1.1 summarizes the various FS methods discussed in this section. The table also includes the acronyms for each method, the types of FS, and the number of times each method was used.

To summarize, feature selection is an important preprocessing step for microarray data analysis, and several approaches have been developed to solve this issue. Wrapper and filter methods have been combined, and hybrid approaches have been shown to improve classification accuracy while reducing the number of selected genes. Additionally, ensemble approaches that incorporate several filter-based procedures have yielded encouraging results. Furthermore, distributed computing frameworks have been used to speed up the computation of feature selection. These approaches have contributed significantly to the identification of important genes in cancer diagnostics and other microarray-based applications.

Table 1.2: Details of the related literature.

Article	**Type**	**Methods**	**Data**
Nuklianggraita et al. (2020) [46]	Filter	RFE PCA	The Colon Tumor dataset from the Gene Expression Omnibus (GEO) The Leukemia dataset from the University of California, San Francisco (UCSF)

Continued on next page

Table 1.2 – continued from previous page

Article	Type	Methods	Data
Bolón-Canedo et al. (2014) [47]	Filter	CFS PCA MIFS	Acute lymphoblastic leukaemia (ALL) Acute myeloid leukaemia (AML) patients Colon cancer dataset
	Wrapper	RFE RF-FS	Prostate cancer Breast cancer Lung cancer Glioma Lymphoma
Zhu et al. (2007) [48]	Filter	ReliefF mRMR CFS JMI SFS GA	Colon Cancer (Alon) Prostate Cancer (Singh) Lung Cancer (Bhattacharjee)
Meyer et al. (2008) [49]	Filter	IG SU MRMR mRMR	*not specified*
Apolloni et al. (2016) [50]	Hybrid	Two methods	Kent Ridge Biomedical Dataset Repository
Pirooznia et al. (2008) [51]	Filter	Chi Squared CFS	*not specified*
	Wrapper	SVM-RFE	
Chuang et al. (2011) [52]	Hybrid	RPSO + SVM	ALLAML (subtypes: Acute lymphoblastic leukemia (ALL) Acute myeloid leukemia (AML))
Ghosh et al. (2019) [53]	Filter	t-Test Fisher Score Gini Index ReliefF	GCM CLL/SLL
Bolón-Canedo et al. (2015) [54]	Filter	DF DRF	Colon DLBCL CNS Leukemia Prostate Lung Ovarian Breast
Wang et al. (2005) [55]	Filter	IG ReliefF GA	*no specifics specified* (Stanford Microarray Database (SMD) and the National Cancer Institute (NCI))
Liu et al. (2012) [20]	Filter	CFS PCA MIFS GA	*no specifics specified* (Stanford Microarray Database (SMD) and the National Cancer Institute (NCI))

Table 1.1: Count of FS methods used in the literature.

Method	**Acronym**	**Type**	**Count**
Correlation-Based Feature Selection	CFS	Filter/ Wrapper/ Embedded	4
Relief-Based Feature Selection	ReliefF	Filter/ Wrapper	3
Genetic Algorithm	GA	Filter/ Wrapper/ Embedded	3
Information Gain	IG	Filter/ Embedded	3
Principal Component Analysis	PCA	Filter	3
Recursive Feature Elimination algorithm	RFE	Filter	2
Minimum Redundancy Maximum Relevance	mRMR	Filter/ Wrapper	2
Mutual Information-Based Feature Selection	MIFS	Filter/ Embedded	1
Random Forest-Based Feature Selection	RF-FS	Hybrid	1
Joint Mutual Information	JMI	Filter/ Wrapper	1
Symmetrical Uncertainty	SU	Filter/ Embedded	1
Gini Index	—	Filter	1
Distributed Filter	DF	Hybrid/ Embedded	1
Distributed Ranking Filter	DRF	Hybrid/ Embedded	1
Random Particle Swarm Optimization	RPSO	Hybrid	1
Fisher Score	—	Filter	1
t-Test	—	Filter	1
Sequential Forward Selection	SFS	Wrapper	1
Chi Squared	—	Filter	1
SVM recursive feature elimination	SVM-RFE	Wrapper	1

The main strategy for FS, as demonstrated in Table 1.2, is a filter or a filter in conjunction with a wrapper (hybrid). As remarked, filter methods are based on statistical measures or other criteria, and each feature is evaluated independently of the others. They are computationally efficient and widely employed. Correlation-based feature selection (CFS), t-test, Fisher score, information gain (IG), ReliefF, and Gini index are the most commonly utilized filter methods in the studies.

Wrapper approaches analyze subsets of features and measure their classification performance using a machine learning algorithm. They are computationally costly, yet they have the ability to locate the best feature subset. Recursive feature elimination (RFE), sequential forward selection (SFS), and genetic algorithm (GA) are examples of wrapper approaches employed in the investigations.

Embedded approaches embed feature selection within the learning algorithm, eliminating the need for a separate feature selection phase. Decision tree (DT), random forest (RF), support vector machine (SVM), and logistic regression (LR) are examples of embedding approaches employed in the investigations.

Table 1.3: Count of unique FS approaches used in the literature.

FS Approach	Count
Filter	14
Embedded	7
Wrapper	5
Hybrid	4

Table 1.3 provides a summary of the different types of FS techniques employed in previous studies on microarray data. As indicated in the table, the number of studies that employed wrapper and embedded techniques are less frequent compared to the filter. This outcome is consistent with the assumption stated in Section 1.1 that earlier research on microarray data has generally utilized a filter or a combination of filter and wrapper techniques. Nevertheless, the results show that filter techniques have been more widely used than wrapper and embedded techniques in earlier attempts at FS for microarray data.

1.8 Contribution

Researchers face significant challenges due to the complexity and size of large-scale transcriptomics. In an era of rapidly expanding transcriptomics, feature selection approaches are viewed as an excellent solution because they can greatly reduce data complexity, allowing us to assess and convert the data into relevant information.

This work aims to investigate the efficacy of novel feature selection approaches and compare their performance to established methods in order to address the challenge of feature selection in transcriptomics. We will conduct experiments and evaluate the effectiveness of SPFSR, a novel feature selection approach, in microarray data classification. Our goal with this study is to compare this novel approach with different feature selection algorithms for transcriptomics, with a particular emphasis on evaluating the performance of the SPFSR algorithm. This method will be investigated and refined in order to maximize its efficacy in transcriptomics. We believe that our research will help to develop practical applications and new insights into gene expression data analysis, as well as improve gene expression analytics interpretation.

2

Theory

The goal of feature selection in machine learning is to identify a subset of relevant features that can improve the performance of a predictive model. This chapter provides a theoretical understanding of three important optimization algorithms: Simultaneous Perturbation Stochastic Approximation (SPSA), Barzilai and Borwein (BB), and Simultaneous Perturbation Stochastic Approximation for Feature Selection and Ranking (SPFSR). The SPSA and BB algorithms are well-known optimization techniques that have been used in a wide range of applications such as machine learning, signal processing, and control engineering. SPFSR is a novel feature selection method that extends the SPSA and BB algorithms in order to identify the best subset of features. We aim to provide a comprehensive understanding of the theory behind SPFSR and how it can be applied to solve feature selection problems by thoroughly examining these algorithms. By the end of this chapter, readers should have a firm grasp on the mathematical principles underlying each algorithm.

2.1 SPSA

SPSA is a powerful optimization algorithm that is used for problems with noisy, black-box objective functions. The algorithm uses random perturbations to estimate the gradient of the objective function and then performs a stochastic approximation update to find the optimal solution. To ensure the convergence of the algorithm, at each iteration k, the step size sequence α_k is typically chosen to satisfy the Robbins-Monro conditions, which ensure convergence of the algorithm. One common choice is:

$$\alpha_k = \frac{a_k}{(b_k + k)^\gamma}$$

where a_k, b_k, and γ are constants that depend on the problem.

Algorithm 1 explains the steps involved in the SPSA algorithm. Based on the steps outlined in Algorithm 1, SPSA is a stochastic optimization algorithm that aims to find the optimal solution vector x^* for a given objective function $f(x)$. At each iteration, the algorithm generates random perturbation vectors and uses them to estimate the gradient of the objective function. The solution vector is then updated using a stochastic approximation update. The process repeats until the termination criterion is satisfied. The output of the algorithm is the optimal solution vector x^*.

Algorithm 1 Simultaneous Perturbation Stochastic Approximation (SPSA)

Input: Objective function f, initial parameter values θ, step sizes a_n, c_n, number of iterations N, and gradient estimate parameter A_n

Output: Optimal parameter values $\hat{\theta}$

1 Initialize $k \leftarrow 0$, $\theta_k \leftarrow \theta$, $y_k \leftarrow f(\theta_k)$; **while** $k < N$ **do**

2 Generate random vector Δ_k where each component $\Delta_{k,i}$ is drawn independently from the Bernoulli distribution with parameter 0.5; Compute δ_k and $\tilde{\delta}k$ using the formulas:

$$\delta_k = a_k/(k+1+c_k)^{A_k} \;\; \tilde{\delta}k \qquad\qquad = -a_k/(k+1+c_k)^{A_k} \tag{2.1}$$

Compute the gradient estimate g_k using the formula:

3

$$gk, i = \frac{f(\theta_k + \Delta_k \delta_k e_i) - f(\theta_k + \Delta_k \tilde{\delta} k e_i)}{\Delta_k(\delta_k - \tilde{\delta}k)} \tag{2.2}$$

4 Compute the next iterate $\theta k + 1$ using the formula:

$$\theta k + 1 = \theta_k - a_k g_k \tag{2.3}$$

5 Set $k \leftarrow k + 1$, $y_k \leftarrow f(\theta_k)$;

6 **end**

7 **return** $\hat{\theta} \leftarrow \operatorname{argmin} \theta_k y_k$

2.2 Barzilai and Borwein

The Barzilai and Borwein (BB) algorithm is a powerful optimization algorithm that is particularly useful for non-quadratic or non-convex functions. To determine the step size in the BB algorithm, a non-monotone line search is used, which is designed to minimize a quadratic function. This function is defined as:

$$\phi(\alpha) = f(x - \alpha g) - f(x) - c\alpha g^T g$$

where $f(x)$ is the objective function, g is the estimated gradient of f at x, and c is a positive constant. The parameter c controls the trade-off between the step size and the change in the objective function. A larger value of c leads to smaller steps, while a smaller value leads to larger steps [43].

The step size is selected by solving the optimization problem:

$$\min_{\alpha > 0} \phi(\alpha)$$

This problem can be solved analytically by computing:

$$\alpha_k = \frac{|x_k - x_{k-1}|^2}{(x_k - x_{k-1})^T (g_k - g_{k-1})}$$

where $|.|$ denotes the Euclidean norm, x_k is the solution vector at iteration k, and g_k is the estimated gradient of f at x_k. This update rule ensures that the step size is inversely proportional to the dot product of the difference between the solution vectors and the difference between the gradient estimates. By selecting the appropriate step size, the BB algorithm can effectively escape local minima and find better solutions for non-quadratic or non-convex optimization problems. The algorithm iteratively updates the solution vector x by using the estimated gradient g and the non-monotone step size that takes into account the past few iterations of the algorithm. To update the solution vector x, we use the following update rule:

$$x_{k+1} = x_k - \beta_k g_k$$

where β_k is the step size at iteration k and g_k is the gradient estimate obtained from the perturbations. The step size β_k is computed as follows:

$$\beta_k = \gamma_k \frac{||g_k||}{||\Delta_k||}$$

where γ_k is a non-monotone gain sequence typically chosen to satisfy certain conditions, $||g_k||$ is the norm of the gradient estimate, and $||\Delta_k||$ is the norm of the perturbation vector [43].

Notice that the BB method can also be applied to non-linear optimization problems by approximating the Hessian with a suitable matrix. However, the convergence properties of the method are not as well understood in the non-linear case.

The step size β_k in the stochastic gradient descent algorithm is obtained from the gradient estimate gk and the non-monotone gain sequence. This gain sequence is chosen to satisfy certain conditions and is typically non-monotone, allowing the algorithm to escape local minima and find better solutions. The step size β_k is computed as follows [43]:

1. Initialize the algorithm by choosing an initial point x_0, a step size α_0, and a tolerance $\epsilon > 0$.
2. Compute the gradient g_k of the function f at the current point x_k.
3. Compute the next iterate x_{k+1} by the formula: $x_{k+1} = x_k - \frac{\alpha_k g_k}{|g_k|}$ where $|\cdot|$ is the Euclidean norm.
4. Compute the gradient estimate g_{k+1} as the average of $g_{k+1,1}, \ldots, g_{k+1,m}$ (gradient averaging)
5. Compute the step size α_{k+1} using the Barzilai-Borwein formula: $\alpha_{k+1} = \frac{|g_{k+1} - g_k|^2}{|(x_{k+1} - x_k)^T (g_{k+1} - g_k)|}$ where g_{k+1} is the gradient at the next iterate x_{k+1}.

6. If $|g_{k+1}| < \epsilon$, stop the algorithm and return x_{k+1} as the solution. Otherwise, set $k = k + 1$ and repeat from Step 2.

Algorithm 2 Barzilai-Borwein (BB) Algorithm

Procedure BB $f(x), x_0, t_0,$ stopping criterion Compute $\nabla f(x_0)$
Choose a search direction p_0 that is a descent direction of f at x_0
Compute step size α_0 using the Barzilai-Borwein formula:

$$\alpha_0 = t_0 \frac{|p_0|^2}{p_0^\top \nabla f(x_0)}$$

Update the solution vector: $x_1 = x_0 - \alpha_0 p_0$
Compute $\nabla f(x_1)$
while *termination criterion not satisfied* **do**
1 Choose a search direction p_k that is a descent direction of f at x_k
2 Compute step size α_k using the Barzilai-Borwein formula:

$$\alpha_k = t_0 \frac{|p_k|^2}{p_k^\top (\nabla f(x_k) - \nabla f(x_{k-1}))}$$

3 Update the solution vector: $x_{k+1} = x_k - \alpha_k p_k$
end
return $x^* = x_k$

The procedures for the Barzilai-Borwein algorithm are explained in Algorithm 2. The algorithm 2 has been shown to be effective in minimizing non-convex and ill-conditioned functions. The non-monotone gain sequence allows the algorithm to escape from local minima and find better solutions. Note that the choice of the initial point, the search directions, and the stopping criterion are problem-dependent and may require some trial and error. The BB algorithm is usually applied to unconstrained optimization problems and may require some modifications for constrained problems. The algorithm has been shown to have fast convergence rates and good numerical stability properties in practice [43].

2.3 SPFSR

The SPSFR algorithm is a powerful feature selection technique that aims to enhance a model's performance by identifying a subset of features from a larger set. To achieve this, the algorithm utilizes a stochastic approximation method, which involves selecting a random subset of features, assessing the model's performance using that subset, and then updating the importance weights of each feature based on the performance improvement obtained by adding or removing it. This iterative process continues until the algorithm converges or reaches a fixed number of iterations, and it ultimately returns the best subset of features based on the performance measure. The SPFSR algorithm is described in the Algorithm 3.

The algorithm updates the importance weight for each feature in the subset based on the performance improvement achieved by adding or removing that feature. Specifically, at each iteration t, the algorithm computes a weight vector of size k, where $w_{t,i}$ represents the importance weight of the ith feature in the subset S_t. Initially, all weights are set to $1/k$. By utilizing this approach, SPSFR, as described in Algorithm 3, effectively selects the most relevant features and discards the least significant ones, leading to improved model performance.

- If adding feature i to S_t improves performance, then $w_{t+1,i} = w_{t,i} \times (1 + \gamma)$, where γ is a small positive constant.
- If removing feature i from S_t improves performance, then $w_{t+1,i} = w_{t,i} \times (1 - \gamma)$.
- Otherwise, $w_{t+1,i} = w_{t,i}$.

Note that the sum of the weights is always equal to 1.

Once the weights are updated, the algorithm selects the new subset S_{t+1} by sampling k features from X with replacement, where each feature i is selected with probability proportional to its weight $w_{t+1,i}$. The algorithm repeats this process for a fixed number of iterations or until convergence and returns the best subset of features based on the performance measure.

Algorithm 3 SPFSR Algorithm

Input: $\hat{w}_1$, c, M, m, t, k, a_{min}, a_{max}

for $i = 1$ *to* M **do**

  for $j = 1$ *to* m **do**

    Simulate $r_{i,j}$ a Rademacher random vector

    $\hat{w}_{i,j}^{\pm} = \hat{w}_i \pm c r_{i,j}$ `// weight perturbation`

1 $\hat{w}_{i,j}^{\pm} = I(\hat{w}_{i,j}^{\pm})$ `//` $I(\bullet)$`: indicator vector of` k `largest elements`

2 $y_{i,j}^{\pm} = \mathcal{L}(\hat{w}_{i,j}^{\pm}) + \varepsilon_{i,j}^{\pm}$ `// (noisy) function measurement`

3 $\hat{g}_{i,j}(\hat{w}_i) = \left(\frac{y_{i,j}^{+} - y_{i,j}^{-}}{2c} \right) r_{i,j}$ `//` $\hat{g}_{i,j}(\hat{w}_i)$`: gradient estimate`

  $\hat{g}_i(\hat{w}_i) = \frac{1}{m} \sum_{j=1}^{m} \hat{g}_{i,j}(\hat{w}_i)$ `// gradient averaging`

4 $\hat{a}_i = \frac{\Delta \hat{w}^T \Delta \hat{w}}{\Delta \hat{w}^T \Delta \hat{g}(\hat{w})}$ `//` $\hat{a}_i$`: BB step size`

5 $\hat{a}_i = B(\hat{a}_i)$ `//` $B(\bullet)$`:` $[a_{min}, a_{max}]$ `bounding operator`

6 $\hat{a}_i = \frac{1}{t'} \sum_{n=i-t'+1}^{i} \hat{a}_n$, where $t' = \min\{t, i\}$ `// gain smoothing`

7 $\hat{w}_{i+1} = \hat{w}_i - \hat{a}_i \hat{g}_i(\hat{w}_i)$ `// weight updating`

Output: $I(\hat{w}_{M+1})$

3

Methods

In this chapter, we present data classification methodology based on three popular classifiers: Naive Bayes (NB), Decision Tree (DT), and Support Vector Machine (SVM). The purpose of this research is to provide a thorough comparison of these classifiers, evaluating their performance on a variety of datasets with varying characteristics. We also go over multiclass classification, which involves predicting an instance's class label from multiple classes. The chapter begins with an introduction to the classification and multiclass classification, and then goes on to explain these classifiers in detail, including their mathematical foundations. In addition, we describe the use of cross-validation, a common technique for evaluating classifier performance, and present the evaluation framework for this study. Finally, we summarize the datasets used in this study, which range in size, type, and complexity.

3.1 Workflow

The thesis involves the following steps: (1) data selection, (2) feature selection, (3) classification, and (4) evaluation. In the *data selection* step, we determine the data sets to be used in the study (see Table 3.1 for more details). Once the data sets are chosen, the next step is *feature selection*. The primary method used in this study is SPFSR, which is elaborated in Chapter 2.3. Additionally, there are several other feature selection methods available based on the related works, which are presented in Table 1.1. After selecting the FS methods, we proceed to choose a sample of classification algorithms for the *classification* stage. These classifiers will be described later in this chapter. Finally, we evaluate the results of the classifiers using cross-fold validation. Figure 3.1 displays this workflow in a sequential manner.

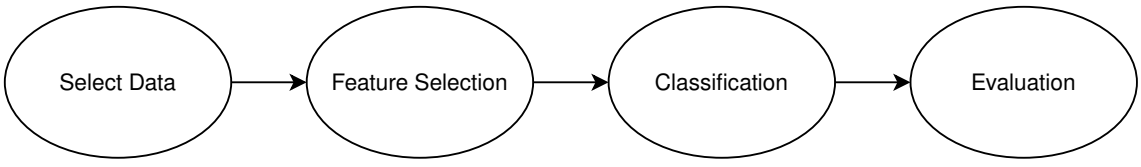

Figure 3.1: Workflow of the thesis illustrating the sequential steps.

3.2 Classification

In the field of machine learning, classification is a widely used supervised learning technique. Its objective is to build a function that maps input variables to output variables, where the output variable represents a class or category. Typically, input variables are a set of features or attributes that describe the objects or instances being classified. Various practical applications, such as image recognition, spam filtering, fraud detection, and medical diagnosis, utilize classification [42].

To achieve accurate and reliable classification results, data scientists must carefully prepare the data, perform feature engineering, and select appropriate classification algorithms. Skiena (2017) discusses the strengths and limitations of several popular classification algorithms, including decision trees, logistic regression, support vector machines, and neural networks [42].

3.2.1 Multiclass classification

The goal of multiclass classification is to assign a single input to one of several possible categories or classes. It is a type of supervised learning problem. In multiclass classification, the output variable is a discrete value representing the predicted class, rather than a continuous value as in regression. One common method for performing multiclass classification is to train several binary classifiers, each distinguishing between one class and the others, and then combine their outputs using a voting system or decision function. The time complexity of multiclass classification algorithms can be expressed in terms of the number of classes (denoted by k) and the number of training instances (denoted by n), with $O(kn^2)$ for some algorithms like the Support Vector Machine with the One-versus-All approach and $O(knlog(n))$ for others like softmax regression [56].

3.2.2 Naïve Bayes

Naïve Bayes is a probabilistic classification algorithm that is often used for text classification, spam filtering, and sentiment analysis [42], [43]. The algorithm is based on Bayes theorem, which provides a way to calculate the probability of a hypothesis given some evidence. To use Nave Bayes, we must first do two steps: *training* and *prediction*. The training and prediction stages of this method are described by the algorithms 1 and 2. See Appendix B.1.1 for more details regarding Naïve Bayes.

3.2.3 K-Nearest Neighbours

The k-Nearest Neighbors (k-NN) algorithm is a well-known and straightforward classification algorithm. It works on the proximity principle and produces predictions by locating the k closest neighbors to a given data point. One vital stage in the k-NN technique is to establish a distance metric that evaluates the closeness or dissimilarity of data points. Metrics like Euclidean distance and Manhattan distance

quantify the spatial separation between points in a feature space [57]. See Appendix B.1.2 for more information.

3.2.4 Decision Tree

A Decision Tree is a hierarchical tree structure that is commonly used for classification and regression problems. It recursively splits the data into subsets based on the values of the features in the dataset. The splits are chosen based on the feature that provides the most information gain, calculated using metrics such as the Gini impurity or entropy. Information gain is a measure of the reduction in uncertainty that is achieved by splitting the data based on a particular feature. The Gini impurity, on the other hand, is a measure of how often a randomly chosen element from the set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset. To split the data, the Decision Tree algorithm considers all possible splits for each feature and chooses the one that maximizes the information gain. This process is repeated recursively until a stopping criterion is met, such as a maximum depth or a minimum number of samples in a leaf node [56], [57]. More in-depth details regarding DT see Appendix B.1.3.

3.2.5 Support Vector Machine

Support Vector Machines (SVM) is a powerful machine learning algorithm that finds the best possible boundary to separate two classes in input data. It does so by finding a hyperplane in the input space that maximally separates the two classes, with the hyperplane chosen so that the distance between the hyperplane and the nearest data points from each class is maximized. These nearest data points are called support vectors, and the hyperplane that separates the two classes is defined as $w^T x + b = 0$, where w is the weight vector, x is the input vector, and b is the bias term. To achieve this, the SVM algorithm optimizes a cost function that penalizes misclassifications and encourages a large margin between the hyperplane and the nearest data points. The cost function is typically a convex optimization problem that can be solved using optimization algorithms like gradient descent [56], [57]. See Appendix B.1.4 for more information.

3.2.6 Cross-validation

Cross-validation is a powerful technique for assessing the performance of a machine-learning model when the amount of available data is limited. It involves dividing the available data into k non-overlapping folds, each containing approximately the same number of instances. The model is trained on $k-1$ folds and tested on the fold that was held out. This process is repeated k times, with each fold being used exactly once as the test set, to provide a reliable estimate of the model's generalization performance [56], [57].

3.3 Data

Table 3.1 provides a list of open-source datasets used for feature selection benchmarking. It contains information such as the dataset's name, the number of instances, the number of features, the number of classes, and the dataset's source. A careful selection of datasets is required to ensure that the results of the FS benchmarking study are generalizable and meaningful. We can evaluate the performance of FS methods across different types of data by selecting a variety of datasets from various sources, with varying numbers of features and classes.

- **Name**: This column contains the name of the open-source dataset.
- **Instances**: This column contains the number of instances or observations in the dataset.
- **Features**: This column contains the number of features or variables in the dataset.
- **Classes**: This column contains the number of classes or categories in the dataset.
- **Sources**: This column contains the source or origin of the dataset, such as the website or repository where it can be found.

3.3.1 Benefits and Limitations of Open Source Microarray Data

The use of open-source microarray data provides several benefits, such as reducing the cost and time required to collect and generate data, increasing the transparency and reproducibility of research, and allowing for the validation of existing findings. However, there are also limitations to using open-source microarray data, such as the potential for variability in experimental conditions, differences in data quality and preprocessing, and the lack of control over the original experimental design [17], [80], [81].

3.4 Evaluation framework

To assess the accuracy of the model training, cross-validation (CV) is used. Figure 3.2 shows a visual representation of how CV is designed. Each row is divided into equal data chunks. The *green* boxes represent test data, while the *red* boxes represent train data. The test data is moved one block up after each iteration (row). There are k chunks of data and k iterations to validate our model. For each iteration, the validation data is shifted one block further to avoid overfitting/underfitting and reuse the same data for multiple training sessions.

For each iteration in the validation process, several steps are done to the model. Figure 3.3 represents the steps taken. Firstly, the **Inputs** is sent to the *classifier* and the dataset is divided into chucks (training and testing), $1 - 1/k$ *training data*

Table 3.1:
Benchmark datasets used for benchmarking

Name	Instances	Features	Classes	Disease	Source
Alon	62	2000	2	Colon Cancer	Alon, et al. (1999) [58]
Borovecki	31	22283	2	Huntingtons Disease	Broovecki, et al. (2005) [59]
Burcyznski	127	22283	3	Crohns Disease	Burcyznski, et al. (2006) [60]
Christensen	217	1413	2	*not specified*	Christiansen, et al. (2009) [61]
Chin	118	22215	2	Breast Cancer	Chin, et al. (2006) [62]
Chowdary	104	22283	2	Breast Cancer	Chowdary, et al. (2006) [63]
Chiaretti	31	12625	2	Leukemia	Chiaretti, et al. (2004) [64]
Gordon	181	12533	2	Lung Cancer	Gordon, et al. (2002) [65]
Golub	72	7129	2	Leukemia	Golub, et al. (1999) [66]
Gravier	168	2905	2	Breast Cancer	Gravier, et al. (2010) [67]
Khan	63	2308	4	SRBCT	Khan, et al. (2001) [68]
Nakayama	105	22283	10	Sarcoma	Nakayama, et al. (2007) [69]
Pomeroy	60	7128	2	CNSET	Pomeroy, et al. (2002) [70]
Shipp	77	6817	2	Lymphoma	Shipp, et al. (2002) [71]
Singh	102	12600	2	Prostate Cancer	Singh, et al. (2002) [72]
Sorlie	85	456	5	Breast Cancer	Sorlie, et al. (2001) [73]
Su	102	5565	4	*not specified*	Su, et al. (2002) [74]
Subramanian	50	10100	2	*not specified*	Subramanian, et al. (2005) [75]
Sun	180	54613	4	Glioma	Sun, et al. (2006) [76]
Tian	173	12625	2	Myeloma	Tian, et al. (2003) [77]
West	49	7129	2	Breast Cancer	West, et al. (2001) [78]
Yeoh	248	12625	6	Leukemia	Yeoh, et al. (2002) [79]

and $1/k$ *test data*. The only processing on the test data is FS, otherwise is the test data not touched for the remanding until the model is tested on it respectively. Furthermore, the training data is run through the FS. Notice, the "FS Method" has only one path, the *wrapper*. This is because the wrapper is dependent on the *classifier* to perform FS. As mentioned in Chapter 1, *filters* perform a statistical calculation (ranking) instead of depending on the *classifier*. Afterwards, when the FS has been performed, the data is fitted into the *classifier* model. Lastly, the trained model is evaluated and the **Output** is the *classification accuracy*.

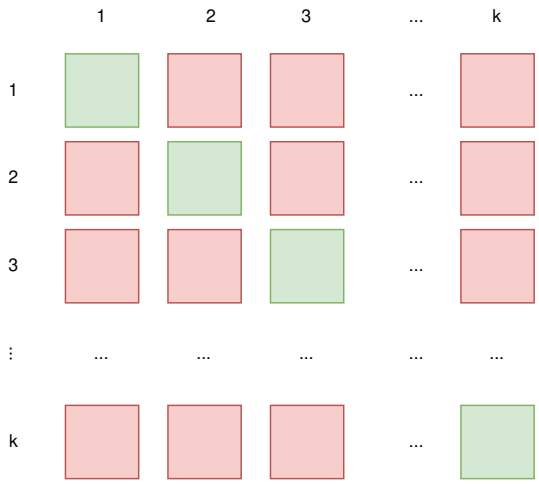

Figure 3.2: Cross-validation for evaluation

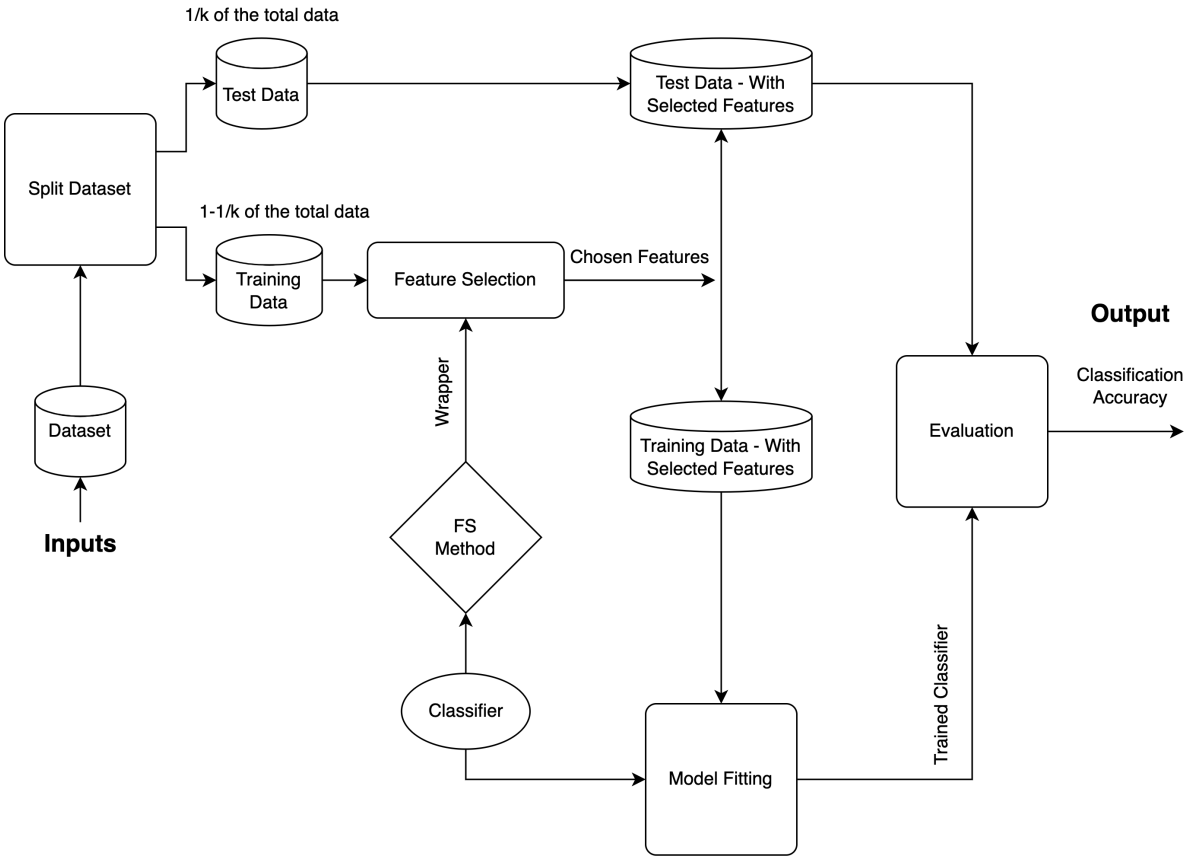

Figure 3.3: One iteration of the evaluation methodology

4

Results

4.1 Sensitivity Analysis

In this section, a comprehensive sensitivity analysis is performed to fine-tune the parameters of the SPFSR algorithm when utilized with microarray data. This algorithm is integrated with several classifiers, including the decision tree, k-nearest neighbour, support vector machine, and naive Bayes. Three significant parameters of the SPFSR algorithm - the number of gradient averaging (num_avg_grad), stall limit[1] (stall_limit), and the maximum number of iterations (max_iterations) - are examined in this analysis. These were selected due to their considerable influence on the algorithm's efficiency. Through the systematic adjustment of these parameters within a predetermined range, and subsequent evaluation of the associated performance metrics, we aim to boost the accuracy of SPFSR in feature selection for microarray data. It's important to note that this analysis is conducted on the *Alon* dataset, a representative sample chosen for its typical structure of microarray datasets, comprising 62 instances and 2000 features.

The sensitivity analysis is constructed in the following way. The parameters mentioned above are set to their default values. For the analysis, only one value at a time is changed, see Table 4.1 for the default values.

Table 4.1: Parameters for SPFSR

Parameters	**value**
M (max no. of iterations)	100
Stall limit	35
m (no. of gradient averaging)	4
num_features	10

4.1.1 Stall Limit

Figure 4.1 illustrates the impact of varying the *stall limit* parameter on the performance of four distinct classifiers: Decision Tree, Naive Bayes, K-Nearest Neighbor, and Support Vector Machine. This visual representation allows for a more intuitive

[1]The algorithm stops if there are no improvements in the objective function during an interval of steps equal to stall limit.

understanding of the influence of the *stall limit* on the efficiency of these classifiers. For exact numerical values related to these variations, please refer to Appendix C.1.1.

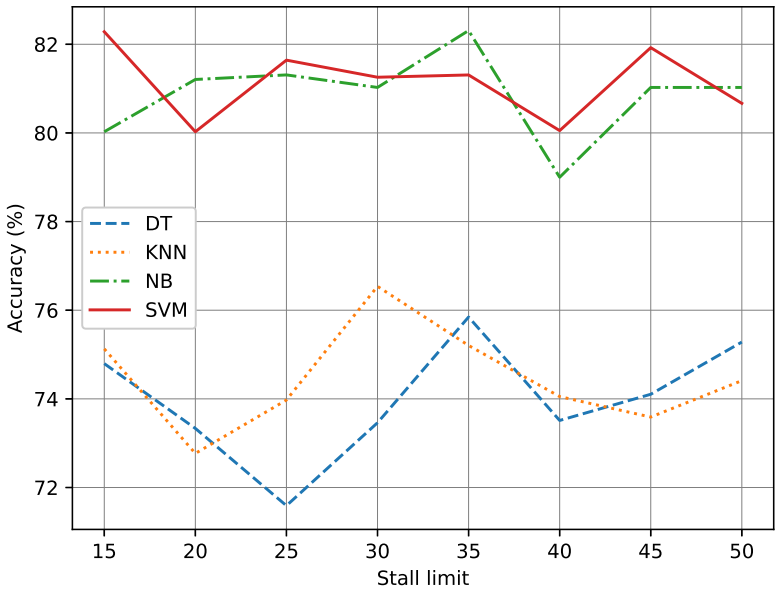

Figure 4.1: Mean CV accuracy percentages for different classifiers based on different stall limit values.

The findings show that the accuracy of the DT classifier varies as the stall limit parameter is changed. However, the accuracy remains within a narrow range of 71.59% to 75.85%. Notably, the accuracy reaches a peak of 75.85% at stall limit 35. Given the pattern and the fact that the variations in accuracy are so minor, it may not be required to fine-tune the stall limit value any further. The default value of 35 appears to provide enough performance, and there is no discernible benefit to adjusting it to the other levels examined.

The analysis reveals that the variations in the accuracy of the NB classifier remain within a range of around 79.00% to 82.31%. It is noteworthy that the accuracy reaches a peak of 82.31% at stall limit 35, which is greater than the other values examined. Thus similar to DT, it may not be required to fine-tune the stall limit value any further. The default value of 35 seems sufficient to produce acceptable performance without requiring extra iterations.

Regarding the KNN classifier, similar to NB, the variations in accuracy are rather slight, with the accuracy remaining within a range of 72.77% to 76.54%. Note that the accuracy reaches its peak at maximum iterations 30, with a value of 76.54%. Following that, the accuracy reduces significantly for stall limit 35 and 40 before returning to normal. Accordingly, based on the observed trend and the very slight fluctuations in accuracy, fine-tuning the stall limit parameter will produce a slightly higher result. Hence, the parameter value should be changed to 30 for the KNN

classifier.

For the SVM classifier, the accuracy remains within a range of roughly 80.03% to 82.28%. Surprisingly, the highest accuracy is attained at stall limit 15, with 82.28% accuracy. However, for maximum iterations between 20 and 45, the accuracy stays very stable, with values ranging from 80.03% to 81.92%. As the stall limit of 35 still provides acceptable performance in terms of accuracy values, we prefer to stick to this default setting for the SVM classifier.

4.1.2 Max Iterations

Figure 4.2 shows the impact of setting different values for the parameter of *maximum iteration* on the performance of four classifiers for feature selection methods (See Appendix C.1.2 for the exact percentages).

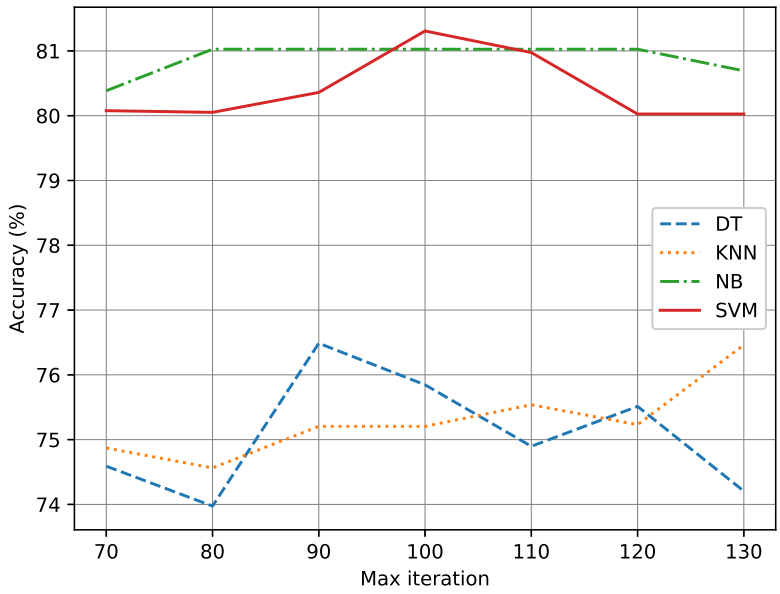

Figure 4.2: Mean CV accuracy percentages for different classifiers based on different maximum number of iterations.

Based on these results, it appears that setting the maximum number of iterations to 90 provides the DT classifier with the highest performance. Accordingly, we set this parameter for DT as 90 across our numerical experiments.

NB and SVM classifiers seem to be insensitive to changes in the max iterations parameter. Accordingly, modifications in the max iterations parameter have little effect on the accuracy of these classifiers. Thus we set this parameter for both classifiers to the default value.

In general, unless there are compelling reasons to investigate additional iterations, setting the max iterations option to 100 (the typical value) should be adequate for obtaining decent performance with minimal computational overhead. It's always

important to achieve a balance between computing expense and performance, and sticking with the default setting of 100 iterations seems fair in this case. This is also true for the KNN classifier as the increase in performance with high values of the max iterations is minor, so fine-tuning the max iterations parameter for the KNN classifier may also not be necessary. Any value within the range of iterations tested, including the default should be sufficient to deliver acceptable performance without the need for more iterations.

4.1.3 Number of Gradient Averaging

Figure 4.3 demonstrates the effect of different numbers of gradient averaging on the performance of the four classifiers. The exact percentages are reported in Appendix C.3.

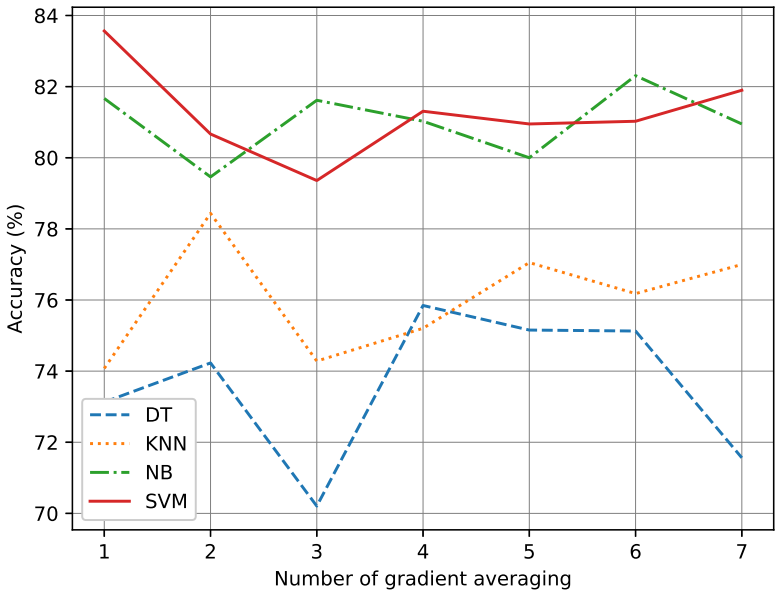

Figure 4.3: Mean CV accuracy percentages for different classifiers based on different gradient averaging values.

The DT classifier's accuracy varies as the gradient averaging parameter is changed. The accuracy reaches a maximum of 75.85% at the maximum iteration of 4. It is critical to choose a suitable value for the gradient averaging option based on the observed pattern. Setting the maximum number of iterations to 4 may be a good decision because it achieves the highest accuracy and subsequent increases in iterations result in insignificant changes in performance.

We observe that the accuracy of the NB classifier does not vary much as the gradient averaging parameter is changed. That's why we decided to set this parameter to the default value of four.

For the KNN classifier, the accuracy reaches its peak of 78.44% after the gradient averaging of two. Notably, after four cycles of gradient averaging, the accuracy drops

marginally to 75.20%. Therefore, we set the number of gradient averaging for our experiments to two as it achieves the maximum accuracy.

The accuracy of the SVM classifier varies as the number of gradient-averaging iterations changes. The accuracy reaches its peak of 83.56% without any gradient averaging. However, lower numbers for gradient averaging may result in the SPFSR algorithm not converging effectively. Thus, setting the number of gradient averaging to the default value of 4 may be a good decision as it achieves acceptable accuracy, and additional increases in this parameter result in insignificant changes in performance.

4.2 Comparison of the Algorithms

In this section, we delve into the performance evaluation of various FS approaches, particularly focusing on their accuracy across different classifiers and datasets. Through a series of figures, tables, and statistical tests, we aim to provide a comprehensive understanding of how each method performs, with a special emphasis on the SPFSR technique. The discussions and visual representations will clarify the comparative strengths and weaknesses of these methods, offering insights into their efficacy in feature selection tasks.

The mean accuracy for each FS approach, as applied to each classifier, is depicted in Figure 4.4 and Figure 4.5 for the Chowdary and Khan datasets, respectively. Results for other datasets can be found in Appendix D. Additionally, Figure 4.6 presents the average accuracy across all FS techniques for each classifier, considering all datasets. These plots showcase the mean accuracy achieved by various FS methods for each classifier. Within each plot, the performance of different FS techniques is compared in terms of mean accuracy. The x-axis represents the features selected by each FS method, while the y-axis indicates the corresponding accuracy percentage. Notably, the SPFSR technique exhibits a marginally superior accuracy compared to other methods, as evident in the figures.

When analysing Figure 4.6, it becomes evident that in the case of the Naive Bayes and Decision Tree models, the proposed SPFSR exhibits comparable performance. However, upon closer examination, it is apparent that the secondary feature selection methods differ for these classifiers. For Naive Bayes, the FScore ranks second, whereas for the Decision Tree, RFI takes the second spot, albeit with the FScore closely trailing behind. Transitioning to the evaluation of KNN and SVM, we observe that SPFSR consistently delivers strong performance, although it doesn't secure the top position. In the case of KNN, RFI emerges as the superior feature selection method, even though SPFSR achieves slightly lower accuracy. Furthermore, SPFSR outperforms the remaining feature selection methods in terms of accuracy. The performance of SVM stands out as well. SFS leads as the primary feature selection method, with GA taking second place. Interestingly, the accuracy of GA improves with an increasing number of features. SPFSR follows closely behind GA in terms of accuracy, with only a slight difference in percentages.

Figure 4.7 displays boxplots of the mean accuracy attained by the FS methods for

each distinct classifier, without accounting for the number of features. These visuals provide a summary of the performance of various FS techniques across different classifiers. The x-axis denotes the FS methods, while the y-axis indicates their respective accuracy percentages. As with earlier figures, it's evident that SPFSR outperforms other approaches in the feature selection task.

Table 4.2 presents the average cross-validation accuracy percentages of SPFSR and the other eight FR methods across all datasets, along with the number of features selected for each classifier. For additional context, the table also includes the average classification accuracy when using all available features. A review of the average results in Table 4.2 reveals that SPFSR stands out, achieving an impressive average classification accuracy of 72.46%.

Figure 4.8 showcases the win-tie-loss results from a paired t-test, performed at a 5% significance level, pitting SPFSR against other FR methods across all classifiers for each dataset. Our findings underscore SPFSR's consistent edge over other FS techniques. While there were instances where the performance was neck-and-neck, SPFSR's overall dominance was evident. Broadly speaking, SPFSR emerged as the top method for all four classifiers in most scenarios. Specifically, in over 93.8% of the tests, SPFSR either matched or significantly surpassed the performance of other FR algorithms. Delving deeper, SPFSR exhibited a statistically significant improvement in more than 69% of the tests.

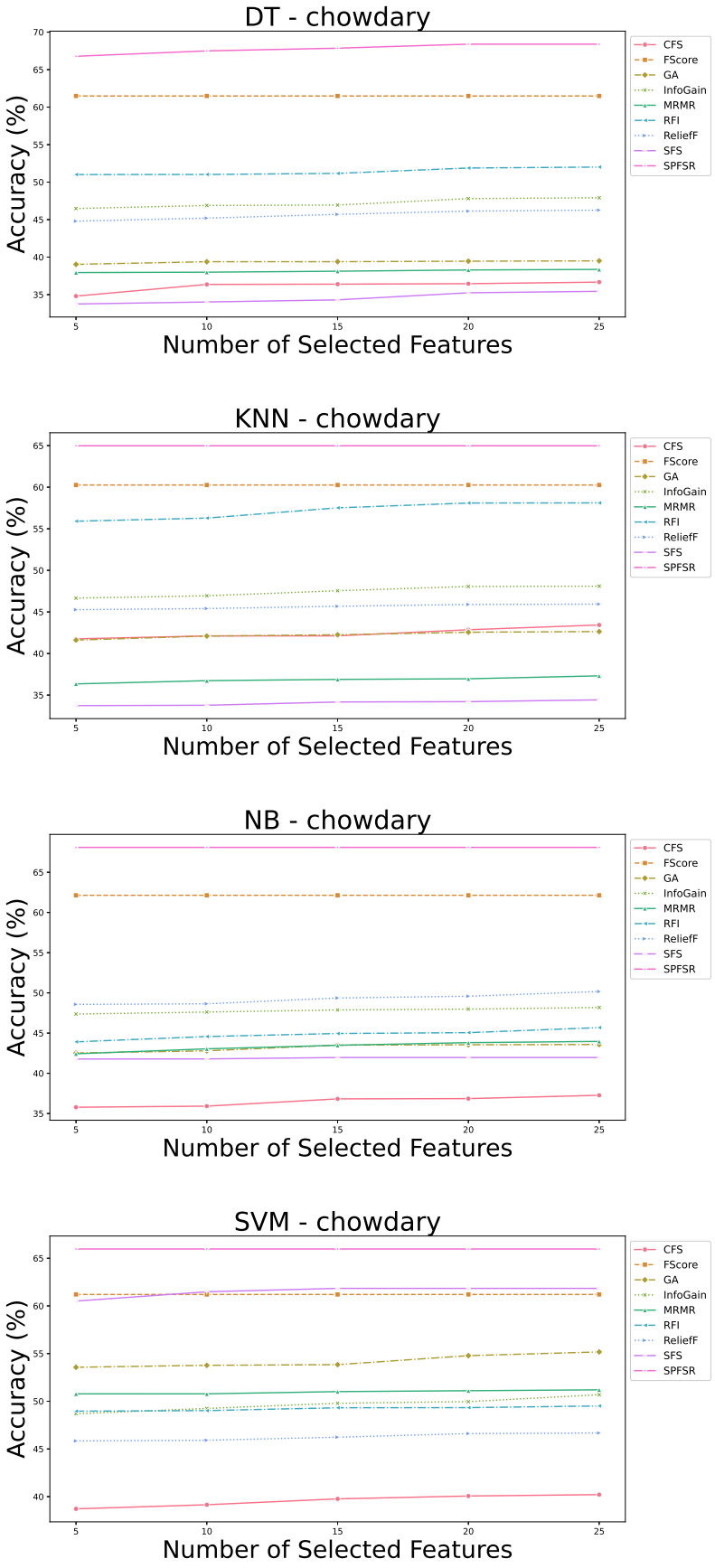

Figure 4.4: The average accuracy - Chowdary

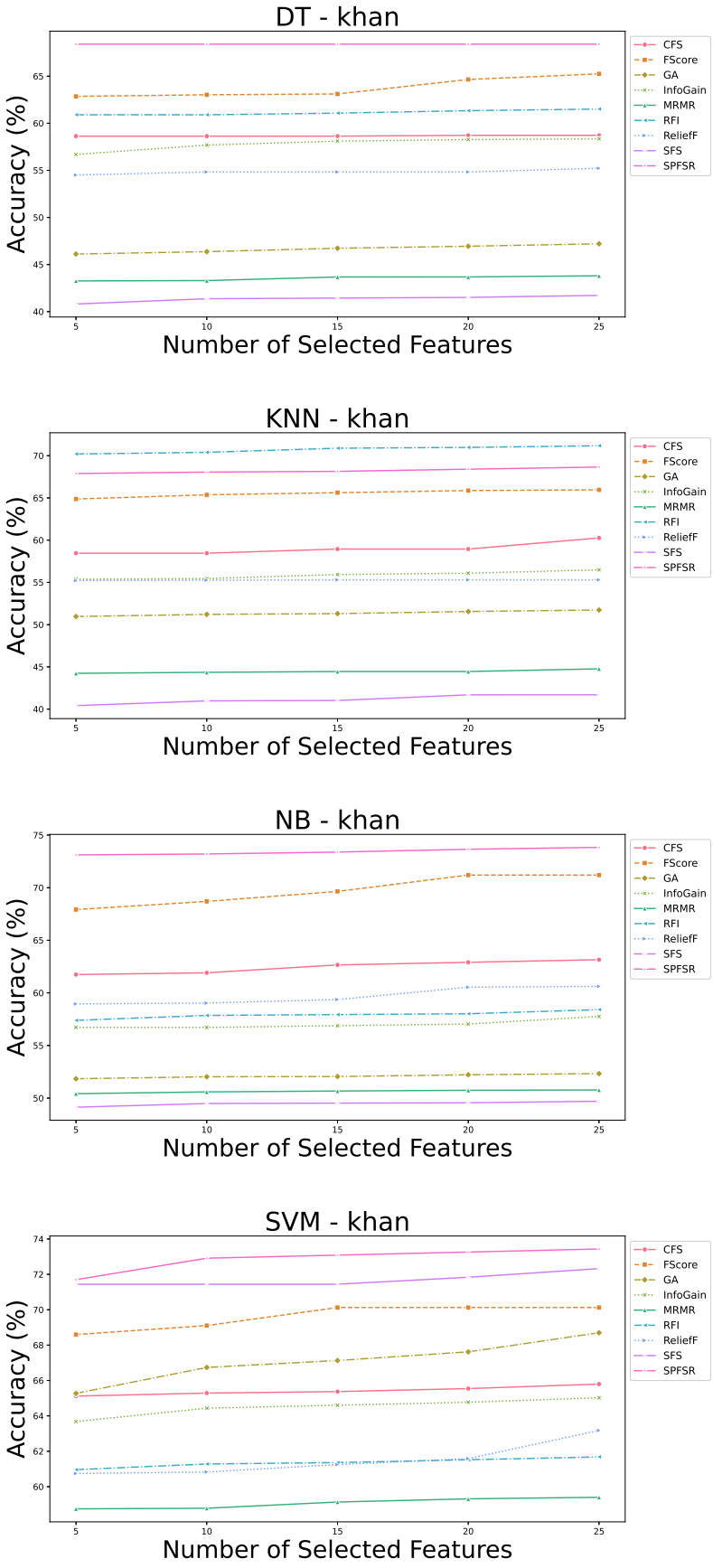

Figure 4.5: The average accuracy - Khan

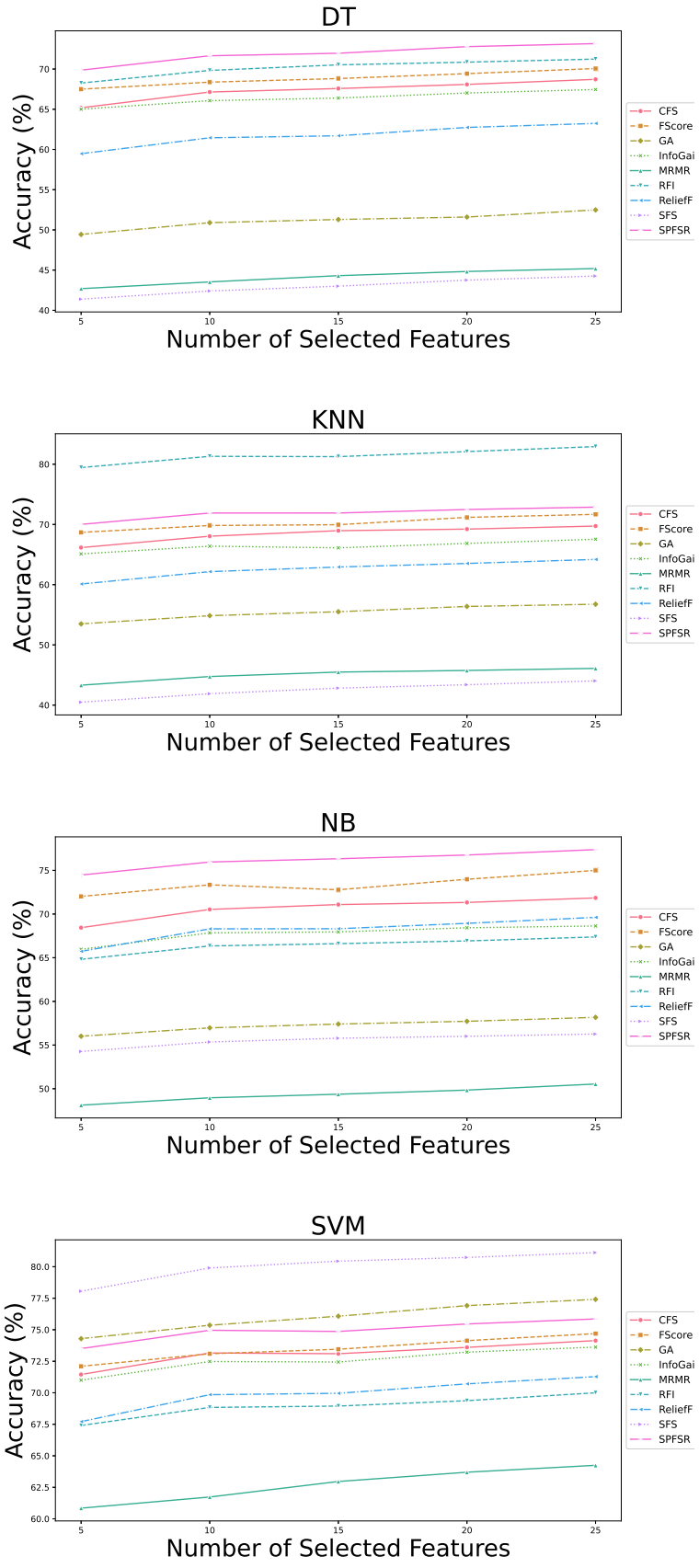

Figure 4.6: The average accuracy for each classifier

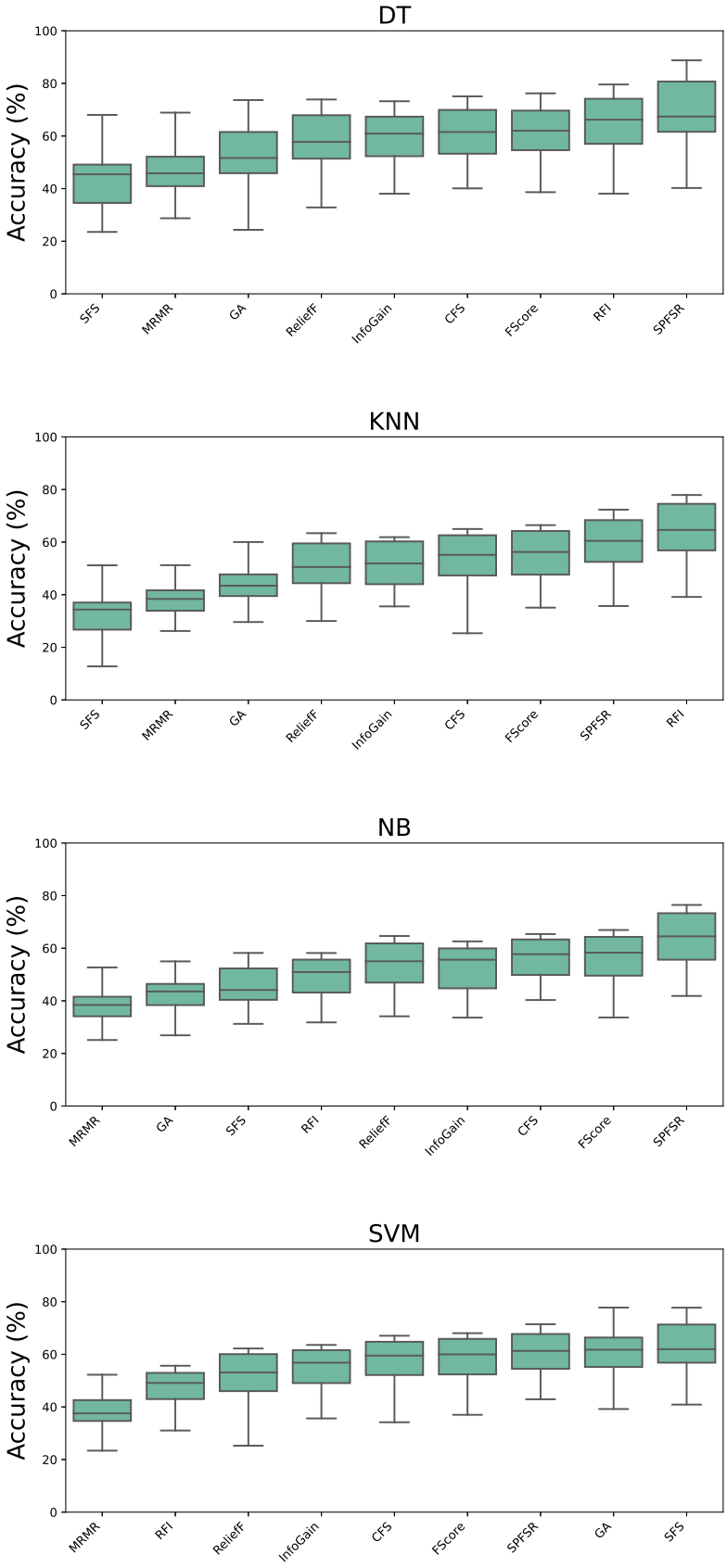

Figure 4.7: Percentage point improvement of the FR methods for each wrapper.

Table 4.2: The average classification accuracy of the FS methods with 5, 10, 15, 20 and 25 features

Classifier	CFS	FScore	GA	InfoGain	MRMR	RFI	ReliefF	SFS	SPFSR
DT	62.67	67.18	53.67	62.36	40.90	67.69	62.24	40.45	68.75
KNN	68.39	70.22	55.34	66.40	48.91	81.40	64.20	42.46	71.67
NB	70.63	71.71	57.23	67.77	49.32	66.41	67.95	55.51	72.75
SVM	73.10	73.19	77.49	70.04	69.55	64.60	69.90	81.67	76.66
Average	68.70	70.58	60.93	66.64	52.17	70.02	66.07	55.02	72.46

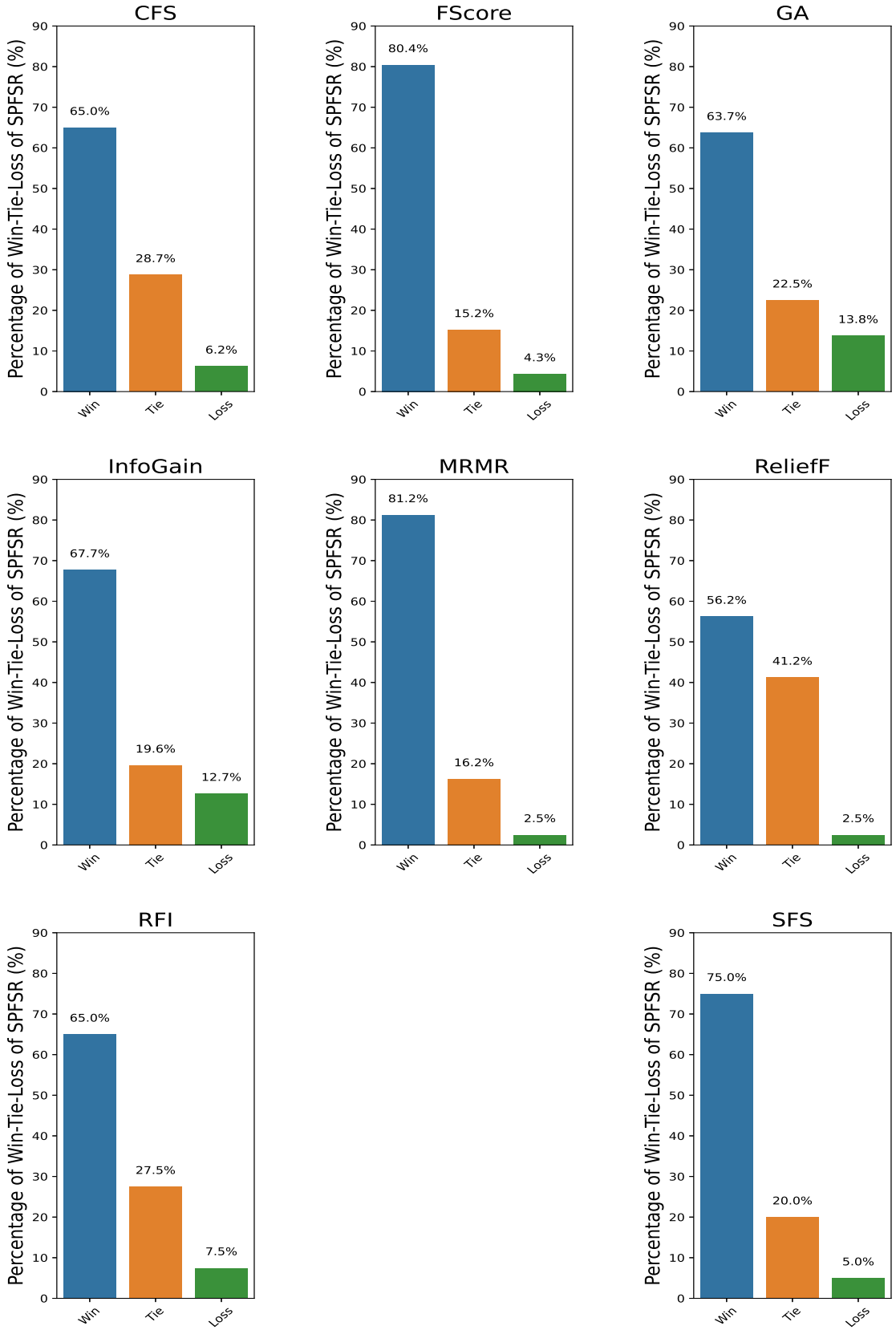

Figure 4.8: SPFSR's percentage of wins, ties, and losses in the classification job when compared to other FR approaches.

5

Conclusion

In our research, we conducted a comprehensive benchmarking of the SPFSR algorithm, comparing it with various FS methods tailored for microarray data. The SPFSR algorithm excels in finding informative features while effectively discarding the irrelevant ones. This can be interpreted as categorizing features with top-ranking weights among the highest k elements as "important" and relegating the rest as "unimportant". A notable aspect of SPFSR is its utilization of the non-monotone BB search approach, which ensures rapid convergence. This is further enhanced by integrating gradient averaging and gain smoothing, mitigating the effects of noise in gradient estimation and loss function evaluation.

Our empirical analysis spanned classification tasks across 22 datasets, assessing the performance of SPFSR through a 5-repeated 5-fold cross-validation. This evaluation was compared against models using the complete feature set and other prominent FR techniques, such as CFS, FScore, GA, InfoGain, MRMR, ReliefF, RFI, and SFS.

The results compellingly indicate that SPFSR, on average, surpasses other models across the assessed machine learning algorithms. When compared against individual FS methods, SPFSR consistently showcased either a statistically equivalent or a significant improvement, achieving this distinction in nearly 94% of the experiments.

Bibliography

[1] D. V. Akman, M. Malekipirbazari, Z. D. Yenice, *et al.*, "k-best feature selection and ranking via stochastic approximation," *Expert Systems with Applications*, vol. 213, p. 118 864, 2023.

[2] T. Barrett, D. B. Troup, S. E. Wilhite, *et al.*, "NCBI GEO: Mining Tens of Millions of Expression Profiles–Database and Tools Update.," *Nucleic Acids Res*, vol. 35, no. Database issue, pp. D760–5, Jan. 2007.

[3] J. Chen, E. E. Bardes, B. J. Aronow, and A. G. Jegga, "ToppGene Suite for gene list enrichment analysis and candidate gene prioritization.," *Nucleic Acids Res*, vol. 37, no. Web Server issue, W305–11, Jul. 2009.

[4] D. M. Mutch, A. Berger, R. Mansourian, A. Rytz, and M.-A. Roberts, "Microarray data analysis: a practical approach for selecting differentially expressed genes," *Genome Biology*, vol. 2, no. 12, preprint0009.1, 2001.

[5] H.-Y. Chen, S.-L. Yu, C.-H. Chen, *et al.*, "A Five-Gene Signature and Clinical Outcome in Non–Small-Cell Lung Cancer," *New England Journal of Medicine*, vol. 356, no. 1, pp. 11–20, 2007.

[6] E. R. Gamazon, R. S. Huang, M. E. Dolan, and N. J. Cox, "Copy number polymorphisms and anticancer pharmacogenomics," *Genome biology*, vol. 12, no. 5, pp. 1–12, 2011.

[7] P. Du, W. A. Kibbe, and S. M. Lin, "lumi: a pipeline for processing Illumina microarray," *Bioinformatics*, vol. 24, no. 13, pp. 1547–1548, May 2008.

[8] B. M. Bolstad, R. A. Irizarry, M. Åstrand, and T. P. Speed, "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias," *Bioinformatics*, vol. 19, no. 2, pp. 185–193, 2003.

[9] H. Jiang, Y. Deng, H.-S. Chen, *et al.*, "Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes," *BMC bioinformatics*, vol. 5, no. 1, pp. 1–12, 2004.

[10] Y. Saeys, I. Inza, and P. Larranaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[11] Y. H. Yang, S. Dudoit, P. Luu, *et al.*, "Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation," *Nucleic acids research*, vol. 30, no. 4, e15–e15, 2002.

[12] A. A. Alizadeh, M. B. Eisen, R. E. Davis, *et al.*, "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, no. 6769, pp. 503–511, 2000.

[13] F. Cappuzzo, M. Varella-Garcia, H. Shigematsu, *et al.*, "Increased HER2 Gene Copy Number Is Associated With Response to Gefitinib Therapy in Epider-

mal Growth Factor Receptor–Positive Non–Small-Cell Lung Cancer Patients," *Journal of Clinical Oncology*, vol. 23, no. 22, pp. 5007–5018, 2005.

[14] Draghici, Sorin and Khatri, Purvesh and Bhavsar, Pratik and Shah, Abhik and Krawetz, Stephen A and Tainsky, Michael A, "Onto-Tools, the toolkit of the modern biologist: Onto-Express, Onto-Compare, Onto-Design and Onto-Translate.," *Nucleic Acids Res*, vol. 31, no. 13, pp. 3775–3781, Jul. 2003.

[15] Z. Wang, M. Gerstein, and M. Snyder, "RNA-Seq: a revolutionary tool for transcriptomics," *Nature reviews genetics*, vol. 10, no. 1, pp. 57–63, 2009.

[16] C. Wang, B. Gong, P. R. Bushel, J. Thierry-Mieg, and Thierry-Mieg, "The concordance between RNA-seq and microarray data depends on chemical treatment and transcript abundance.," *Nat Biotechnol*, vol. 32, no. 9, pp. 926–932, Sep. 2014.

[17] L. Shi and Campbell, "The MicroArray Quality Control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models," *Nature Biotechnology*, vol. 28, no. 8, pp. 827–838, 2010.

[18] K. Raza, "Analysis of microarray data using artificial intelligence based techniques," in *Handbook of Research on Computational Intelligence Applications in Bioinformatics*, IGI Global, 2016, pp. 216–239.

[19] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[20] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.

[21] C. F. Aliferis, I. Tsamardinos, and A. Statnikov, "HITON: a novel Markov Blanket algorithm for optimal variable selection," in *AMIA annual symposium proceedings*, American Medical Informatics Association, vol. 2003, 2003, p. 21.

[22] Y. Yang and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," in *International Conference on Machine Learning*, 1997.

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

[24] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[25] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[26] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern recognition letters*, vol. 31, no. 14, pp. 2225–2236, 2010.

[27] J. Brownlee, *How to Choose a Feature Selection Method For Machine Learning, howpublished = https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/*, Accessed: 2023-03-1, 2021.

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

[29] M. A. Hall, "Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning," in *Proceedings of the Seventeenth International Con-*

ference on Machine Learning, ser. ICML '00, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 359–366, ISBN: 1558607072.

[30] D. Suriyamurthi and V. Thambusamy, "A Comprehensive survey on Filter approach to feature selection methods for High Dimensional Data," Jan. 2015.

[31] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data classification: Algorithms and applications*, p. 37, 2014.

[32] J. Han and M. Kamber, *Data mining: Concepts and techniques*, en. 2012.

[33] J. Abdollahi and B. Nouri-Moghaddam, "A hybrid method for heart disease diagnosis utilizing feature selection based ensemble classifier model generation," *Iran Journal of Computer Science*, vol. 5, no. 3, pp. 229–246, 2022.

[34] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005. DOI: 10.1109/TPAMI.2005.159.

[35] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 1, pp. 131–156, 1997.

[36] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, Aug. 2007.

[37] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, pp. 389–422, 2002.

[38] K. Kira and L. A. Rendell, "The Feature Selection Problem: Traditional Methods and a New Algorithm," in *AAAI Conference on Artificial Intelligence*, 1992.

[39] P. Somol, B. Baesens, P. Pudil, and J. Vanthienen, "Filter-versus wrapper-based feature selection for credit scoring," *International Journal of Intelligent Systems*, vol. 20, no. 10, pp. 985–999, 2005.

[40] B. A. Md. Alamgir Sarder Md. Maniruzzaman, "Feature Selection and Classification of Leukemia Cancer Using Machine Learning Techniques," *Machine learning*, vol. 5, pp. 18–27, 2020.

[41] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, 2018.

[42] L. Igual, S. Segu, L. Igual, and S. Segu, *Introduction to data science*. Springer, 2017.

[43] V. Aksakalli, Z. D. Yenice, M. Malekipirbazari, and K. Kargar, "Feature selection using stochastic approximation with Barzilai and Borwein non-monotone gains," *Computers & Operations Research*, vol. 132, p. 105 334, 2021.

[44] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE transactions on automatic control*, vol. 37, no. 3, pp. 332–341, 1992.

[45] J. C. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins apl technical digest*, vol. 19, no. 4, pp. 482–492, 1998.

[46] T. N. Nuklianggraita, A. Adiwijaya, and A. Aditsania, "On the Feature Selection of Microarray Data for Cancer Detection based on Random Forest Classifier," *JURNAL INFOTEL*, vol. 12, no. 3, pp. 89–96, 2020.

[47] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, J. Bentez, and F. Herrera, "A review of microarray datasets and applied feature selection methods," *Information Sciences*, vol. 282, pp. 111–135, 2014.

[48] Z. Zhu, Y.-S. Ong, and M. Dash, "Wrapper–Filter Feature Selection Algorithm Using a Memetic Framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 70–76, 2007.

[49] P. E. Meyer, C. Schretter, and G. Bontempi, "Information-Theoretic Feature Selection in Microarray Data Using Variable Complementarity," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 3, pp. 261–274, 2008.

[50] J. Apolloni, G. Leguizamón, and E. Alba, "Two hybrid wrapper-filter feature selection algorithms applied to high-dimensional microarray experiments," *Applied Soft Computing*, vol. 38, pp. 922–932, 2016.

[51] M. Pirooznia, J. Y. Yang, M. Q. Yang, and Y. Deng, "A comparative study of different machine learning methods on microarray gene expression data," *BMC Genomics*, vol. 9, no. 1, S13, 2008.

[52] L.-Y. Chuang, C.-H. Yang, K.-C. Wu, and C.-H. Yang, "A hybrid feature selection method for DNA microarray data," *Computers in Biology and Medicine*, vol. 41, no. 4, pp. 228–237, 2011.

[53] M. Ghosh, S. Adhikary, K. K. Ghosh, A. Sardar, S. Begum, and R. Sarkar, "Genetic algorithm based cancerous gene identification from microarray data using ensemble of filter methods," *Medical & Biological Engineering & Computing*, vol. 57, no. 1, pp. 159–176, 2019.

[54] 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.

[55] Y. Wang, I. V. Tetko, M. A. Hall, *et al.*, "Gene selection from microarray data for cancer classification—a machine learning approach," *Computational Biology and Chemistry*, vol. 29, no. 1, pp. 37–46, 2005.

[56] A. Geron, *Hands-on machine learning with scikit-learn, keras, and TensorFlow*, 2nd ed. Sebastopol, CA: O'Reilly Media, Oct. 2019.

[57] S. S. Skiena, *The Data Science Design Manual* (Texts in computer science), en, 1st ed. Cham, Switzerland: Springer International Publishing, Jul. 2017.

[58] U. Alon, N. Barkai, D. A. Notterman, *et al.*, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays.," *Proc Natl Acad Sci U S A*, vol. 96, no. 12, pp. 6745–6750, Jun. 1999.

[59] F. Borovecki, L. Lovrecic, J. Zhou, *et al.*, "Genome-wide expression profiling of human blood reveals biomarkers for Huntington's disease.," *Proc Natl Acad Sci U S A*, vol. 102, no. 31, pp. 11 023–11 028, Aug. 2005.

[60] M. E. Burczynski, R. L. Peterson, N. C. Twine, *et al.*, "Molecular classification of Crohn's disease and ulcerative colitis patients using transcriptional profiles in peripheral blood mononuclear cells.," *J Mol Diagn*, vol. 8, no. 1, pp. 51–61, Feb. 2006.

[61] B. C. Christensen, E. A. Houseman, C. J. Marsit, *et al.*, "Aging and environmental exposures alter tissue-specific DNA methylation dependent upon CpG island context.," *PLoS Genet*, vol. 5, no. 8, e1000602, Aug. 2009.

[62] K. Chin, S. DeVries, J. Fridlyand, *et al.*, "Genomic and transcriptional aberrations linked to breast cancer pathophysiologies.," *Cancer Cell*, vol. 10, no. 6, pp. 529–541, Dec. 2006.

[63] D. Chowdary, J. Lathrop, J. Skelton, *et al.*, "Prognostic gene expression signatures can be measured in tissues collected in RNAlater preservative.," *J Mol Diagn*, vol. 8, no. 1, pp. 31–39, Feb. 2006.

[64] S. Chiaretti, X. Li, R. Gentleman, *et al.*, "Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival.," *Blood*, vol. 103, no. 7, pp. 2771–2778, Apr. 2004.

[65] G. J. Gordon, R. V. Jensen, L.-L. Hsiao, *et al.*, "Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma.," *Cancer Res*, vol. 62, no. 17, pp. 4963–4967, Sep. 2002.

[66] T. R. Golub, D. K. Slonim, P. Tamayo, *et al.*, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.," *Science*, vol. 286, no. 5439, pp. 531–537, Oct. 1999.

[67] E. Gravier, G. Pierron, A. Vincent-Salomon, *et al.*, "A prognostic DNA signature for T1T2 node-negative breast cancer patients.," *Genes Chromosomes Cancer*, vol. 49, no. 12, pp. 1125–1134, Dec. 2010.

[68] J. Khan, J. S. Wei, M. Ringnér, *et al.*, "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks.," *Nat Med*, vol. 7, no. 6, pp. 673–679, Jun. 2001.

[69] R. Nakayama, T. Nemoto, H. Takahashi, *et al.*, "Gene expression analysis of soft tissue sarcomas: characterization and reclassification of malignant fibrous histiocytoma.," *Mod Pathol*, vol. 20, no. 7, pp. 749–759, Jul. 2007.

[70] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, *et al.*, "Prediction of central nervous system embryonal tumour outcome based on gene expression.," *Nature*, vol. 415, no. 6870, pp. 436–442, Jan. 2002.

[71] M. A. Shipp, K. N. Ross, P. Tamayo, *et al.*, "Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning.," *Nat Med*, vol. 8, no. 1, pp. 68–74, Jan. 2002.

[72] D. Singh, P. G. Febbo, K. Ross, *et al.*, "Gene expression correlates of clinical prostate cancer behavior.," *Cancer Cell*, vol. 1, no. 2, pp. 203–209, Mar. 2002.

[73] T. Sørlie, C. M. Perou, R. Tibshirani, *et al.*, "Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications.," *Proc Natl Acad Sci U S A*, vol. 98, no. 19, pp. 10 869–10 874, Sep. 2001.

[74] A. I. Su, M. P. Cooke, K. A. Ching, *et al.*, "Large-scale analysis of the human and mouse transcriptomes," *Proceedings of the National Academy of Sciences*, vol. 99, no. 7, pp. 4465–4470, 2002.

[75] A. Subramanian, P. Tamayo, V. K. Mootha, *et al.*, "Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles," *Proceedings of the National Academy of Sciences*, vol. 102, no. 43, pp. 15 545–15 550, 2005.

[76] L. Sun, A.-M. Hui, Q. Su, *et al.*, "Neuronal and glioma-derived stem cell factor induces angiogenesis within the brain.," *Cancer Cell*, vol. 9, no. 4, pp. 287–300, Apr. 2006.

[77] E. Tian, F. Zhan, R. Walker, *et al.*, "The role of the Wnt-signaling antagonist DKK1 in the development of osteolytic lesions in multiple myeloma.," *N Engl J Med*, vol. 349, no. 26, pp. 2483–2494, Dec. 2003.

[78] M. West, C. Blanchette, H. Dressman, *et al.*, "Predicting the clinical status of human breast cancer by using gene expression profiles," *Proceedings of the National Academy of Sciences*, vol. 98, no. 20, pp. 11 462–11 467, 2001.

[79] E.-J. Yeoh, M. E. Ross, S. A. Shurtleff, *et al.*, "Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling.," *Cancer Cell*, vol. 1, no. 2, pp. 133–143, Mar. 2002.

[80] M. N. McCall, P. N. Murakami, M. Lukk, W. Huber, and R. A. Irizarry, "Assessing affymetrix GeneChip microarray quality," *BMC Bioinformatics*, vol. 12, no. 1, p. 137, 2011.

[81] J. Quackenbush, "Microarray data normalization and transformation," *Nature Genetics*, vol. 32, no. 4, pp. 496–501, 2002.

[82] D. K. Slonim and I. Yanai, "Getting started in gene expression microarray analysis.," *PLoS Comput Biol*, vol. 5, no. 10, e1000543, Oct. 2009.

[83] J. Sambrook and D. Russell, *Molecular Cloning*, 3rd ed. New York, NY: Cold Spring Harbor Laboratory Press, Dec. 2000.

[84] A. Schroeder, O. Mueller, S. Stocker, *et al.*, "The RIN: an RNA integrity number for assigning integrity values to RNA measurements.," *BMC Mol Biol*, vol. 7, p. 3, Jan. 2006.

[85] L. Garibyan and N. Avashia, "Polymerase chain reaction.," *J Invest Dermatol*, vol. 133, no. 3, pp. 1–4, Mar. 2013.

[86] T. Nolan, R. E. Hands, and S. A. Bustin, "Quantification of mRNA using real-time RT-PCR," *Nature Protocols*, vol. 1, no. 3, pp. 1559–1582, 2006.

[87] M. A. Valasek and J. J. Repa, "The power of real-time PCR.," *Adv Physiol Educ*, vol. 29, no. 3, pp. 151–159, Sep. 2005.

[88] P. Hegde, R. Qi, K. Abernathy, *et al.*, "A concise guide to cDNA microarray analysis.," *Biotechniques*, vol. 29, no. 3, pp. 548–550, Sep. 2000.

[89] A. L. Tarca, R. Romero, and S. Draghici, "Analysis of microarray experiments of gene expression profiling.," *Am J Obstet Gynecol*, vol. 195, no. 2, pp. 373–388, Aug. 2006.

[90] D. J. Lockhart, H. Dong, M. C. Byrne, *et al.*, "Expression monitoring by hybridization to high-density oligonucleotide arrays.," *Nat Biotechnol*, vol. 14, no. 13, pp. 1675–1680, Dec. 1996.

[91] C. Romualdi, S. Trevisan, B. Celegato, G. Costa, and G. Lanfranchi, "Improved detection of differentially expressed genes in microarray experiments through multiple scanning and image integration.," *Nucleic Acids Res*, vol. 31, no. 23, e149, Dec. 2003.

[92] R. Shyamsundar, Y. H. Kim, J. P. Higgins, *et al.*, "A DNA microarray survey of gene expression in normal human tissues," *Genome Biology*, vol. 6, no. 3, R22, 2005.

[93] C. H. Wilson, A. Tsykin, C. R. Wilkinson, and C. A. Abbott, "1 - Experimental Design and Analysis of Microarray Data," in *Applied Mycology and Biotechnology*, ser. Applied Mycology and Biotechnology, D. K. Arora, R. M. Berka, and G. B. Singh, Eds., vol. 6, Elsevier, 2006, pp. 1–36.

[94] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene Selection for Cancer Classification using Support Vector Machines," *Machine Learning*, vol. 46, no. 1, pp. 389–422, 2002.

A

Appendix 1

A.1 Sample Collection

The significance of high-quality RNA samples in generating trustworthy microarray data. They recommend that appropriate RNA isolation procedures be chosen based on sample parameters such as tissue type or cell culture system. They also go into detail on the potential causes of RNA degradation and the importance of using proper sample storage methods to prevent RNA degradation prior to microarray analysis. Physical damage to cells or tissues during sample collection, such as mechanical disruption or heat exposure, as well as enzymatic degradation by RNases found in cells, tissues, and environmental samples, are both potential causes of RNA degradation. Prolonged exposure to air or UV light, pH fluctuations, and the presence of metal ions are all variables that can contribute to RNA breakdown. It is critical to adopt adequate sample collecting protocols, including the use of RNA stabilizing reagents, and to keep samples at optimum temperatures to prevent RNA degradation and for accurate microarray analysis [82].

A.2 RNA Extraction

RNA extraction is another critical step in the microarray process, as it involves the isolation and purification of RNA from the collected samples. This step can be challenging, as RNA is a relatively unstable molecule that can be easily degraded by RNases and other contaminants. Various methods for RNA extraction have been developed, including organic extraction, column-based purification, and magnetic bead-based purification, each with its own advantages and limitations [83]. Additionally, new methods for quality control of RNA samples, such as the use of RNA integrity number (RIN) measurements, have been developed to ensure that the RNA extracted from samples is of high quality and suitable for microarray analysis [84].

A.3 Amplification

Amplification is an essential step in microarray analysis, as the amount of RNA that can be extracted from a single sample is often limited. However, traditional amplification methods, such as the polymerase chain reaction (PCR). Figure A.1

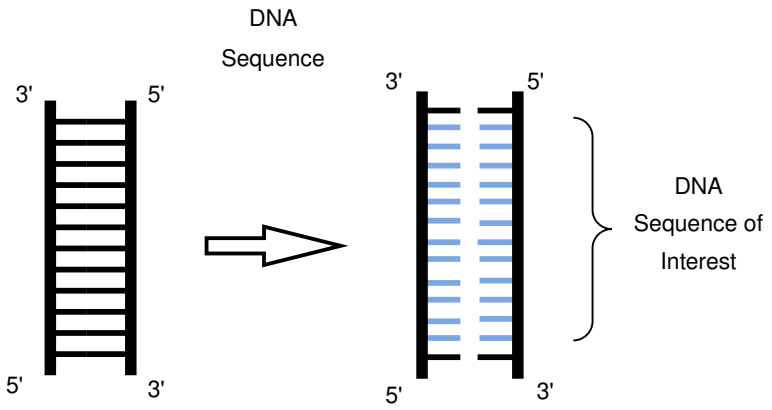

Figure A.1: Visual representation of the important DNA sequence

depicts a DNA sequence, showing the specific DNA sequence that is of particular interest and that we intend to reproduce. Following that, Figure A.2 depicts the PCR method, which is used in microarray analysis to amplify the specific DNA sequence.

PCR is a good method for making multiple copies of a specific portion of DNA from a complex mixture of DNA. Kary Mullis described it as having the ability to "pick the piece of DNA you're interested in and have as much of it as you want.". PCR requires template DNA, primers, nucleotides, and DNA polymerase. The DNA polymerase enzyme is in charge of joining individual nucleotides together to generate the PCR product. The nucleotides are the building blocks of DNA polymerase and are made up of the four bases found in DNA: adenine, thymine, cytosine, and guanine (A, T, C, G). Primers in PCR are small DNA fragments with a preset sequence that complements the target DNA that we want to amplify. They serve as a foundation for DNA polymerase, allowing us to target and amplify the desired DNA product [85].

To perform PCR, we put the components in a test tube or 96-well plate and place them in a heat cycler. The test tubes or glass slides containing the PCR reaction mixture are inserted through perforations in the heat block of this machine. The thermal cycler progressively raises and lowers the block's temperature. Denaturation begins with heating the reaction solution above the melting point of the target DNA's two complementary DNA strands. The strands can now be separated. After that, the temperature is decreased to allow the right primers to bind to the target DNA segments. Hybridization or annealing occurs only when the primers and target DNA are complementary in sequence (for example, A binding to T) [85].

When the temperature is raised again, the DNA polymerase can add nucleotides to the developing DNA strand to extend the primers. With each repetition of these three phases, the quantity of replicated DNA molecules doubles. Because PCR is so sensitive, we only need trace amounts of DNA to make enough copies for analysis [85].

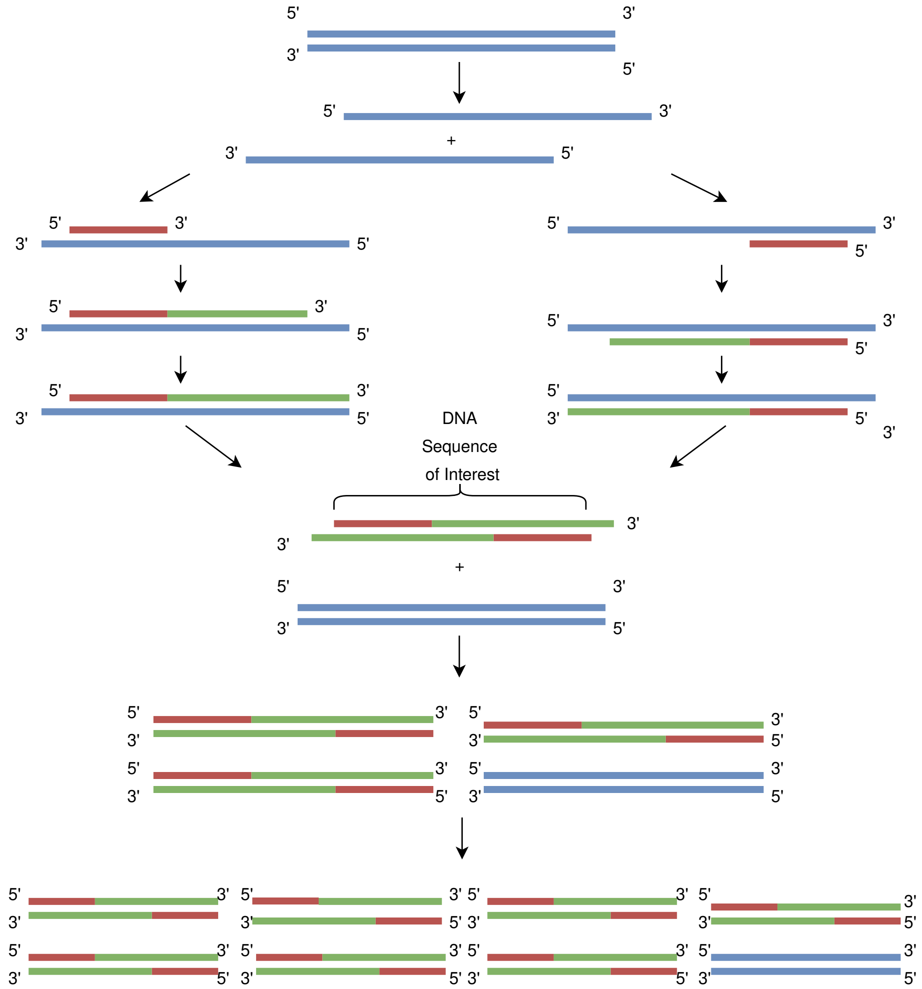

Figure A.2: Visual representation of the PCR process

A.4 Labelling

Many molecular biology techniques, such as microarray analysis and PCR, require labelling. Labelling in microarray analysis entails adding fluorescent or biotinylated nucleotides into cDNA to facilitate detection and quantification of the cDNA during microarray hybridization [86]. PCR labelling, on the other hand, involves the incorporation of fluorescent probes that bind specifically to the target DNA during amplification, allowing for real-time detection and quantification of the amplified DNA [87].

Traditional labelling approaches, such as indirect and two-step labelling, can inject technical noise and variability into the data. Direct labelling approaches have been developed to address these difficulties. Direct labelling eliminates the requirement for amplification, lowering the possibility of amplification bias and producing more accurate and dependable data. The Cy3/Cy5 direct labelling approach, for example, uses reactive Cy3 and Cy5 dyes to identify RNA targets, which then hybridize directly to the microarray probe without amplification. This method enables for direct quantification of RNA targets while reducing the possibility of introducing

technical noise or variability [88].

A.5 Hybridization

The hybridization process involves binding the labelled cDNA to complementary DNA probes on the microarray, which results in the generation of a detectable and quantifiable signal. This procedure is analogous to the hybridization stage used in real-time PCR, in which specific primers are permitted to bind to target DNA segments via annealing or hybridization. The binding between the probes and the target DNA is dependent on their complementary sequences in both circumstances [85].

A.6 Data Collection and Images

There are two distinct approaches to conducting a microarray experiment: one-channel and two-channel. The terms "one" and "two" refer to the number of samples used in the experiment. In the one-channel microarray experiment, there is only one sample we are converting into data. The sample is treated with a binding agent. This procedure aims to determine whether the binding agent effectively binds to the single-stranded DNA. Since the sample is single-stranded DNA there is a possibility that they bind together with the agent. If the genes are expressed, the probe on the glass slide does give a glow depending on the amount of binding agent that actually has bound to the DNA strain. The glass slide is put into a scanner, and each spot on the glass slide is scanned and converted into a numerical value [89]–[91]. In the two-channel approach, two different samples (infected tissue and control sample) are used. Each sample is labelled with a fluorescent binding agent, but the agents are coloured differently for easy differentiation. The subsequent steps are similar to the one-channel approach, where the samples are placed on the glass slide, and the scanner records the emitted light [89], [90], [92]. The datasets used in this thesis belong to the one-channel category and for this reason, we describe this approach in more detail below.

In the case of one-channel microarrays, a single sample is hybridized to each glass slide. The fluorescent signal emitted by each probe on the microarray is evaluated during the hybridization procedure to quantify the amount of binding agent attached to it [89]. If a probe appears black, it indicates that none of the binding agents has bound to the DNA segment at that specific location, implying no gene expression (see Figure A.3 for a visual representation). The microarray glass slide is washed before to scanning to eliminate any leftover binding agents that did not bind to the single-stranded DNA. The emitted fluorescent signals from the microarray data are then recorded using a scanner [93].

Each probe on the microarray corresponds to a unique DNA sequence representing a specific gene. During hybridization, the intensity of the fluorescent signal emitted from each probe reflects the overall brightness and indicates the level of gene expression. The standard deviation of the fluorescent signal provides valuable information

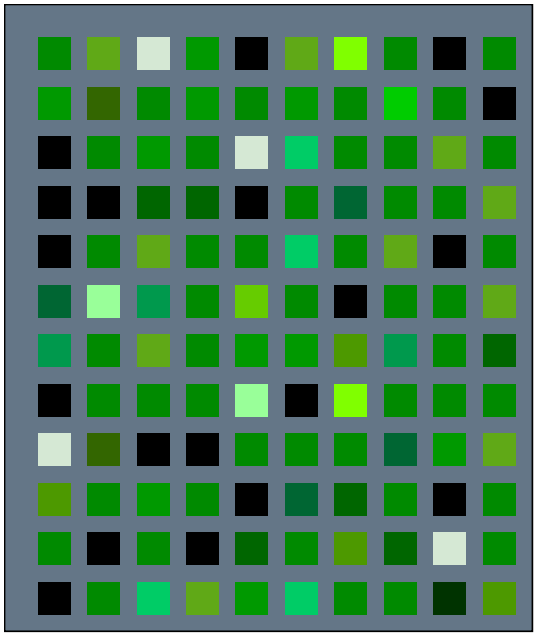

Figure A.3: A visual representation of how an image after hybridization is represented for a one-channel microarray experiment.

about the variation in signal strength among different probes, aiding in the measurement process. Two crucial measurements obtained during scanning include the intensity of the emitted light and the standard deviation of the signal. By knowing the origin of each row on the glass slide, whether it pertains to a control or cancer sample, further analysis and interpretation of the data obtained from the scanning process can be conducted [90], [91].

A.7 Data Analysis

In microarray data processing, normalization procedures are used to normalize intensity levels, assuring comparability across different microarrays within an experiment. This standardization allows for meaningful comparisons of samples [90]. Gene expression data analysis includes processing and discovering patterns and relationships between genes. The identification of a group of relevant genes that are most informative for a certain research issue is a critical step in microarray data analysis. In this sense, machine learning approaches, particularly classification algorithms, have grown in favour of the analysis of microarray data. These algorithms may effectively identify samples and discover genes important for disease diagnosis and prognosis [94].

B

Appendix 2

B.1 Methods

This section in the appendix provides more in-depth details regarding the classifiers used in this study.

B.1.1 Naïve Bayes

Naïve Bayes is a probabilistic classification algorithm commonly employed in text classification, spam filtering, and sentiment analysis [42], [43]. It utilizes Bayes theorem to calculate the probability of a hypothesis based on evidence. The Naïve Bayes process involves two key steps: training and prediction, detailed by algorithms 1 and 2.

The basic idea behind Naïve Bayes is to calculate the probability of a given data point belonging to a particular class, based on the probabilities of the features of the data point [56], [57]. Naïve Bayes assumes that the features are independent of each other, which makes the calculations much simpler.

To calculate the probability of a class label given a new data point x_new, Naïve Bayes uses Bayes theorem. According to Bayes theorem, the probability of a class label y given a data point x is calculated based on the probability of observing the data point x given that it belongs to class $y(P(x|y))$, the prior probability of class $y(P(y))$, and the probability of observing the data point $x(P(x))$.

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

$$P(x_{\text{new},1}, x_{\text{new},2}, ..., x_{\text{new},m}|y) = \prod_{j=1}^{m} P(x_{\text{new},j}|y)$$

Naïve Bayes uses a probability distribution to model each feature, depending on its type. For example, it can use a Gaussian distribution for continuous features and a categorical distribution for discrete features. To calculate the prior probability of class $y(P(y))$, Naïve Bayes simply counts the number of data points in each class and divides them by the total number of data points. To calculate the probability

of observing the data point $x(P(x))$, Naïve Bayes uses the law of total probability [56], [57]. It calculates the sum of the products of the probability of observing the new data point x_new given that it belongs to each class y_i and the prior probability of class y_i.

$$P(x_{\text{new}}) = \sum_{i=1}^{k} P(x_{\text{new}}|y_i)P(y_i)$$

where $P(x_{\text{new}}|y_i)$ is the probability of observing the new data point x_{new} given that it belongs to class y_i, and $P(y_i)$ is the prior probability of class y_i. Once Naïve Bayes has calculated the probabilities for each class label, it chooses the one with the highest probability as the predicted class label for the new data point x_{new} [56], [57].

Algorithm 1 Naïve Bayes Training [56]

Input: Training dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$ consisting of m samples, number of classes k

Output: Prior probabilities $P(y_i)$ for each class i and likelihood probabilities $P(x_j|y_i)$ for each feature j and class i

for $i \leftarrow 1$ **to** k **do**
  Let D_i be the set of all samples in D with class y_i Compute prior probability $P(y_i) = |D_i|/m$ **for** *each feature* j **do**
    Compute likelihood $P(x_j|y_i)$ using the training samples in D_i (e.g., Gaussian distribution for continuous features, multinomial distribution for discrete features)
  end
end

Algorithm 2 Naïve Bayes Prediction [56]

Input: Trained model: prior probabilities $P(y_i)$ and likelihood probabilities $P(x_j|y_i)$ for each feature j and class i, new sample $x = (x_1, x_2, ..., x_n)$ to be classified

Output: Predicted class for the new sample x

for $i \leftarrow 1$ **to** k **do**
  Compute $P(y_i|x) = P(y_i) * P(x_1|y_i) * P(x_2|y_i) * ... * P(x_n|y_i)$ using the trained model and the new sample x
end
Choose the class i with the highest posterior probability $P(y_i|x)$ and output it as the predicted class for the new sample x;

B.1.2 K-Nearest Neighbours

The k-Nearest Neighbors (k-NN) algorithm is a straightforward classification method widely recognized for its simplicity. It operates based on proximity, predicting outcomes by identifying the k nearest neighbors to a specific data point. A crucial

aspect of k-NN involves selecting a distance metric, such as Euclidean or Manhattan distance, to measure the closeness or dissimilarity between data points in a feature space [57].

Neither explicit model training nor parameter estimates are used in the k-NN approach. Instead, it uses previously stored training data to make predictions based on their closeness to labeled occurrences. As a result, it is classified as a lazy learning algorithm (see Algorithm 3 for the pseudo-code) [57]. To produce a forecast with the k-NN algorithm, we first choose a value for k, which denotes the number of nearest neighbors to take into account. Given a new data point, the method determines the k nearest neighbors using the distance metric of choice and checks the class labels of these neighbors. If the majority of the neighbors belong to the same class, the algorithm labels the new data point with that class label [57].

It is vital to note that the value chosen for k has an effect on the algorithm's performance. A lower value of k, such as $k = 1$, may result in more flexible decision boundaries, but it is more susceptible to noise in the data. A bigger value of k, on the other hand, may create smoother decision limits but may also muddy the distinctions between different classes [57].

Algorithm 3 k-Nearest Neighbors (k-NN) Algorithm

Input: Training dataset $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is the i-th input vector and y_i is the corresponding target output (-1 or 1)

Output: Predicted class for a new input $\mathbf{x}_{\text{new}}$

1 Choose the number of neighbors k **foreach** *new input* $\mathbf{x}_{new}$ **do**

2 Calculate the distance between $\mathbf{x}_{\text{new}}$ and each training sample $\mathbf{x}_i$ Select the k nearest neighbors based on the smallest distances Assign the predicted class for $\mathbf{x}_{\text{new}}$ based on majority voting among the k neighbors

3 **end**

4 **return** *Predicted class for* $\mathbf{x}_{new}$

B.1.3 Decision Tree

A Decision Tree is a hierarchical structure used for classification and regression. It divides data into subsets based on feature values, selecting splits that maximize information gain (reduction in uncertainty) using metrics like Gini impurity or entropy. The algorithm explores all possible splits for each feature, repeating recursively until a stopping criterion, like maximum depth or minimum samples in a leaf node, is met [56], [57].

As shown in the equations below, to calculate the information gain for a split, the algorithm subtracts the weighted average of child entropies from the parent entropy. The parent entropy is calculated by summing the negative product of the proportion of each class label and the logarithm of the proportion (in base 2) for all the labels in the parent node. The child entropies are calculated similarly, but weighted by the proportion of samples in each child node. The Gini impurity can also be calculated using a similar method, with the entropy term replaced by the Gini impurity [56],

[57].

$$\text{Information Gain} = \text{Parent Entropy} - \text{Weighted Average of Child Entropies}$$

$$\text{Parent Entropy} = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Algorithm 4 describes the process for Decision Trees. The algorithm recursively splits the data into subsets based on the most informative feature, chosen using information gain calculated using metrics such as the Gini impurity or entropy. The algorithm considers all possible splits for each feature and chooses the one that maximizes the information gain, repeating the process recursively until a stopping criterion is met, such as a maximum depth or a minimum number of samples in a leaf node. To calculate the information gain for a split, the algorithm subtracts the weighted average of child entropies from the parent entropy, with both the parent and child entropies calculated using the proportion of samples in each class label. The resulting tree can be used for both classification and regression problems.

Algorithm 4 Decision Tree

Input: Training set S, sample x to be classified
Output: Prediction for x

```
Function DecisionTree(S, x):
    if stopping criteria are met then
        return leaf node with prediction
    end
    f, t ← choose splitting criterion based on S; S_L, S_R ← split S based on f and t;
     if x_f < t then
        return DecisionTree(S_L, x)
    end
    else
        return DecisionTree(S_R, x)
    end
```

B.1.4 Support Vector Machine

Support Vector Machines (SVM) is a powerful machine learning algorithm that seeks to find the optimal hyperplane to separate two classes in input data. It maximizes the distance between this hyperplane and the nearest data points from each class, known as support vectors. The hyperplane is defined as $w^T x + b = 0$, where w is the weight vector, x is the input vector, and b is the bias term. SVM uses a cost function to minimize misclassifications and maximize the margin, typically solved through convex optimization techniques like gradient descent [56], [57].

The cost function for SVM is the hinge loss function, given by:

$$L(w,b) = \frac{1}{n}\sum_{i=1}^{n}\max\Big(0, 1 - y_i\big(w^T x_i + b\big)\Big) + \frac{\alpha}{2}||w||^2$$

Algorithm 5 SVM Training [56]

Input: Training set $S = (x_1, y_1), ..., (x_n, y_n)$, where x_i is the i-th input vector and y_i is the corresponding target output (-1 or 1)

Output: Weight vector w and bias term b

Initialize weight vector w and bias term b to zero or a small random value; Set learning rate α and regularization parameter C; **repeat**
  foreach *training example* $(x_i, y_i) \in S$ **do**
    Calculate predicted output $\hat{y}_i = \text{sign}(w^T x_i + b)$; **if** $y_i(w^T x_i + b) \geq 1$ **then**
      Set gradient to $w = w - \alpha C w$ and $b = b$;
    else
      Set gradient to $w = w - \alpha(Cw - y_i x_i)$ and $b = b - \alpha y_i$;
    end
  end
  Update weight vector w and bias term b by subtracting the average gradient;
until *convergence or maximum number of iterations*;
return w *and* b

As shown in Algorithm 5, n is the number of training instances, x_i is the ith input vector, y_i is the corresponding target output, and α is a regularization hyperparameter that controls the tradeoff between maximizing the margin and minimizing the misclassifications. The first term in the cost function is the classification loss, which penalizes misclassifications, and the second term is the regularization term, which encourages a small weight vector. To optimize this cost function, one can use optimization algorithms such as stochastic gradient descent or quadratic programming. The optimal solution for the SVM algorithm is found when the weight vector w and the bias term b are determined such that they minimize the cost function while satisfying the constraints that ensure the hyperplane separates the two classes with the maximum margin. Therefore, SVM is a linear model for binary classification that tries to find the best possible boundary that can separate the two classes in the input data by maximizing the margin between the hyperplane and the nearest data points while minimizing misclassifications using the hinge loss function [56], [57].

The Sequential Minimal Optimization (SMO) algorithm is a popular method for training SVMs. The SMO algorithm is described in Algorithm 6. One advantage of SMO is that it can handle much larger datasets than a normal SVM. This is because it only needs to work with a small subset of the training data at any given time. Another advantage of SMO is that it can handle non-separable datasets by using a soft margin, allowing some misclassifications in the training data for a more robust model [56].

The SMO algorithm starts by initializing the Lagrange multipliers to zero and choosing a kernel function and its parameters. It then computes the kernel matrix and

sets a tolerance threshold for convergence. In each iteration of the algorithm, it selects two Lagrange multipliers to optimize and computes their bounds based on the constraints imposed by the SVM's dual problem. It then uses a closed-form solution to update these multipliers, while ensuring that they satisfy the constraints. The algorithm also updates the bias term and weight vector in each iteration [56].

The SMO algorithm continues to iterate until it converges to a solution, i.e., until all Lagrange multipliers satisfy the Karush-Kuhn-Tucker (KKT) conditions within the specified tolerance. At the end of the algorithm, the support vectors are identified as the feature vectors with non-zero Lagrange multipliers. The bias term is then computed using these support vectors, and the weight vector is updated accordingly. Finally, the SVM classifier function is returned, which can be used to predict the label of new feature vectors [56].

Although SMO has advantages, a normal SVM can be more accurate than SMO on small datasets because it optimizes the objective function globally, whereas SMO only optimizes it locally at each iteration. Additionally, a normal SVM can handle more complex kernels, such as radial basis function (RBF) kernels, which can capture more complex patterns in the data [56].

B.1.5 Cross-validation

One of the main advantages of cross-validation is that it allows us to assess the performance of a model more reliably than using a single train-test split. By using multiple splits of the data, we can get a better sense of how well the model will perform on unseen data. Additionally, cross-validation can help us to tune the hyperparameters of a model, such as the learning rate in a neural network or the number of trees in a random forest. By assessing the model's performance for different hyperparameter values, we can select the best settings [56], [57].

The mathematical definition of the k-fold cross-validation algorithm involves the following: Let D be the dataset with n instances $x_1, x_2, ..., x_n$ and corresponding labels $y_1, y_2, ..., y_n$. Let S be the set of indices $1, 2, ..., n$. Let k be the number of folds. Algorithm 7 summarizes the k-fold cross-validation algorithm. The algorithm ensures that each instance in the dataset is used for testing exactly once and that the model is trained on all the other instances.

Algorithm 6 Sequential Minimal Optimization (SMO) algorithm for training an SVM classifier [56]

Input: Training set $(\mathbf{x}^{(i)}, y^{(i)})_{i=1}^{m}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^n$ and $y^{(i)} \in -1, 1$
Output: The SVM classifier function $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$, where $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$

Initialize the Lagrange multipliers $\alpha_i = 0$ for $i = 1, \ldots, m$;
Choose a kernel function $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ and kernel parameters, if any;
Compute the kernel matrix $\mathbf{K}$, where $\mathbf{K}_{i,j} = K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$;
Set the tolerance threshold ϵ;
while *not converged* **do**
    for $i = 1$ **to** m **do**
        Compute the margin $z^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)} + b$; Compute the prediction $\hat{y}^{(i)} = \text{sign}(z^{(i)})$; Compute the error $\varepsilon^{(i)} = \hat{y}^{(i)} - y^{(i)}$; **if** $(\alpha_i < C$ *and* $y^{(i)}\varepsilon^{(i)} < -\epsilon)$ *or* $(\alpha_i > 0$ *and* $y^{(i)}\varepsilon^{(i)} > \epsilon)$ **then**
            Choose a second Lagrange multiplier α_j uniformly at random from $1, \ldots, m \setminus i$; Compute the kernel $k_{ij} = K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$; Compute the bounds L and H for α_j; **if** $L = H$ **then**
                continue;
            end
            Compute the unclipped new value $\alpha_j^{\text{new,unc}}$ using Equation (12-16) in the book; Clip the new value α_j^{new} to be within the bounds L and H; Compute the change in α_j as $\Delta\alpha_j = \alpha_j^{\text{new}} - \alpha_j$; Compute the change in α_i as $\Delta\alpha_i = -y^{(i)}y^{(j)}\Delta\alpha_j$; Update the Lagrange multipliers $\alpha_i \leftarrow \alpha_i + \Delta\alpha_i$ and $\alpha_j \leftarrow \alpha_j^{\text{new}}$; Compute the bias term b using Equation (12-7) in the book; Update the weight vector $\mathbf{w} \leftarrow \sum_{i=1}^{m} \alpha_i y^{(i)} mathbf x^{(i)}$;
        else
    end
    if *all Lagrange multipliers satisfy the KKT conditions within the tolerance* ϵ **then**
        break;
    end
end
Compute the set of support vectors $\mathbf{x}^{(i)} \mid \alpha_i > 0$;
Compute the bias term b using Equation (12-5) in the book;
return The SVM classifier function $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$;

Algorithm 7 K-fold Cross-Validation Algorithm [56]

Input: Dataset D with n instances
Output: Performance estimate of a machine learning model
Data: Number of folds k

```
1 Shuffle the dataset randomly;
  Split the dataset into k groups;
  for each unique group do
2     Take the group as a hold out or test data set;
      Take the remaining groups as a training data set;
      Fit a model on the training set and evaluate it on the test set;
      Retain the evaluation score and discard the model;
3 end
4 Summarize the skill of the model using the sample of model evaluation scores;
```

C

Appendix 3

C.1 Sensitivity analysis

This section shows the exact values represented in the graphs for the sensitivity analysis.

C.1.1 Stall limit

Table C.1: Performance of SPFSR parameter: stall limit

Classifier	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
DT	74.79	73.33	71.59	73.46	75.85	73.51	74.10	75.28
NB	80.03	81.21	81.31	81.03	82.31	79.00	81.03	81.03
KNN	75.13	72.77	73.97	76.54	75.20	74.05	73.59	74.41
SVM	82.28	80.03	81.64	81.26	81.31	80.05	81.92	80.67

C.1.2 Max iterations

Table C.2: Performance of SPFSR parameter: max iterations

Classifier	**70**	**80**	**90**	**100**	**110**	**120**	**130**
DT	74.59	73.97	76.49	75.85	74.90	75.51	74.21
NB	80.38	81.03	81.03	81.03	81.03	81.03	80.69
KNN	74.87	74.56	75.20	75.20	75.54	75.23	76.46
SVM	80.08	80.05	80.36	81.31	80.97	80.03	80.03

C.1.3 Number of Gradient Averaging

Table C.3: Performance of SPFSR parameter: number of gradients average

Classifier	1	2	3	4	5	6	7
DT	73.13	74.23	70.21	75.85	75.15	75.13	71.56
NB	81.67	79.46	81.62	81.03	80.00	82.31	80.95
KNN	74.08	78.44	74.28	75.20	77.05	76.18	76.99
SVM	83.56	80.67	79.36	81.31	80.95	81.03	81.90

C.1.4 Hot start range: RFI

Table C.4: Performance of hot_start_range - RFI

Classifier	0.1	0.2	0.5	1
KNN	76.30	76.88	76.68	77.16
DT	74.91	76.42	74.26	73.47
NB	81.37	81.33	80.35	81.33
SVM	81.24	80.90	81.88	82.62

C.1.5 Hot start range: FScore

Table C.5: Performance of hot_start_range - FScore

Classifier	0.1	0.2	0.5	1
KNN	75.42	75.76	75.26	76.15
DT	77.18	77.00	76.08	74.66
NB	79.67	79.17	80.05	79.86
SVM	81.14	81.63	81.88	82.62

C.1.6 Hot start range: InfoGain

Table C.6: Performance of hot_start_range - InfoGain

Classifier	0.1	0.2	0.5	1
KNN	76.30	75.76	75.63	74.92
DT	74.91	74.06	74.22	74.44
NB	78.23	79.83	79.89	78.90
SVM	80.72	81.59	81.65	82.71

D

Appendix 4

D.1 Result plots

This section of the appendix contains the results for the remaining plots.

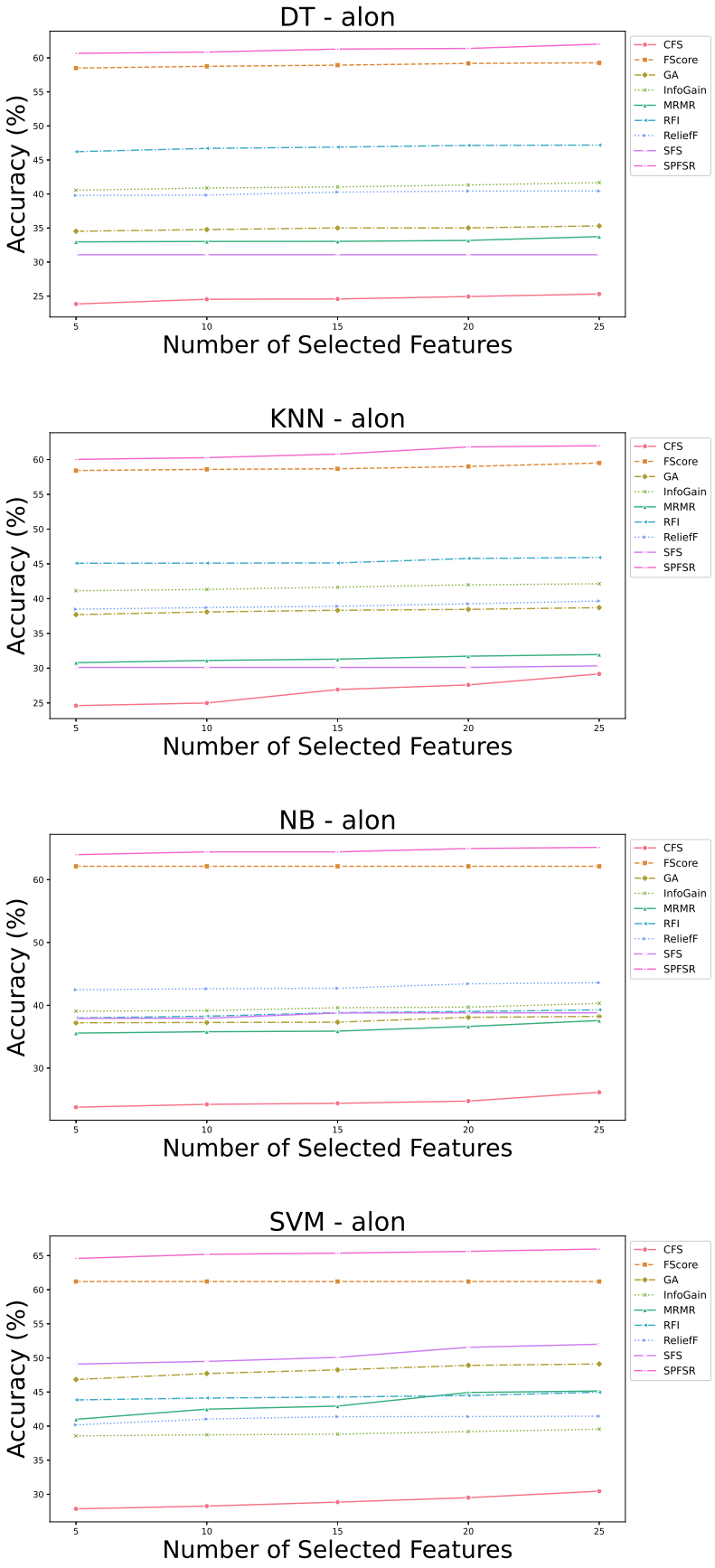

Figure D.1: The average accuracy - Alon

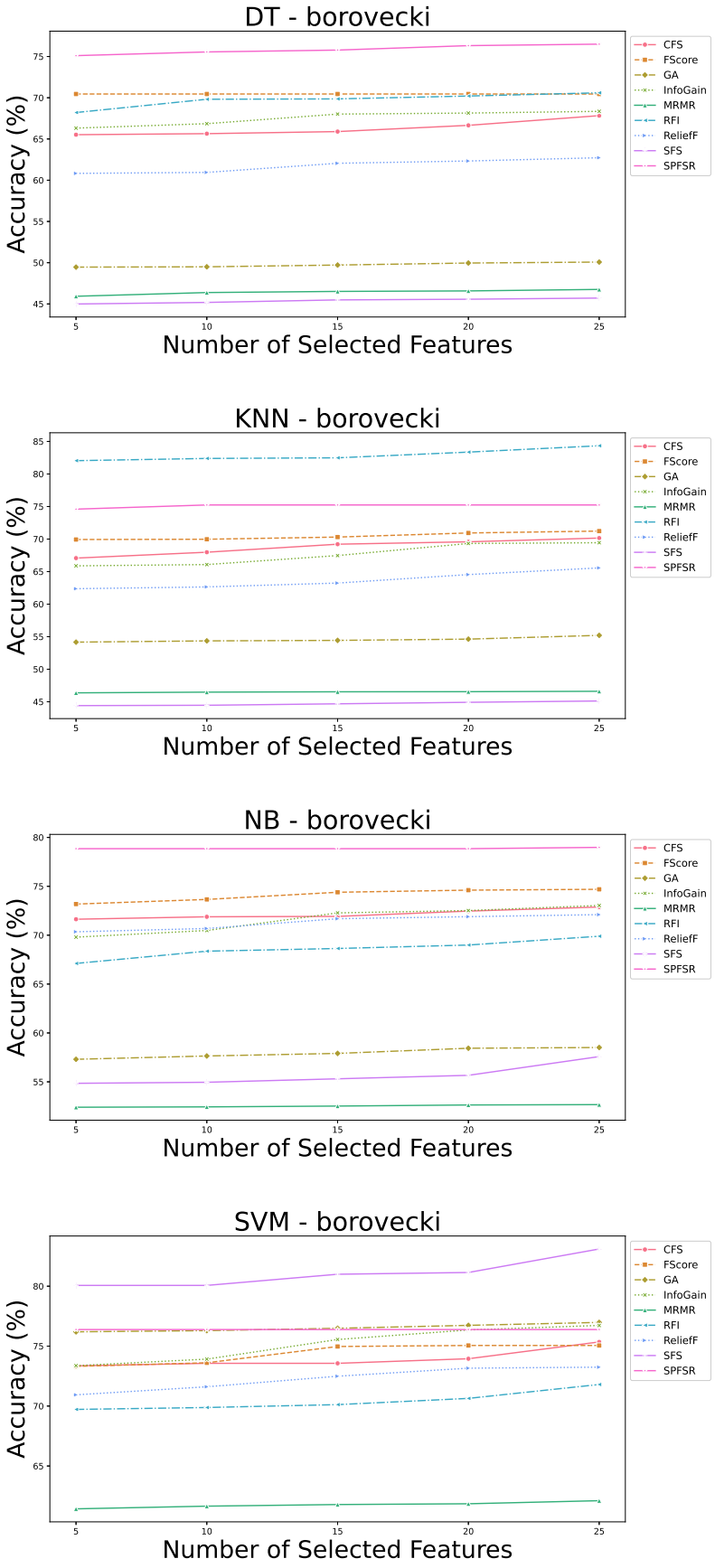

Figure D.2: The average accuracy - Boroveki

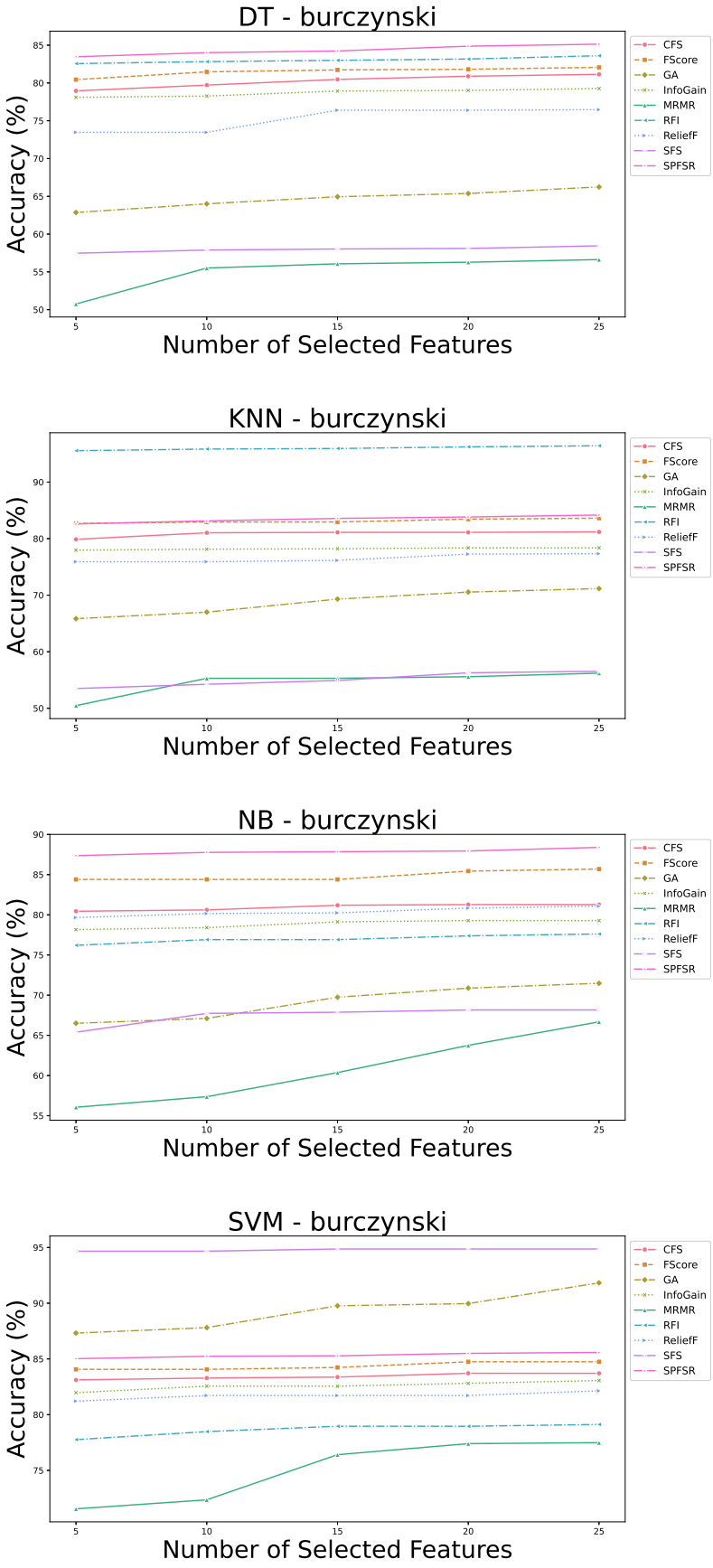

Figure D.3: The average accuracy - burczynski

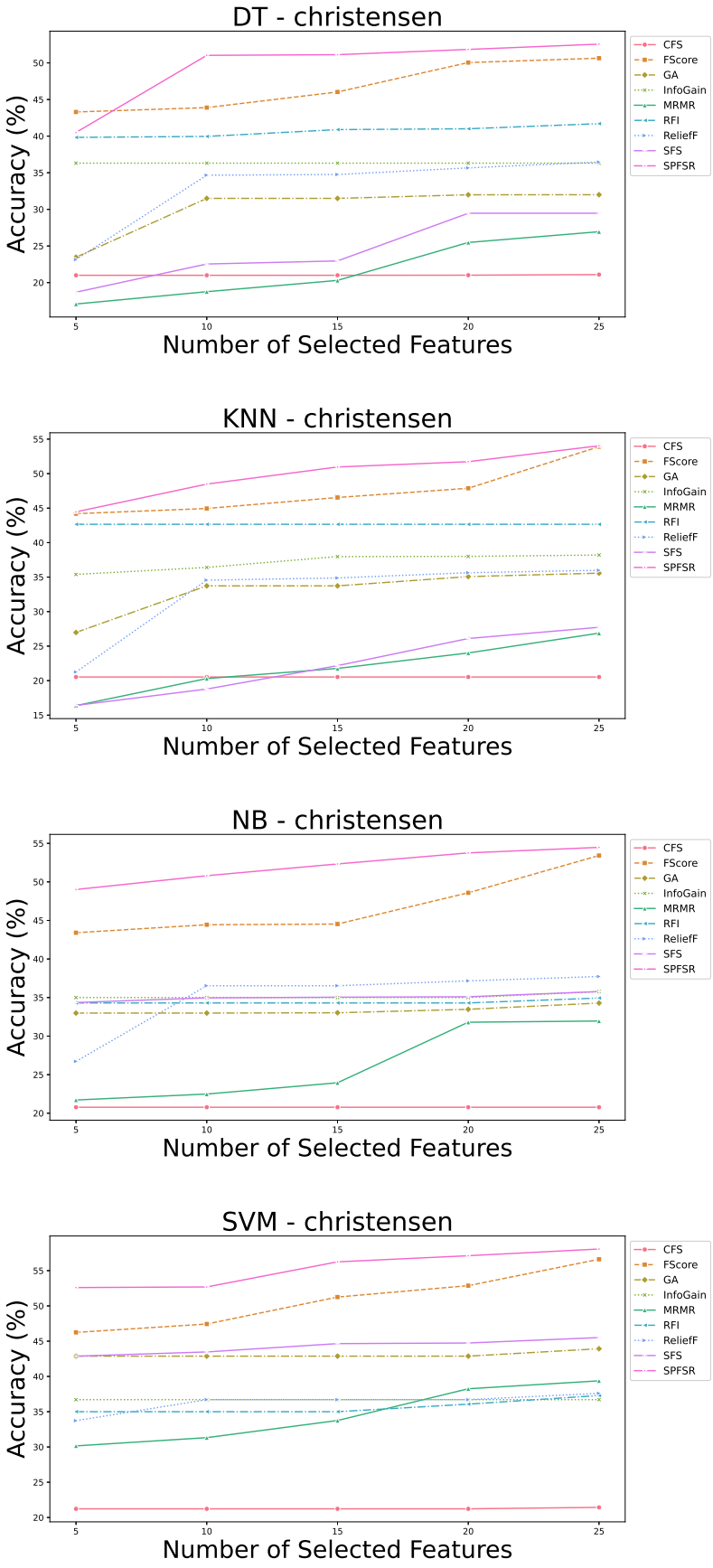

Figure D.4: The average accuracy - Christensen

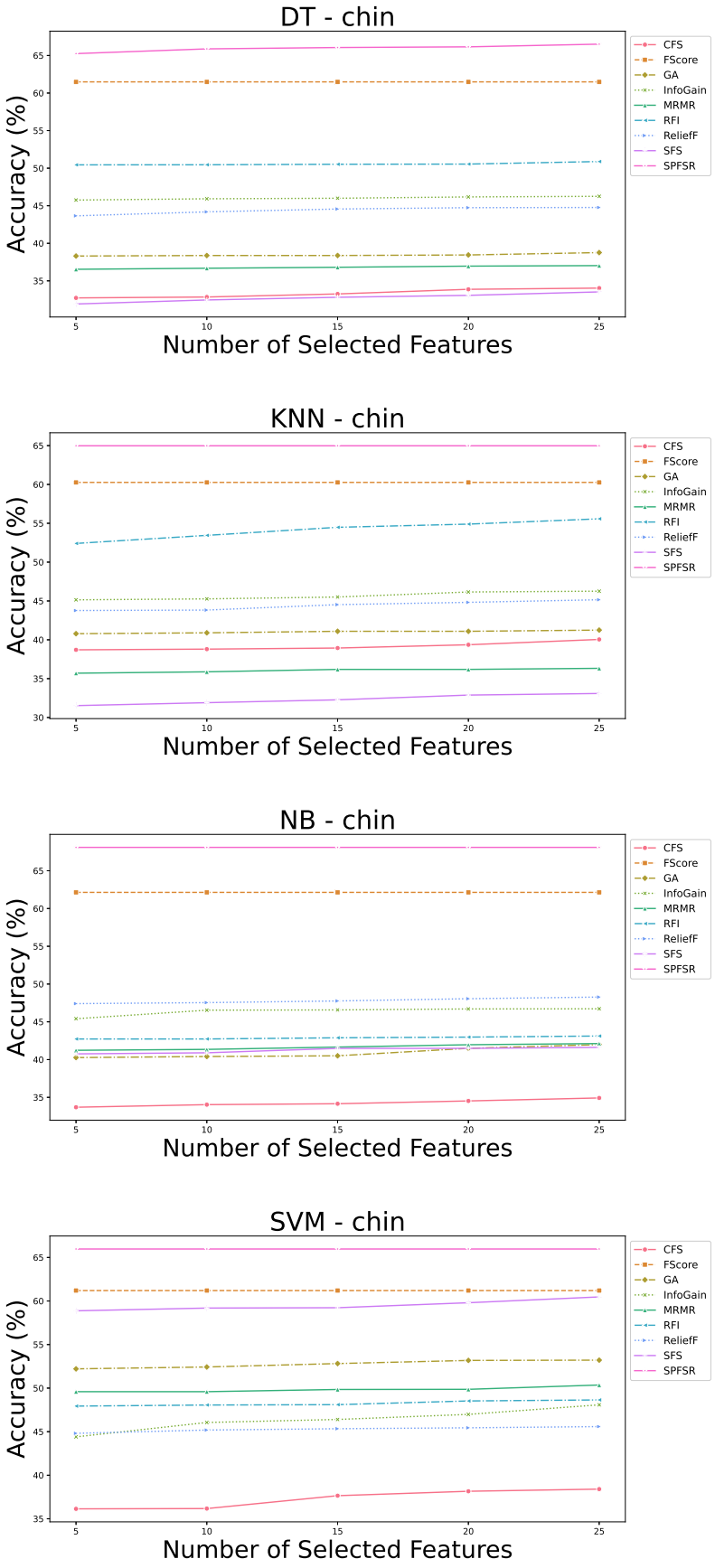

Figure D.5: The average accuracy - Chin

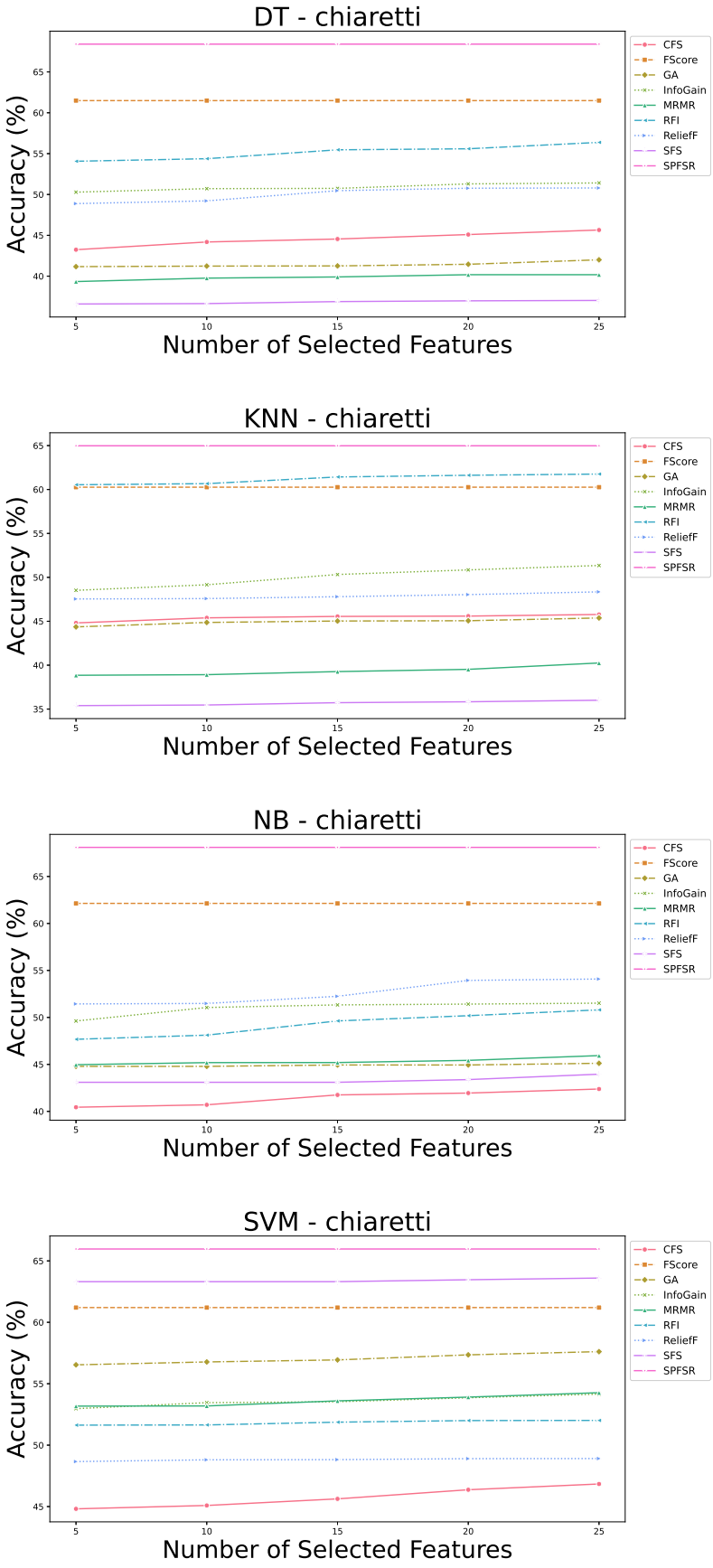

Figure D.6: The average accuracy - Chiaretti

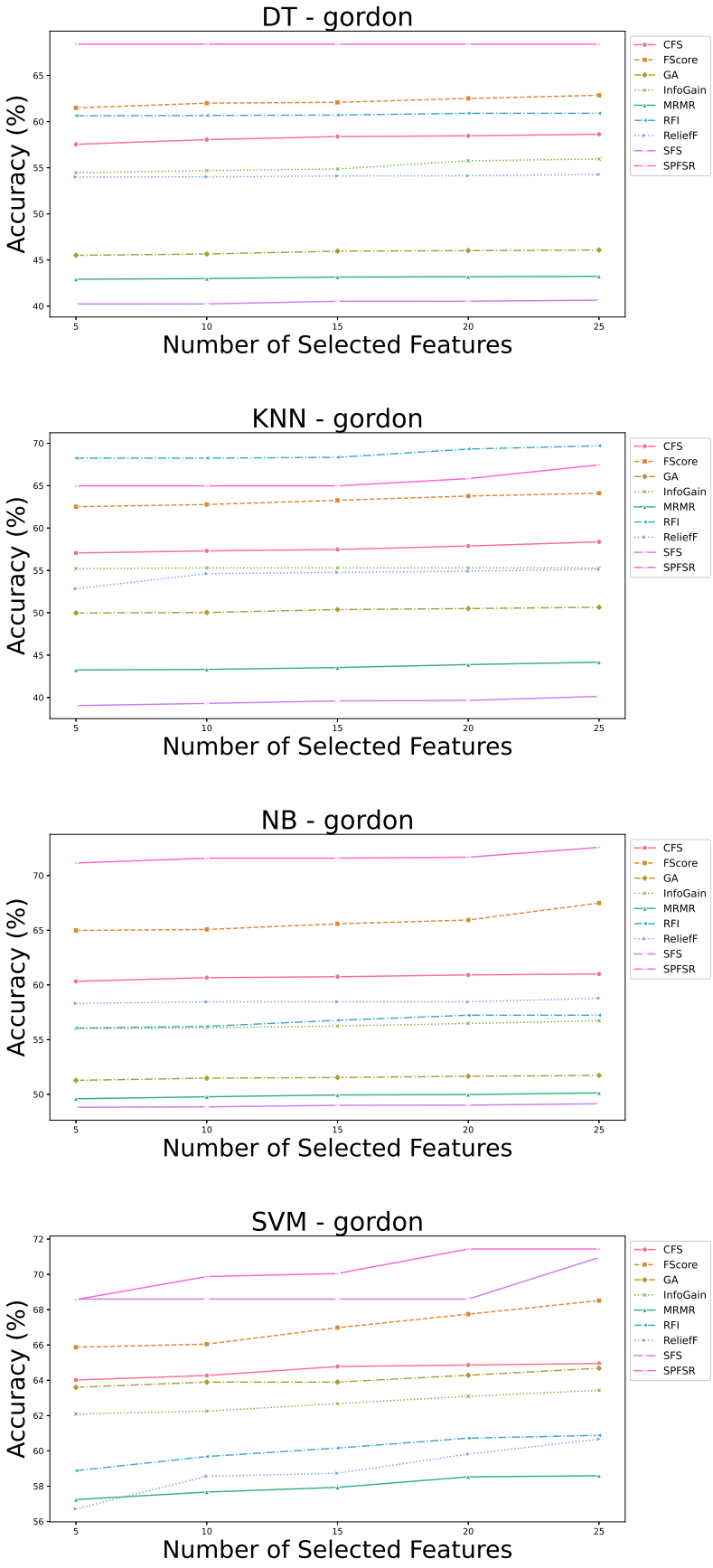

Figure D.7: The average accuracy - Gordon

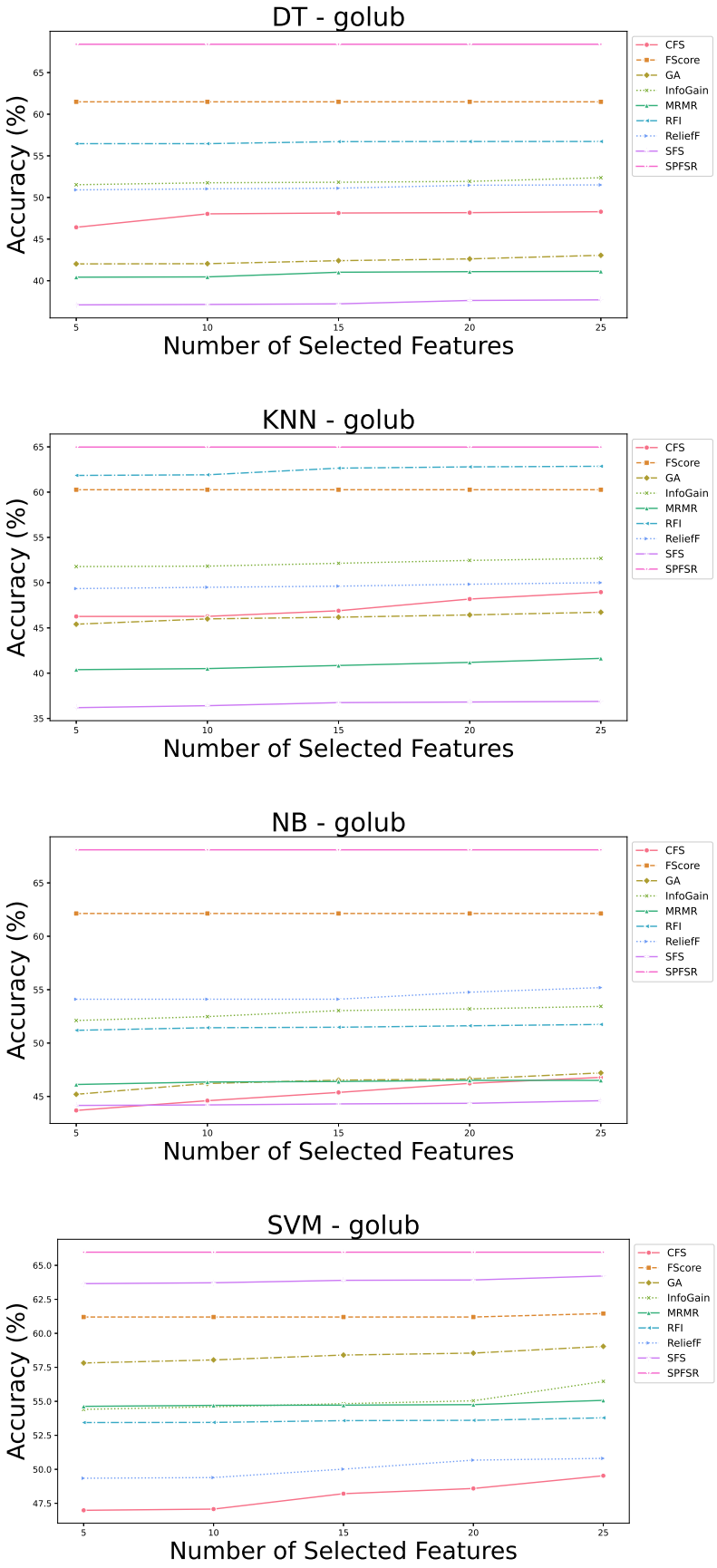

Figure D.8: The average accuracy - Golub

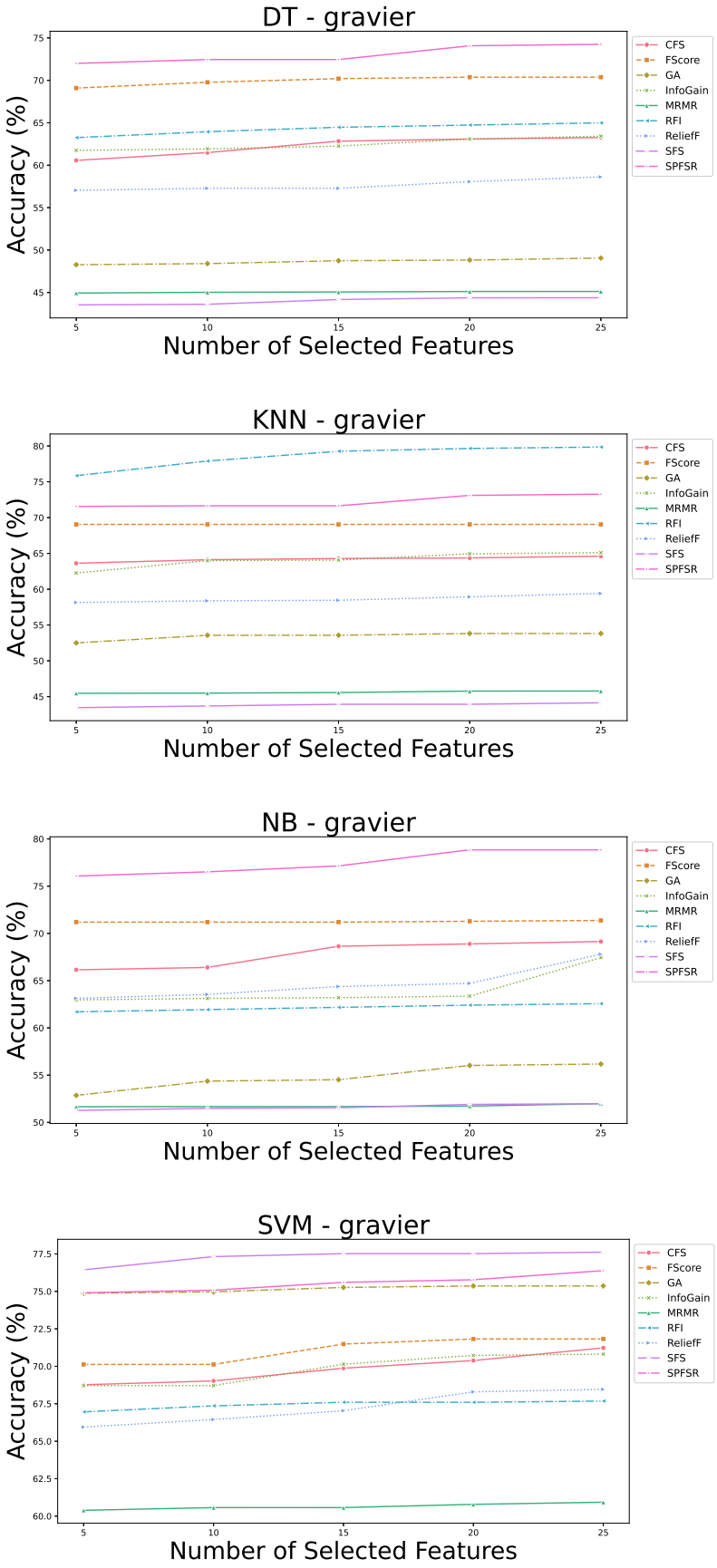

Figure D.9: The average accuracy - Gravier

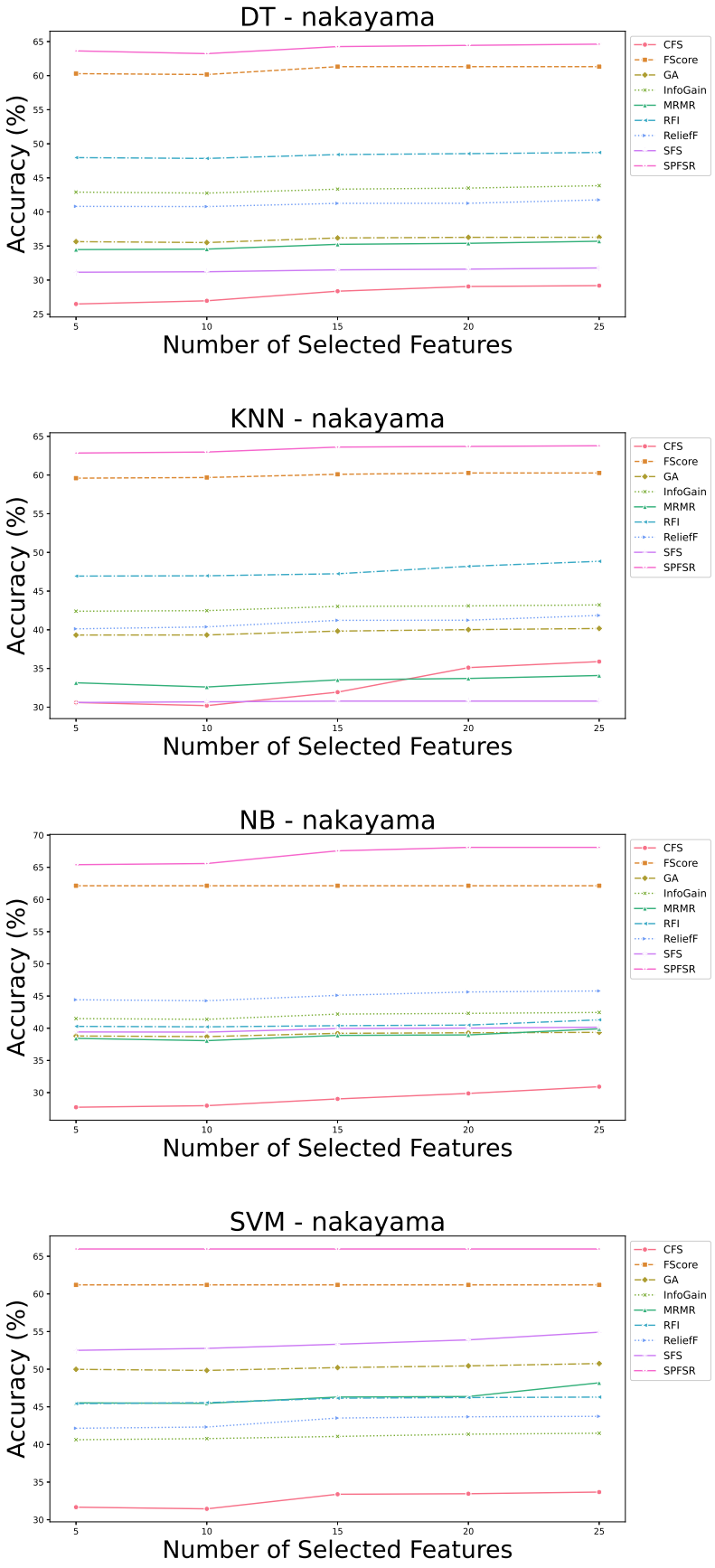

Figure D.10: The average accuracy - Nakayama

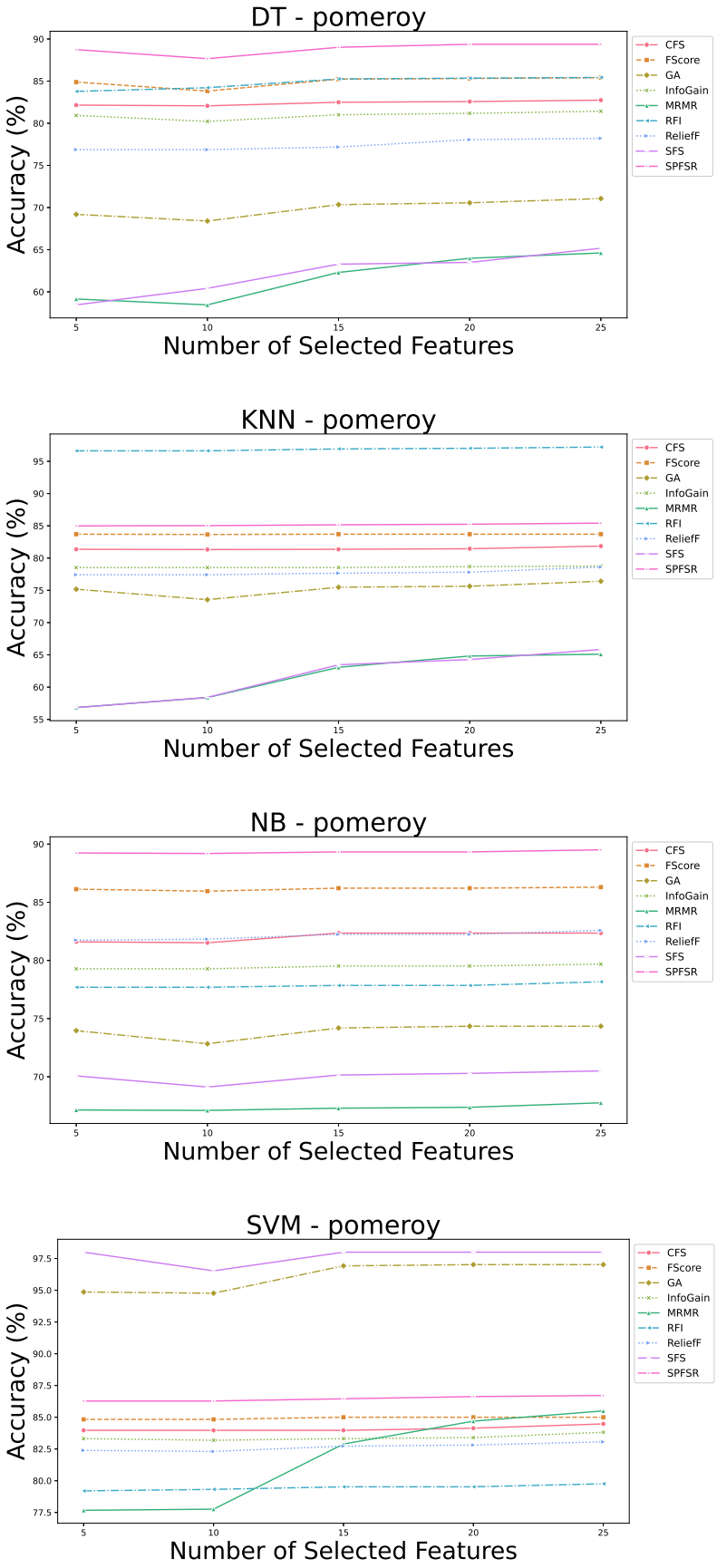

Figure D.11: The average accuracy - Pomeroy

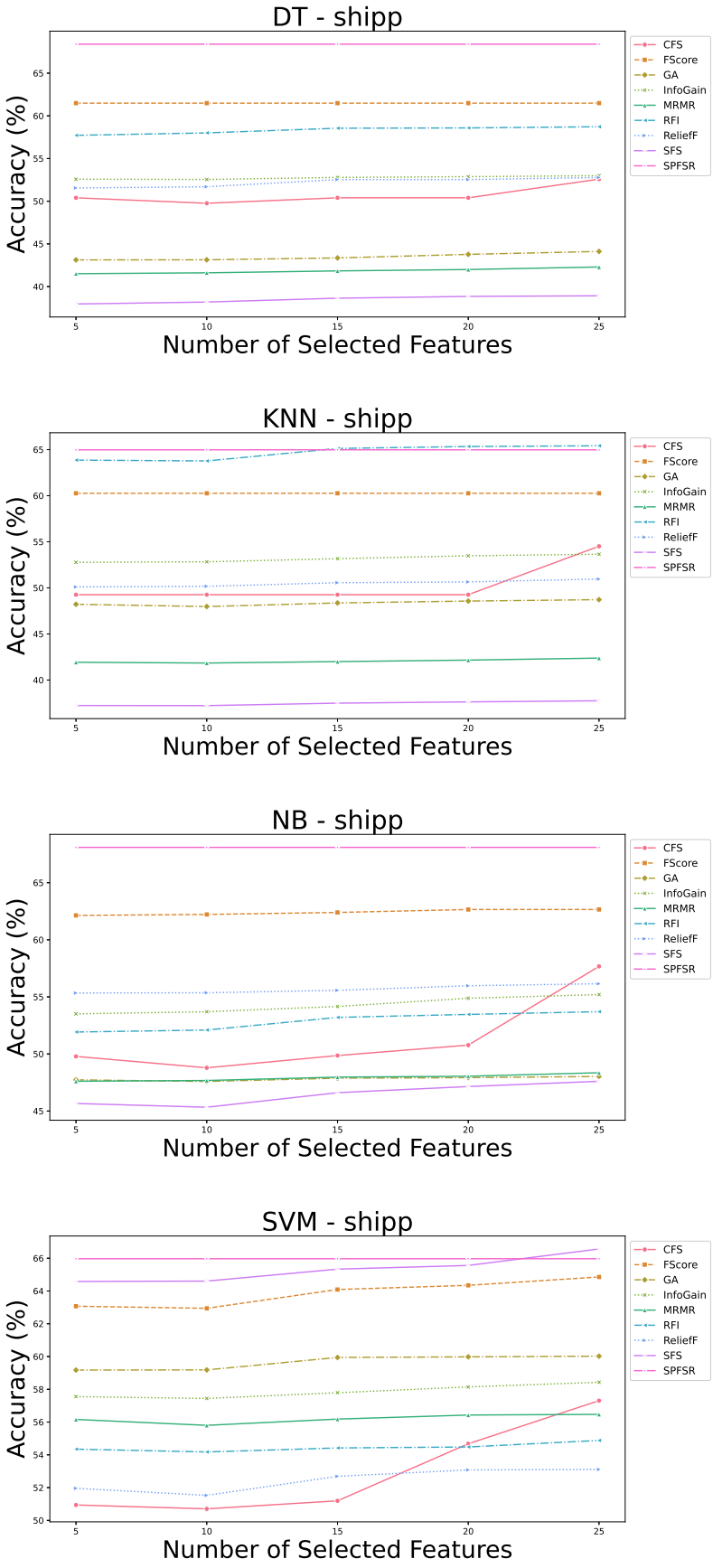

Figure D.12: The average accuracy - Shipp

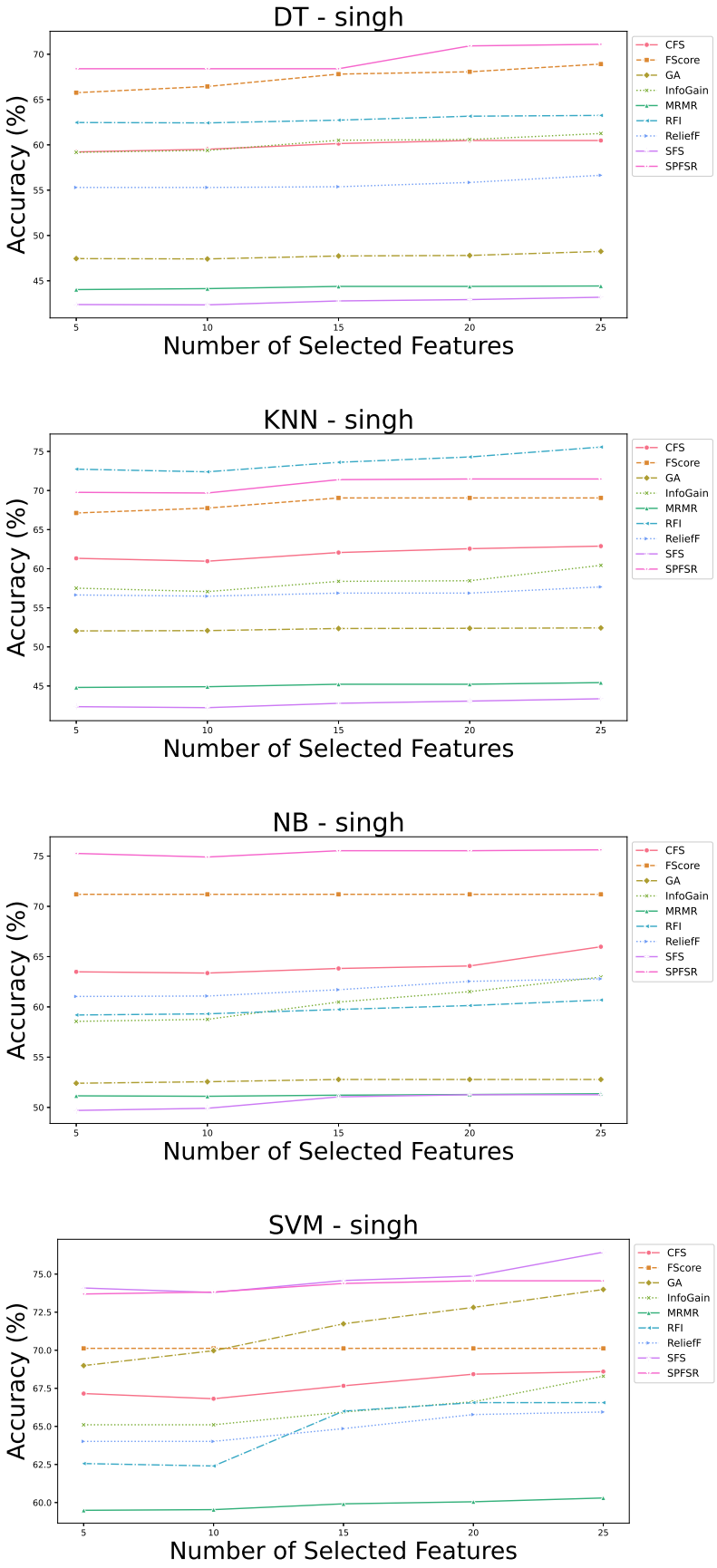

Figure D.13: The average accuracy - Singh

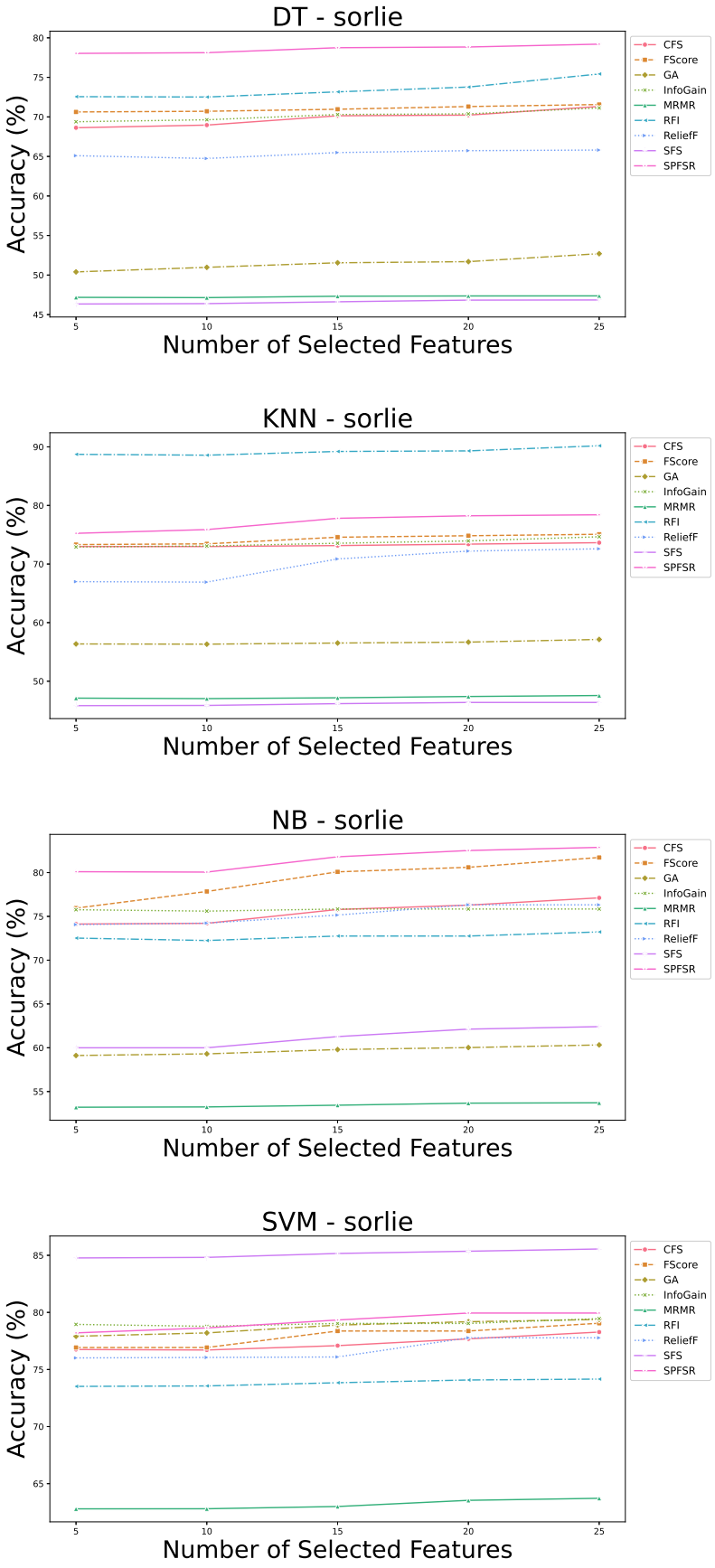

Figure D.14: The average accuracy - Sorlie

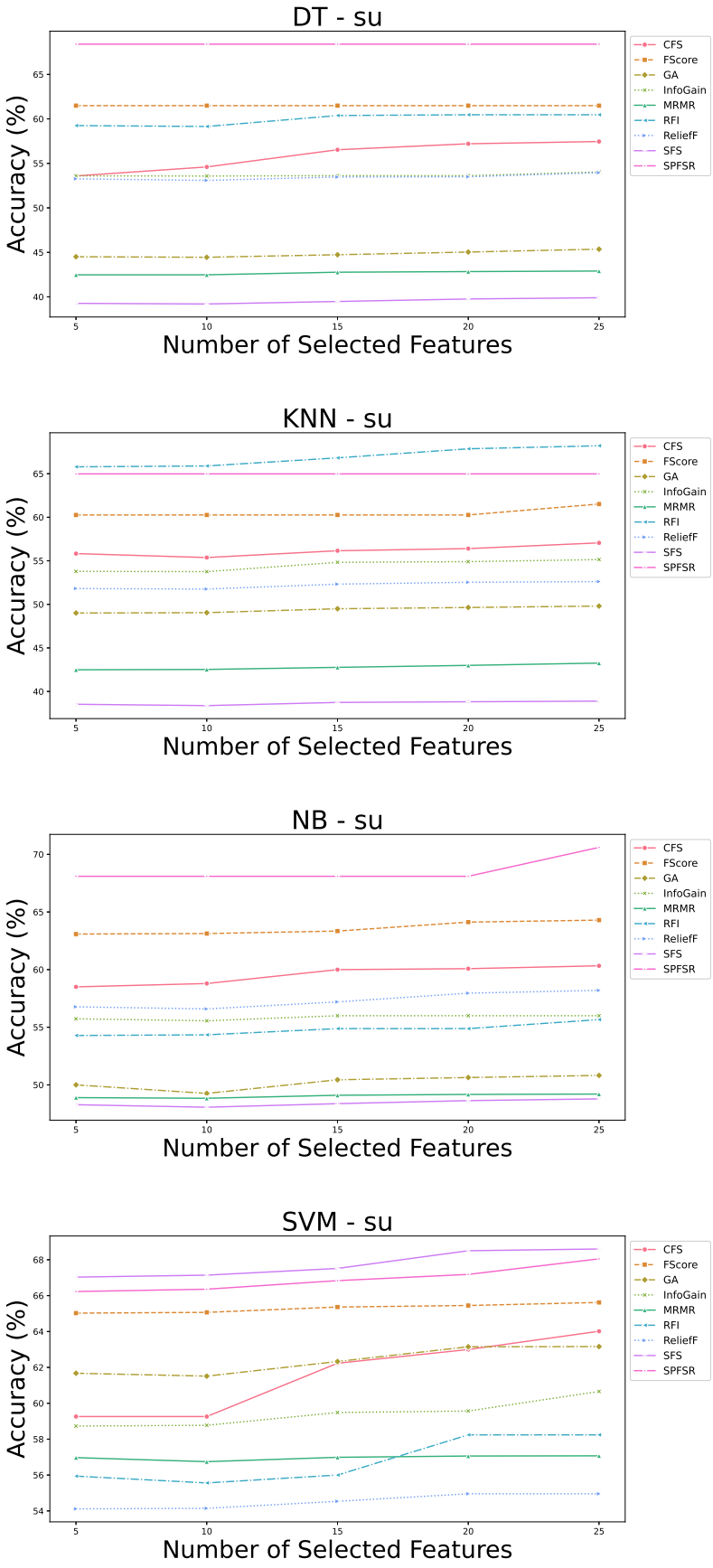

Figure D.15: The average accuracy - Su

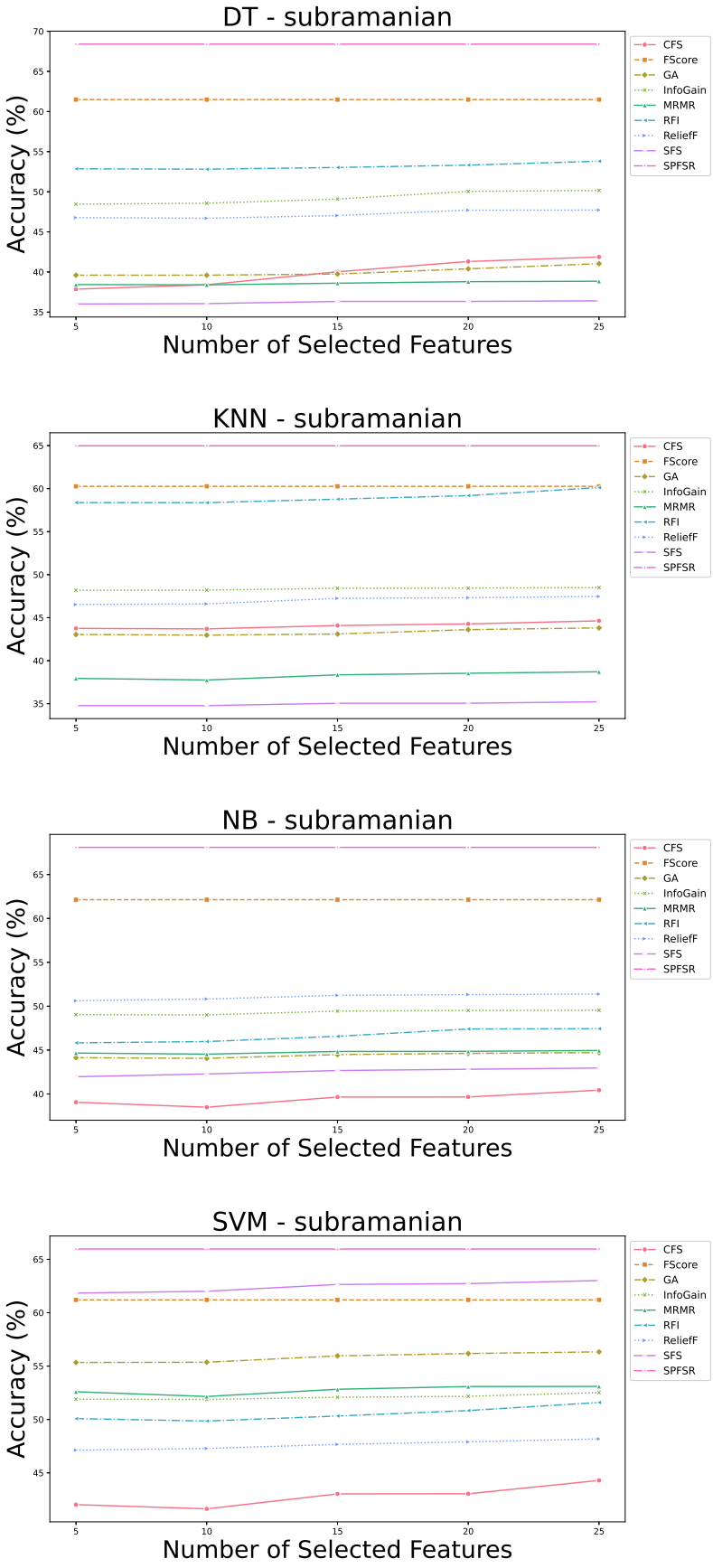

Figure D.16: The average accuracy - Subramanian

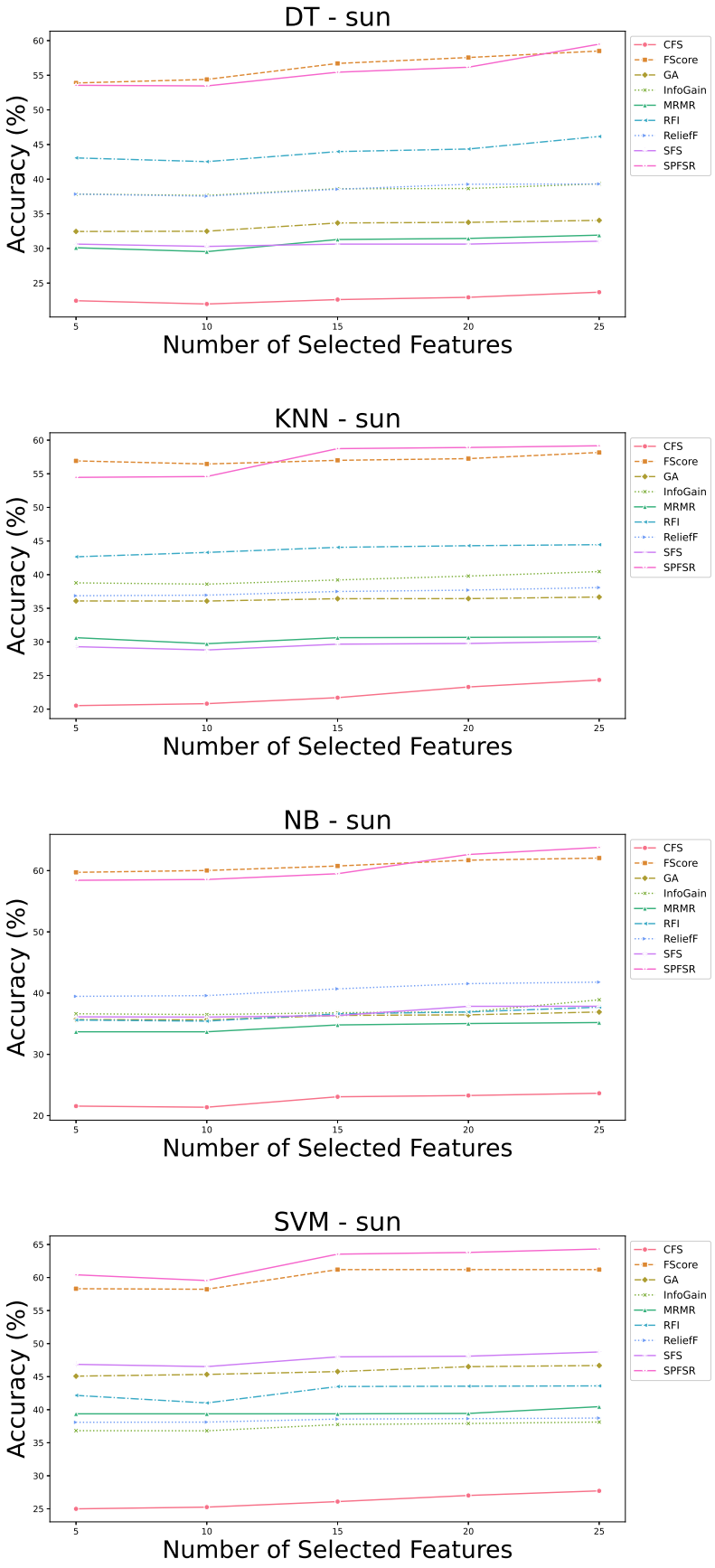

Figure D.17: The average accuracy - Sun

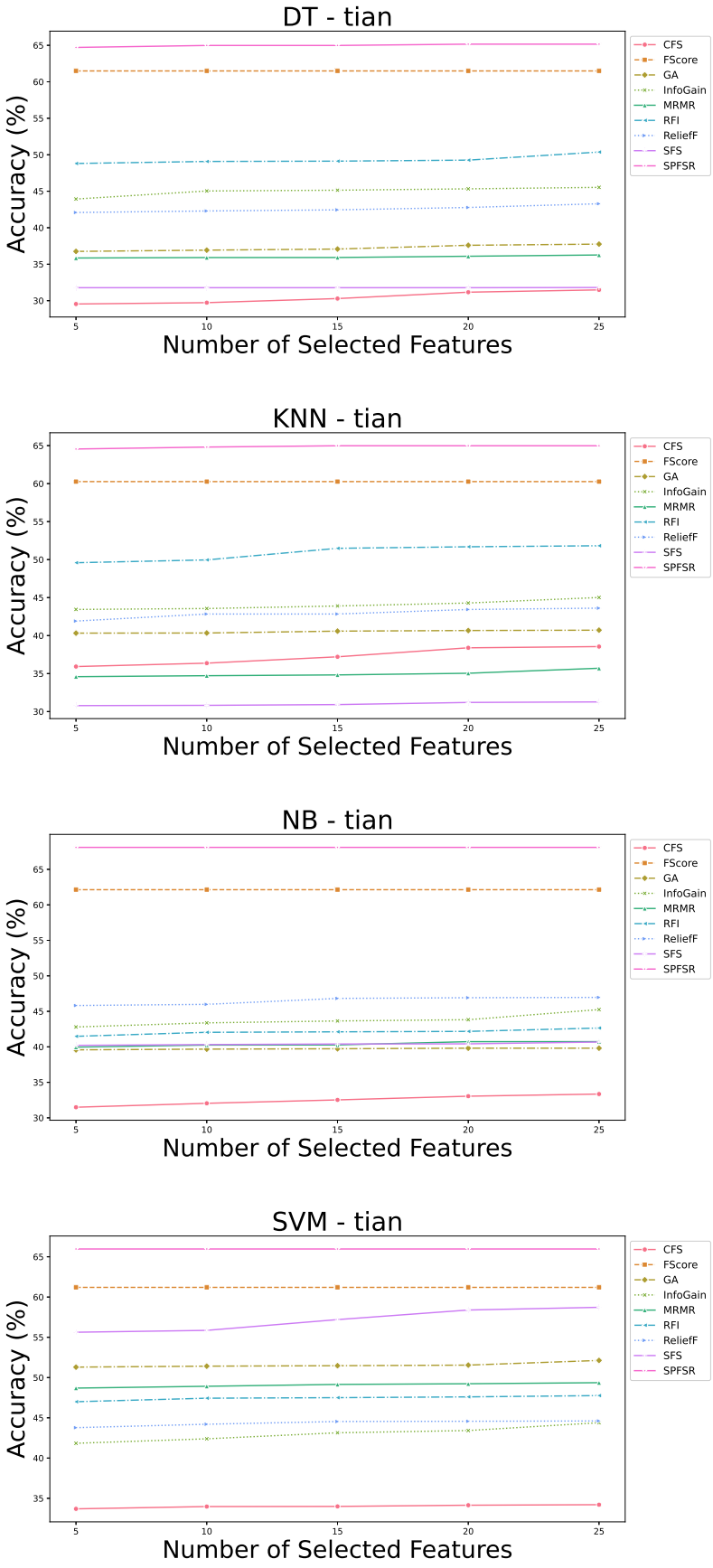

Figure D.18: The average accuracy - Tian

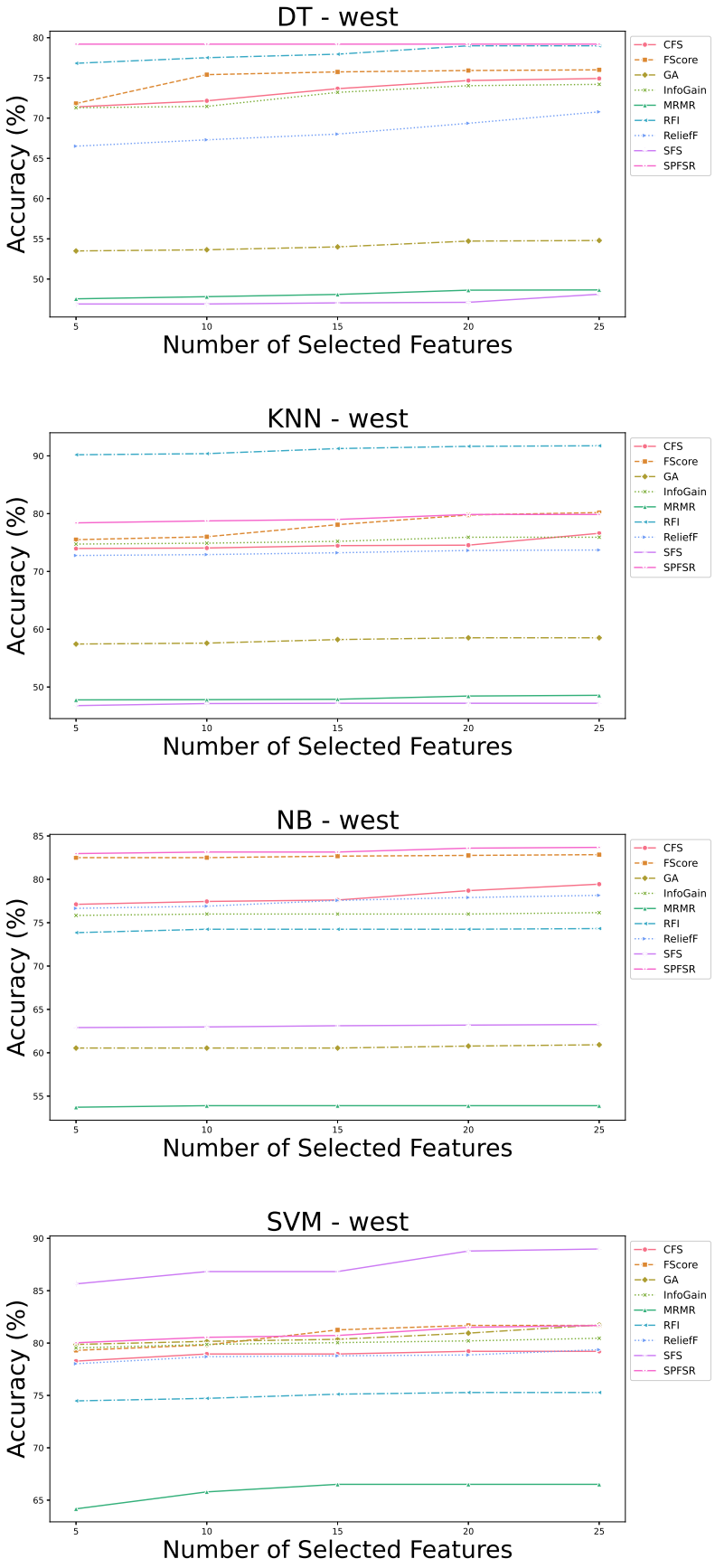

Figure D.19: The average accuracy - West

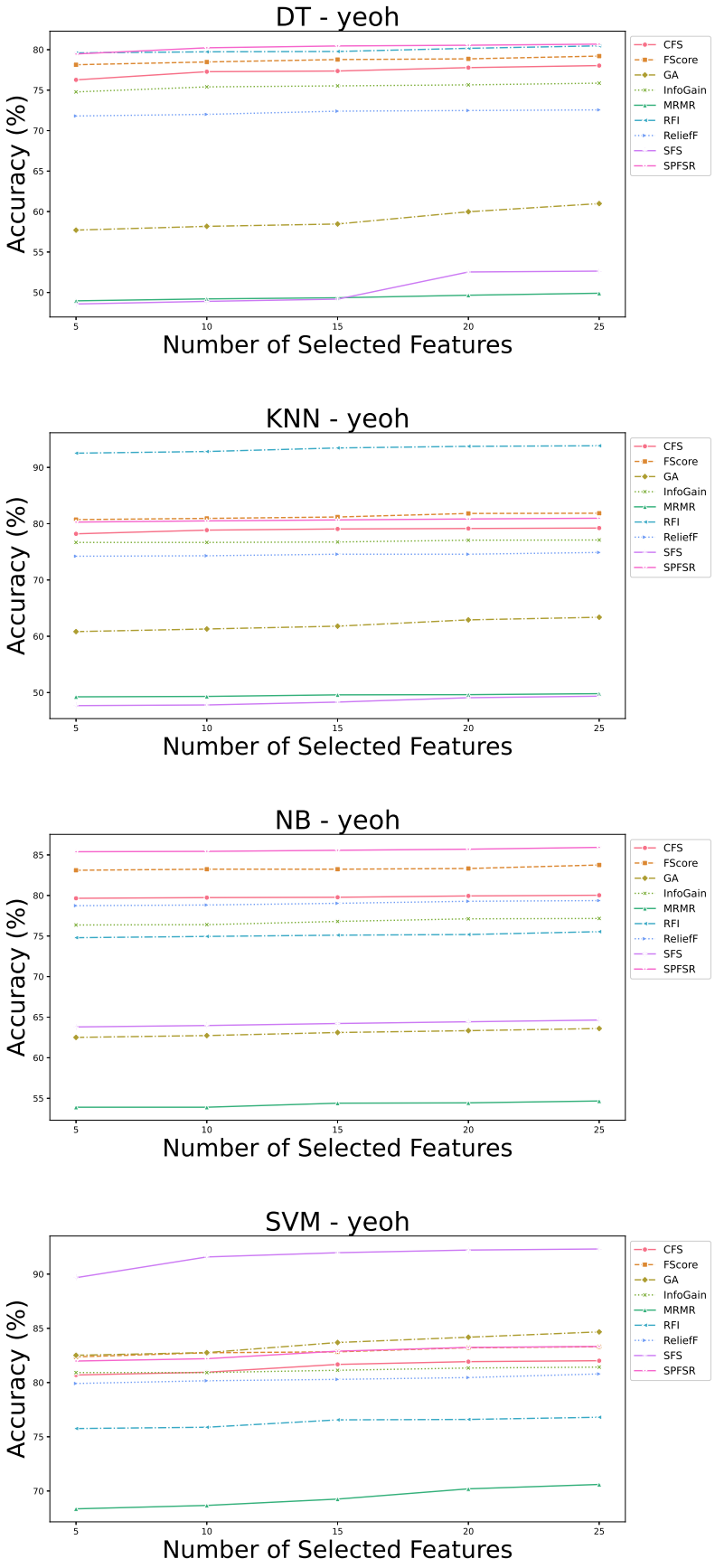

Figure D.20: The average accuracy - Yeoh