AN UPGCHI FEATURE SELECTION METHOD FOR MULTICLASS MICROARRAY DATA WITH APACHE SPARK FRAMEWORK

LWIN MAY THANT

UNIVERSITY OF COMPUTER STUDIES, YANGON

JUNE, 2024

An UpgCHI Feature Selection Method for Multiclass Microarray Data with Apache Spark Framework

Lwin May Thant

University of Computer Studies, Yangon

A thesis submitted to the University of Computer Studies, Yangon in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

June, 2024

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

…..…… ……..……

Date Lwin May Thant

ACKNOWLEDGEMENTS

First of all, I would like to thank the Union Minister, the Ministry of Science and Technology, for allowing me the advanced study and providing full facilities during the Ph.D. course at the University of Computer Studies, Yangon.

Secondly, I would like to express very special thanks to Dr. Mie Mie Khin, Rector of the University of Computer Studies, Yangon, for allowing me to develop this thesis and giving me general guidance during the period of my study.

I would also like to express my special appreciation and thanks to Dr. Win Pa Pa, Professor and Course-coordinator of the Ph.D 12th Batch, University of Computer Studies, Yangon, for her kindness, advice, and understanding, all of which have been extremely beneficial to me.

I would like to express my deepest gratitude to my supervisor, Dr. Tin Zar Thaw, Professor, Cloud Computing Lab., University of Computer Studies, Yangon, for her outstanding instruction, compassion, and patience, as well as for presenting me with excellent research suggestions.

I would like to express my respectful gratitude to Daw Aye Aye Khine, Professor and Head of English Department for her invaluable language assistance, which included pointing out proper usage not only in my Ph.D. course work but also in my dissertation.

I deeply and specially thank the external examiner, Dr. Aung Nway Oo, Professor, University of Information Technology, Yangon, for his patience in critical reading, and for his insightful recommendations and remarks during the thesis preparation.

My heartfelt gratitude also goes to all of my honorable professors for providing us with invaluable lectures and knowledge during our Ph.D. course work.

I also want to express my gratitude to my Ph.D. 12th Batch friends for providing me with the support and friendship that I required.

I owe a huge debt of gratitude to my family for never losing faith in me and for their unending love and support. Throughout my Ph.D. studies, they have always been kind and encouraging. Without them, this feat would not have been possible.

ABSTRACT

Thousands of gene expressions can be monitored using microarray technology in a variety of biological circumstances. Microarray data has the number of features is very large with respect to their samples and also has the nature of high-dimensionality. Due to the high-dimensionality, multiclass and complexity of gene expression data, there are many unknown and undiscovered functional relations in the physical delivery system used for collecting the data itself. Analyzing a microarray high-dimensional dataset, identifying the specific and intriguing genes that are responsible for the cause of cancer is critical. Generally, the selected attributes are not normalized in term of representatively per class which can impact the process of classification. To make up for the chi-square problem which caused the absence of attributes under some classes, an upgrade chi-square algorithm is proposed to balance the selection of the number of gene attributes per class. The proposed model is implemented using five microarray datasets like Leukemia, four classes Tumor and DLBCL cancer. The proposed method calculates the chi square value of each gene attributes for each of classes. Attributes belonging to the same class are sorted by Chi-square value and the features with the highest values in each class are selected. According to the gene attribute selection threshold value, the top number of attributes belong to each class is selected by the ratio values of their gene's records. After choosing the necessary features, the following step of this research is the implementation of different scalable classifiers in an efficient way. The useful classifiers Logistic Regression (LR), Random Forest and Naïve Bayes are evaluated on the scalable framework Spark. The proposed scalable models are tested on a Spark with the outcomes analyzed. The collected results show that the execution of scalable framework is much more efficient than traditional systems for processing large datasets. To evaluate the performance of the proposed system, UpgCHI is compared with three other univariate feature selection methods: original Chi-square, Linear Regression and ANOVA. The results show that the upgrade chi-square algorithm provides better performance on scalable frameworks in terms of classification accuracy, precision, recall and F-1score.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1
INTRODUCTION

Microarray technology represents a novel and efficient strategy for extracting diverse biomedical data suitable for a multitude of applications. The high-throughput analysis provides substantial insights into tumor variances among individuals in cancer research, with DNA, mRNA expression, and protein expression all being inherent factors. To analyze the gene expression levels, extract the right features, and classification of microarray data, the system needs more efficient methods, and a suitable platform is needed. Microarray data is a type of large-scale data having multiple dimensions and features that are greater than the samples. It contains a wide range of information, including RNA, DNA, and numerous cancer types. Differentially expressed genes in malignancies, pharmacological treatments, virus infection, and animal development have all been studied using microarray techniques. Due to the intricacy of differential gene expression and high-dimensional throughput sequencing, the physical system contains many little-known valuable correlations for data storage. In the field of gene expression studies, high-throughput sequencing technologies are crucial.

The microarray data comprises high-dimensional information and concerning their features quantity is much greater than the number of samples which is a main challenge in the research area. In these facts, storing, processing, and extracting features and classification of the disease is the major challenge in this area. The problem of the dimensionality curse is caused by microarray data with high dimensions that lose the required data information and happened to noisy data, and unreal correlations. Large sample sizes lead to several issues, including high computational cost and algorithmic instability [1]. To solve this issue, the genes or features selection techniques are the important processes in multiclass and large dimensional microarray data, which are procedures such as reducing the high-dimension and selecting appropriate features from the training data, and thus the classifier algorithms only look to predict the features and analysis of model and this ensures that the learning algorithm (classifiers) concentrates exclusively on the attributes of the training data that are pertinent for analysis and future prediction.

This research employs the utilization of distributed frameworks, Spark, to build machine-learning algorithms for analyzing high-dimensional microarray datasets. Feature

selection processes are started using UpgCHI algorithm employ on Apache spark platform. These selections are loaded with multiple machine learning pipeline according their microarray data classes on different Spark cluster. By selecting the genes data, this system can improve classification accuracy, time consuming and reduce spaces. In feature selection functions, the system is designed to extract, transform, and select features from various classes of microarray data. The aim is to reduce the computational cost of modeling and enhance model performance. In this research, novel filtering methods are introduced for high-dimensional classification using microarray gene expression datasets.

1.1 Statement of the Issue

Large-scale data applications have recently garnered significant attention due to the exponential growth in data collection and storage. However, traditional data mining techniques are ill-suited to meet the demands of this new landscape, posing challenges in extracting information from big data [2]. To address these challenges, researchers have explored various open-source frameworks such as MapReduce, Spark, and others to develop scalable algorithms capable of handling large-scale data processing [3]. In posing and addressing computational challenges, the researchers discovered that intelligent procedures have a substantial significance to the development of data science which addresses challenges related to imprecision, uncertainty, learning, and evolution [4-12].

To examine these datasets, different machine-learning approaches have been investigated in the field of bioinformatics [13-17]. On a conventional system with the best processing capabilities, these techniques take a long time to evaluate and explore massive datasets. To address this issue, the distributed computing concept has been implemented, in which multiclass data is analyzed utilizing various parallel processing pipelines such as Apache Spark. Because of the massive increase in data collection and storage, big data applications have recently become the center of attention.

Feature selection of multiclass microarray data to deal with the curse of dimensionality is a big problem. Small sample size, high dimensions and class imbalance are the most typical issues to overcome. Class imbalance can lower the credibility of classification accuracy. As the number of classes grows, the classification accuracy looks to rapidly deteriorate. One of the key components of microarray is the volume of quantitative information where several features are very large for samples. It can cause heavy computational costs in terms of space and time complexity and algorithmic instability.

Proposing UpgCHI algorithm is suitable for feature selection to the multiclass framework. To address these challenges, various open-source frameworks such as MapReduce, Spark, etc., have been explored for developing scalable algorithms. The distributed frameworks such as MapReduce and Spark for the implementation of machine learning techniques, are used to analyze the high-dimensional multiclass microarray datasets. Hence, the existing methodologies have been implemented on scalable platforms Apache Spark to analyze the microarray datasets.

1.2 The Motivation of the Research

One of the reasons why feature selection is being studied is to produce reliable classification or clustering findings for Microarray bioinformatics data. In addition, the lack of large-scale parallel processing is a driving force behind the research. Although different feature selection techniques exist for other microarray data, there is no parallel multiclass data processing accessible for feature selection, particularly for high-dimensional data. As a result, a novel feature selection approach is proposed for Apache Spark that has a huge scalable performance.

Another point is that the previous studies that can lead to computational imbalance because of the case of dimensionality. It is difficult to storage and processing microarray data in a set amount of time. Because storing microarray data in a standard database is wasteful, a better solution must be found. Feature selection methods are very enormous, so need to classify what method is to select the exact features based on the Apache Spark framework. Therefore, this research is to use different Apache Spark pipelines with MLlib and to develop the machine library in Spark system.

1.3 Objectives of the Research

The main objective of this study is to use multiple trials to construct a traditional machine learning system. Today, the statistical test approach is usually used in feature selection of machine learning systems. However, these approaches need to be implemented according to their data nature and problem. Feature selection of multiclass microarray data is still required to produce better performance model results. This is due to a variance of data and an unbalanced problem. By using the UpgCHI method, various multiclass data have been tested based on the Apache Spark framework for feature selection to get a good result. The following are the other objectives of this research area:

(i) To investigate the state-of-the-art approach for feature selection based on microarray

(ii) To develop an innovative approach for feature selection using a distributed methodology on the Apache Spark platform

(iii) To extract the optimal set of highly expressed features (genes) using a feature selection strategy leveraging the MapReduce and Spark frameworks.

(iv) To develop a fast feature selection using Apache Spark to select the dataset in a short amount of time.

(v) To improve the performance of the proposed model

1.4 The Contributions of the Research

Until recently, there were many feature selection techniques available in machine learning systems. There were several thousand genes and this sample data is smaller than the feature. The contribution is to build a large-scale parallel pipeline with an upgraded chi-square test over Apache Spark. By considering the variance of data and imbalance class for the selection of features in multiclass microarray data. This system's contributions are these points. Therefore, proposing upgrading the Chi-square algorithm that is suitable for feature selection to a multiclass framework.

To overcome the above challenges, many open-source frameworks like MapReduce, Spark, etc. have been considered to develop scalable algorithms. The distributed frameworks Spark for the implementation of machine learning techniques, which are used to analyze the high-dimensional multiclass microarray datasets. Hence, the existing methodologies have been implemented on scalable platforms Apache Spark to analyze the microarray datasets.

1.5 The Organization of The Research

This research paper begins with an overview of microarray high-dimensional data, as well as research obstacles and goals. It is divided into seven chapters, each of which depicts contributions specific to multiclass microarray data. The following is a diagram of the arrangement of the research.

In chapter 2, the present research on microarray data classification is reviewed, with a focus on two primary domains: (a) feature selection and (b) classification. A table

containing a comparison of several methodologies and their corresponding authors is offered. After study of literature, the related work is presented.

Chapter 3, the background theory of feature selection, detailed explain the distributed computing of Apache Spark and the nature of microarray data.

Chapter 4 describes the detailed explanation of filter-based feature selection with chi-square, ANOVA and linear regression feature selection over Apache Spark platform that used in this research.

In chapter 5, an UpgCHI based feature selection framework is presented under Apache Spark and system architecture are explained in both Spark platform.

In chapter 6, an UpgCHI test is implemented on scalable frameworks like MapReduce and Spark as a feature selection method. The presented UpgCHI method is used to identify significant characteristics from diverse microarray multiclass data. Finally, the performance of these solutions is evaluated using the Spark pipeline and compared to a traditional system. Traditional chi-square method in the machine learning library is used to investigate the performance of the classifier, and a comparison is done on the Spark platform.

In chapter 7, the results drawn from the proposed approaches are presented, with a focus on accomplishments and limits. In conclusion, the possibilities for future research are highlighted.

CHAPTER 2
LITERATURE REVIEW

This chapter focuses on cutting-edge techniques, such as feature extraction and selection, dimensionality reduction, and building the classifier in pattern classification systems. This chapter emphasizes the classification of microarray datasets and the research done by various authors. The authors picked many criteria for the survey, including feature selection and classification methods, as well as the dataset used. According to the findings of the survey work on microarray data classification, a large number of academics and practitioners used statistical tests as a feature selection method and various machine learning algorithms to categorize the dataset.

In this chapter, many different methods such as the chi-square test, T-test, Wilcoxon test, F-test, Signal-to-Noise Ratio (SNR), Information Gain, Fisher Score, and Gini Index are applied to select the features. Using several classifiers, the top-rated genes are utilized to classify the microarray data like Support Vector Machine (SVM), Naïve Bayes (NB), Logistic Regression (LR), K-Nearest Neighbor (KNN), and Probabilistic Neural Network (PNN) and then analyzing the result. The remainder of the chapter is laid out as follows: a literature review is presented in Section 2.1 and the feature selection using scalable ANOVA, Kruskal-Walli's test, and Friedman test is also described in Section 2.2.

2.1 Literature Review

This section provides a quick overview of the work done by several authors on microarray data. In [18], author Osareh et al. presented the Signal-to-Noise Ratio (SNR) feature selection technique to select the genes and classify with Support vector machine (SVM), K-nearest neighbor (KNN) and Probabilistic neural network (PNN) are used. In [19], Bharathi and Natarajan are considered the implementation with two steps. In the first phase, the researchers used a two-way Analysis of Variance (ANOVA) ranking scheme to choose several key genes. In the second phase, they used a strong classifier like Support Vector Machine to assess the classification ability of all simple combinations of those essential genes. With only two genes, this paper was able to achieve great accuracy.

Tang et al. is presented that the use of a sample profile of gene expression as a better indicator for cancer classification should be investigated further in [20]. This paper uses the ANOVA test with a Discriminant kernel partial least square (PLS) classifier and

instead of partial gene sets, the full gene expression profile for all samples was attempted as an alternative. The results indicated a significant improvement in classification accuracy when compared to other traditional methods.

In another study, Mundra et al highlighted the t-statistic into two parts which are according to correspond to relevant and irrelevant data points to select the genes for cancer classification [21]. Support vectors are used to pick important data points, which are subsequently utilized to generate the t-statistic for feature selection. On synthetic and benchmark cancer datasets, much superior classification results are produced by simultaneously selecting data points and genes.

Afterward, Huerta performed the gene selection and classification for microarray data with a t-test and Linear Discriminant Analysis LDA-GA classifier [22]. The main feature of this LDA-based GA algorithm is that it uses not just an LDA classifier in its fitness function, but also LDA discriminant coefficients in its specific crossover and mutation operators.

Author Liu et al. offer a new ensemble gene selection approach (EGS) for selecting multiple gene subsets for classification, in which the significant degree of each gene is quantified using conditional mutual information or its normalized version [23]. After distinct gene subsets have been acquired by using different search starting points, they will be used to train several base classifiers, which will then be aggregated into a consensus classifier using majority voting.

In the next study, author Zhang et al. is to develop a microarray data classifier. The classification process begins with the reduction of microarray data dimension [24]. The Wavelet packet transforms and neighborhood rough set (WPT + NRS) and classifier with SVM approach is used to reduce the microarray data dimension by decomposing the samples until a specific level of decomposition is reached, and then using approximation coefficients at those levels as classifier features.

In the later review, author Abeel et al. analyzed the robustness of a biomarker selection algorithm with ensemble SVM-RFE and SVM. SVMs are sophisticated classification models that have demonstrated state-of-the-art performance on a variety of biological data diagnosis and prognosis applications [25].

Thereafter, author Dina et al. proposed a new filter multiple scoring gene selection technique MGS-CM. This approach is integrated with three classifiers to create three novel classification systems (MGS-SVM, MGS-KNN, and MGS-LDA), all of which have been

tested and evaluated on three microarray datasets [26]. The proposed approaches produced excellent results and ensured accurate classification of previously unidentified samples.

Hereafter, author Lee et al. showed a novel hybrid method in microarray data analysis for feature selection [27]. This method begins by generating several gene subsets using a genetic algorithm with dynamic parameter setting (GADP) and then ranking the genes based on their occurrence frequencies in the gene subsets. The chi-square test for homogeneity is then used to pick a suitable number of top-ranked genes for data analysis. The efficiency of the selected genes is verified using the support vector machine (SVM). The performance of the GADP approach is compared to that of existing methods using six distinct microarray datasets.

In the later review, author Mishra et al. provided two approaches to feature selection. The genes in microarray data are first clustered using k-means clustering, and then SNR ranking is used to extract the top-rated features from each cluster, which are then passed to two classifiers for validation: SVM and k-NN [28]. The features (genes) of the microarray data set are sorted using solely SNR ranking in the second approach, and the highest scoring feature is supplied to the classifier and validated. The result is compared to the accuracy of several approaches for leukemia data set with LOOCV published in the literature, with only the multiple-filter-multiple wrapper approach giving 100 percent accuracy in LOOCV with leukemia data set.

In [29], author Maji et al. introduced a new feature selection approach based on rough set theory. It chooses a collection of genes from microarray data based on the relevance and significance of the genes. The use of both relevance and importance criteria for selecting a reduced gene set with good prediction accuracy is justified using a theoretical analysis. The value of rough set theory in calculating gene relevance and significance is also demonstrated. Using the prediction accuracy of the K-nearest neighbor rule and support vector machine on five cancer and two arthritis microarray data sets, the performance of the proposed approach is investigated, as well as a comparison with other relevant methods.

In [30], author Sun et al. proposed a new gene selection method by preserving cancer diagnostic and classification intrinsic groupings of interacting genes. The proposed method outperforms existing selection methods in terms of classification performance and enrichment score, according to experimental results.

In [31], author Sun et al. also presented a new information-theoretic framework for analyzing feature relevance, dependency, and redundancy Then, a dynamic weighting-

based feature selection technique is proposed, which seeks to keep important inherent groups of interdependent features while selecting the most relevant features and eliminating redundant features. Experiments on six UCI data sets and four gene microarray datasets using three common classifiers like SVM, KNN, and Predictive Analysis of Microarray (PAM) are used to validate the efficiency of our technique.

In [32], author Yeh et al. introduced the OA-SVM wrapper approach, which uses an orthogonal array (OA) to provide systematic feature selection criteria and a support vector machine (SVM) as the classifier. For the classification task, the suggested OA-SVM is used to test eight UCI databases. These studies showed that the proposed OA-SVM for feature selection can successfully eliminate irrelevant or redundant features, resulting in improved classification accuracy. It aims to remove irrelevant or superfluous features to minimize feature dimension and compute complexity while increasing classification accuracy.

In [33], author P. Guo et al. contributed three feature selection methods that were used to provide 21 features belonging to 18 genes as possible markers. The final psoriasis classification model was created with the new Incremental Feature Selection approach, which uses only three characteristics from two distinct genes, IGFL1 and C10orf99. Over three distinct validation procedures, this model has shown exceptionally steady prediction accuracy (averaged at 99.81 percent). The two marker genes IGFL1 and C10orf99 were discovered to be upstream components of the psoriatic pathogenesis growth signal transduction pathway.

In [34], the RotBoost ensemble methodology was used by author Osareh et al. to handle the gene categorization problem. This method combines Rotation Forest and AdaBoost algorithms, preserving both desirable aspects of an ensemble architecture, namely accuracy and diversity. Five alternative feature selection algorithms are evaluated to choose a compact subset of informative genes. Other nonensemble/ensemble techniques such as Decision Trees, Support Vector Machines, Rotation Forest, AdaBoost, and Bagging are used to evaluate RotBoost's efficiency. The combination of the rapid correlation-based feature selection method with the ICA-based RotBoost ensemble is highly effective for gene classification, according to experimental results.

In [35], author Hengpraprohm presented the method for microarray data to classify gene expression cancer data. The suggested method combines classification and feature selection techniques. The feature selection technique is the signal-to-noise ratio, while the classification technique is the Genetic Algorithm (GA) (SNR). The suggested method is

tested using Lymphoma and Leukemia datasets, with a 10-fold cross-validation procedure utilized to provide the experimental results in terms of classification accuracy. The findings reveal that the suggested method outperforms the standard GA-based classifier in terms of classification accuracy and several generations required to find solutions. Furthermore, the findings are compared to various classification and feature selection strategies reported in the literature, and it is discovered that the proposed method produces good results, particularly in the Lymphoma dataset.

When looking back at prior studies, they only employed feature selection to reduce the number of characteristics to improve the performance of constructed classifiers. Nonetheless, these steps are insufficient to produce a more accurate classification. The results of both strategies will be compared in terms of precision, recall, and f-measures. Techniques such as Naive Bayesian (NB), Decision Tree and K-Nearest Neighbors (KNN), Neural Network (NN), and Support Vector Machine (SVM) have been utilized for classification tasks in terms of recall, precision, and f-measure, in addition to the aforementioned methods.

2.1.1 Experimental Analysis of Existing Method

This phase outlines the proposed method for microarray statistics which is divided into several stages:

1. Emphasizes preprocessing the entered facts using various techniques inclusive of lacking information imputation, and normalization.
2. Feature selection is executed through the usage of numerous methods like T-take a look at, F-take a look at, Wilcoxon check, SNR, $\chi 2$-take a look at, records benefit, Gini Index, and Fisher rating.
3. Classification is executed using distinct classifiers like Logistic regression (LR), Naive Bayes (NB), Support vector Machine (SVM), artificial neural network (ANN), Radial basis function network (RBFN), Probabilistic neural network (PNN), and K-Nearest neighbor (KNN).

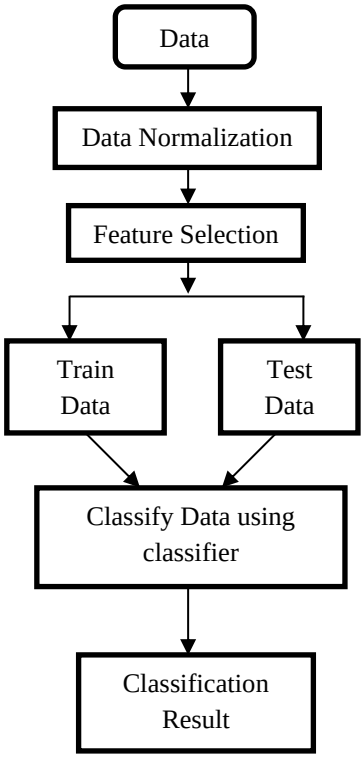

Figure 2.1: Step-by-step Microarray Data Classification

The stages that follow provide a quick overview of the proposed approach.

1. **The collection of Data**

   The classification dataset, which serves as a necessary input to the models, is gathered from Mendeley Data (this is a cloud base repository)

2. **Data Normalization**

   The mean value of the respective feature is used to impute missing data of a feature (gene) in microarray data. The Min-Max normalization approach is used to normalize the input feature values over the range [0, 1].

3. **Grouping of Datasets**

   The dataset is split into two sections: training and testing.

4. **Feature Selection**

   To choose the features with the highest relevance value, different statistical methods have been used for reducing the dimensionality problem.

5. **Classifier**

   The microarray dataset has been classified using a variety of classifiers.

6. **Model Validation**

   The model is tested using the testing dataset, and the classifier's performance is compared to several performance metrics.

2.2 Related Works

Many methods have been applied in the field of Microarray data classification to identify pertinent attributes and ascertain the ideal number of features from the large initial dataset [36]. The enormous dimensionality of microarray data has historically made categorizing it difficult [37]. Thanks to advances in microarray data technology, researchers can now analyze a single experiment involving over 1000 genes. The numerical data in microarray datasets is highly valuable despite being compact, but because of its size, it comes with high computing costs and algorithmic instability. Numerous issues with gene expression analysis have been identified, such as a small sample size, insufficient validation, and a large number of feature genes affected by noise and outliers. Gene expression levels are frequently measured using microarray technology, and the data obtained from this process is used to diagnose and treat a variety of illnesses.

Microarray technology emerges as a potent tool for addressing numerous biological challenges, holding the potential to unveil a plethora of valuable information. While the features within the raw dataset range from 6,000 to 60,000, the number of training and testing samples can be modest, sometimes fewer than 100 instances. Given that this method doesn't mandate the inclusion of all genes, feature selection becomes crucial to isolate the pertinent genes. Faced with high-dimensional complexities such as classification [38, 39] or clustering problems, researchers must employ feature selection techniques [40] to derive meaningful insights and identify genes with significant biological relevance [41]. As not every feature in the dataset contributes to predictive accuracy, feature selection is imperative, removing redundant and extraneous characteristics [42]. Various machine-learning approaches [43] can be leveraged to evaluate microarray datasets. Consequently, the primary objective of this study is to feature selection for microarray data [44].

An improvement on the CHI algorithm that integrates variance and coefficient of variation is the Var-CV-CHI algorithm, which was introduced by Liang-jing Cai. The purpose of this work is to address the shortcomings of the CHI algorithm by introducing a novel technique. Both balanced and unbalanced datasets are used in the experiments, which involve text classification in both Chinese and English using two different classifier

algorithms (KNN and Naive Bayes). Most importantly, it emphasizes that using the suggested algorithm produces significant improvements over results using the original CHI algorithm. The strengths of this paper are its remarkable classification efficacy and the wider range of experiments it includes than previous studies. But only the distribution of feature words is covered in the paper; their semantic information is not taken into account [45].

The problem of choosing a suitable method in the lack of accepted benchmarks for cutting-edge discoveries in science was discussed by the writers. The primary goal of this work is to present a noteworthy experimental comparison that assesses the influence of the feature selection process on various classification algorithms. To achieve this goal, they looked at both novel approaches and ranking-based feature-selection processes, allowing for a comprehensive comparison. Results from multiple traditional microarray datasets with different attributes and patient counts are presented broadly, providing insight into the potential outcomes of these techniques [46].

Agnès Bonnet, Sébastien Gadat, and Kim-Anh Lê Cao: The "Optimal Feature Weighting" (OFW) method for feature selection in multiclass problems is presented in this work. As classifiers, it makes use of one-vs-one Support Vector Machines (SVM) and a Classification and Regression Tree (CART) [47]. Based on the Scikit-learn algorithm, the authors proposed and evaluated $\chi 2$ feature selection over Apache Spark using the Databricks platform. This paper aimed to minimize training time, enhance precision, and prevent overfitting of the training set. Advanced linear algebra, feature selection, and machine learning algorithms over Apache Spark will be studied and implemented in future work [48].

Fuzzy feature selection based on relevancy, redundancy, and dependence (FFS-RRD) stands as a novel ensemble feature selection method proposed by Omar A. M. Salem, Feng Liu, Yi-Ping Phoebe Chen, and Xi Chen. This approach considers the discriminatory capacity of both individual and dependent features to extract all possible feature relationships. Through experimental comparison with eight state-of-the-art and traditional feature selection techniques, the efficacy of the proposed method is evaluated. Results from experiments conducted on 13 benchmark datasets and compared against four established classifiers highlight the superior performance and stability of the suggested method in terms of classification accuracy [49].

Lee and Leu Yungho proposed a novel hybrid strategy for feature selection in microarray data analysis. This method uses a genetic algorithm with dynamic parameter

setting (GADP) to generate multiple gene subsets, which are then used to rank the genes based on their occurrence frequencies in the gene subsets. In order to select a suitable number of the top-ranked genes for data processing, the $\chi 2$-test for homogeneity is also employed in this process. The support vector machine (SVM) should be used to verify the selected genes' efficacy. Six different microarray datasets are used to compare the performance of the GADP approach to that of the current methods [50].

2.3 Chapter Summary

In this chapter, the surge in diseases today has resulted in a significant expansion of data volume across various disease categories, leading to substantial human losses. Identifying the factors causing a specific class of disease within vast datasets early on is crucial. However, existing platforms are inadequate for handling the evolving landscape of data analytics. Leveraging the concept of ‘ Big Data’ can help mitigate the complexity of classifying diseases into specific categories. Achieving this goal is possible by employing technologies like High-Performance Computing (HPC), Hadoop, Spark, etc., which reduce the time needed to classify diseases into specific categories. Based on the aforementioned research, it is noted that the statistical t-test as a feature selection method yields superior outcomes across different classifiers.

CHAPTER 3
BACKGROUND THEORY

This chapter presents an introduction to microarray data and background theory of distributed computing. In addition, the framework of Hadoop with MapReduce is also discussed.

3.1 Introduction to Microarray Data

Numerous websites, such as Kaggle, UCI, Mendeley Dataciteb12, the Global Health Observatory Data Repository, and many more, offer microarray datasets. You may easily store, share, access, and cite your data from anywhere with Mendeley Data, a safe cloud-based data repository. The following datasets were gathered by this approach from the Mendeley Data website (Dabba, Tari, & Mokhtari, 2021): lymphoma, lung, breast, ovary, and CNS (central nervous system). There are enormous datasets available, with most of the features falling between 6000 and 60,000. Table 1 contains details about the datasets. Using the microarray dataset, this system may use a range of machine-learning algorithms. Microarray data, which is usually a high-dimensional dataset, was used in this system. Therefore, let can draw the conclusion that two common aspects of microarray data are high-dimensional features and a small number of samples. The classification of microarray data presents a significant challenge for machine learning researchers because of the small sample sizes and abundance of variables.

One of the most recent developments in experimental molecular biology is the development of microarrays, which enable the simultaneous monitoring of tens of thousands of gene expression levels. Gene expression, genome mapping, SNP discrimination, transcription factor activity, toxicity, pathogen detection, and many other areas of study have all benefited from the use of arrays. Because there are more features in the dataset than there are occurrences, there may be an increased risk of overfitting in machine learning models. When a model learns to perform extraordinarily well on training data but finds it difficult to generalize to new, unknown data, it is said to be overfitting. This is because the model has successfully learned patterns and noise unique to the training set. Excessive feature counts in comparison to instance counts may make overfitting more likely. Each dataset's extensive feature collection probably reflects the complexity and

diversity of the data being examined. Every feature in the data is an attribute or trait that could help with the current classification or prediction task.

In certain sectors, working large datasets containing dozens or even millions of features are commonplace, especially in subjects like image analysis, high-dimensional data analysis, and genomics. Measurements, gene expressions, picture pixels, or any other pertinent data that might have useful patterns could be included in these features. Overfitting is more likely to occur when working with datasets that have a high dimensionality (many features) and a small number of examples. The reason for this is that the model might identify noise or erroneous correlations in the data, which would result in unduly complicated models that would not adapt well to fresh data. The Feature Selection method, which prioritizes feature selection strategies to identify and keep only the most relevant and informative features, is used to prevent overfitting in datasets with a large number of features relative to occurrences. Removing superfluous or unnecessary characteristics helps simplify the model and lower the chance of overfitting.

Table 3.1 Sample Microarray Dataset Details

	Diffuse LargeB-Cell Lymphoma (DLBCL)	Leukemia	Leukemia	Brain Tumor	Brain Tumor
# Genes	5469	7129	7129	5920	5920
#Classes	2	3	4	4	5
#Instances	77	72	72	50	90
#Instances per Class	58/19	38/9/25	38/21/10/3	20/10/10/4/6	60/10/10/4/6
Class Names	B lymphocytes or B cells	B-cell acute lymphoblastic leukemia, T-cell acute lymphoblastic leukemia and Aacute myeloid leukemia	acute lymphoblastic leukaemia (ALL), chronic lymphocytic leukaemia (CLL), acute myeloid leukaemia (AML) and chronic myeloid leukaemia (CML).	Astrocytoma, Diffuse Astrocytoma (grade II), Anaplastic Astrocytoma (grade III), Glioblastoma Multiforme (grade IV),	Astrocytoma, Pilocytic Astrocytoma (grade I), Diffuse Astrocytoma (grade II), Anaplastic Astrocytoma (grade III), Glioblastoma Multiforme (grade IV)

3.2 The Distributed Computing

Distributed computing techniques such as grid and cluster computing have been used for many years to improve the efficiency of a wide range of data-intensive applications. While many of these methods are based on a message-passing paradigm, shared-memory models are used by systems that use parallel algorithms, including graphics processing units (GPUs). For these computers to function properly, they need an interface that makes it easy to access shared file systems and communicate with other machines in the grid or cluster. For these systems, network capacity frequently becomes a constraint. Bringing "compute to data," as opposed to the traditional strategy of "taking data to the compute," is the basic premise of distributed systems. This change turns out to be quite cost-effective, particularly when data sizes are above a certain threshold since it is generally less expensive to move "compute to data" than it is to move "data to compute."

3.2.1 The Framework of Hadoop

Hadoop, an open-source software framework, facilitates the storage and processing of extensive datasets across distributed clusters of commodity hardware [51]. Its architecture is devised to distribute data storage across all nodes (servers) within a cluster by partitioning files into smaller units known as blocks. This approach allows each node to prioritize processing data available locally, thus minimizing network transmissions. Furthermore, Hadoop offers a cost-effective and scalable infrastructure capable of seamlessly expanding from a single server to thousands of servers. Its design incorporates mechanisms for identifying and managing failures at the application layer, ensuring continued service even in error-prone environments. The Hadoop framework comprises three core components, which are maintained and overseen by the Apache Software Foundation. These components include:

3.2.1.1 The Hadoop Distributed File System (HDFS)

Unlike relational databases, this distributed file system with Java support maintains many kinds of data in an unstructured manner. Although it is similar to current distributed systems, it has notable differences. Because of its architecture, it can be quickly recovered from failures and is deployable on cheap hardware. The system, which consists of several data nodes and a name node, functions as follows: Without keeping the actual files, the name node keeps track of all the files in the file system and keeps an eye on where they are being stored throughout the cluster. Additionally, files stored in a data node are

replicated across other data nodes, facilitating swift recovery in case of node failures. Upon file writing, the name node updates the directory tree and dispatches the file to a data node, which then replicates it across other data nodes for redundancy [52]. Refer to Figure 3.1 for an illustration of HDFS architecture.

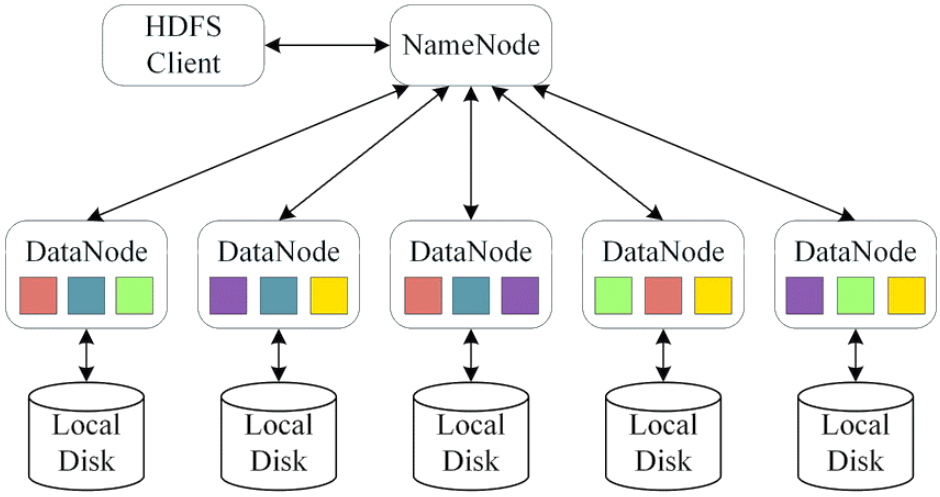

Figure 3.1 Architecture of Hadoop Distributed File System

3.2.1.2 The Yet Another Resource Negotiator (YARN)

This is a framework for resource management that allows distributed applications to schedule and manage resource requests. YARN integrates node manager agents, which keep an eye on processing activities in specific cluster nodes, with a central resource manager, which controls how applications use Hadoop system resources. When a node malfunctions, the task is rescheduled to another node [53].

3.2.1.3 The MapReduce

This programming model was created by Google to process big datasets across clusters of commodity hardware in a distributed fashion [54]. Three steps are involved:

- **Map step**: Every node uses local data to apply the "{map()" function, and then writes the output—along with a key value—to temporary storage.
- **Shuffle step**: the nodes reorganize the data according to the output keys (generated by the map function) so that all of the data associated with a single key are found on the same node.
- **Reduce step**: All output data groups are processed in parallel by each node, per key, and the output is subsequently written to the HDFS.

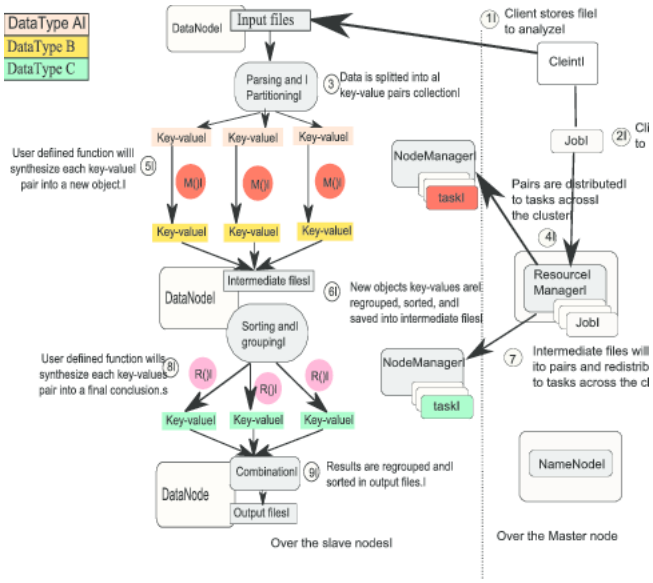

Figure 3.2 MapReduce Architecture

In addition to these core components, the Hadoop ecosystem includes a wide range of related projects and tools that extend its capabilities for various use cases and scenarios. Some of the notable projects in the Hadoop ecosystem include:

- **Apache Hive**: A data warehouse infrastructure built on top of Hadoop for querying and analyzing large datasets using a SQL-like language called HiveQL.
- **Apache Pig**: A high-level platform for creating MapReduce programs using a scripting language called Pig Latin.
- **Apache HBase**: A distributed, scalable, and NoSQL database that provides real-time read/write access to large datasets.
- **Apache Spark**: A fast and general-purpose cluster computing system that provides in-memory data processing capabilities and a more expressive programming model than MapReduce.
- **Apache Kafka**: A distributed streaming platform used for building real-time data pipelines and streaming applications.
- **Apache Sqoop**: A tool designed for efficiently transferring bulk data between Hadoop and structured data stores such as relational databases.
- **Apache Flume**: A distributed, reliable, and available system for efficiently collecting, aggregating, and moving large amounts of log data from various sources to Hadoop.

Overall, Hadoop's flexible architecture, fault-tolerant design, and scalability make it a powerful framework for processing and analyzing big data across distributed computing environments. As organizations continue to grapple with ever-increasing volumes of data, Hadoop remains a fundamental tool for unlocking insights and driving innovation in the field of data analytics.

3.2.2 Spark Architecture and Resilient Distributed Dataset

Apache Spark is a general-purpose, fast cluster computing framework designed for processing massive amounts of data. It is an extension of the widely used MapReduce data flow model, designed to support interactive and iterative computation more effectively. These types of computation are highly challenging to implement with Hadoop MapReduce. Spark was created especially for use cases where a working set of data is used repeatedly in parallel processes.

3.2.2.1 The Architecture of Spark

Spark, an open-source processing engine, leverages a directed acyclic graph along with its proprietary data structure, Resilient Distributed Dataset (RDD), to deliver accelerated speed and advanced analytics capabilities [55]. Integrated atop existing Hadoop infrastructure, Spark extends and enriches its functionality. Its architecture comprises a single driver program and numerous worker nodes. The driver system undertakes the creation of RDDs and Spark Context objects, while the worker nodes engage in parallel RDD transformations. RDD supports two primary operations: Transformation and Action. Spark offers flexibility by accommodating various cluster managers such as YARN and MESOS, in addition to its standalone cluster manager. Refer to Figure 3.3 for an illustration of Spark architecture, which includes the Cluster manager, Driver node, and Worker nodes. The components of Spark are delineated as follows:

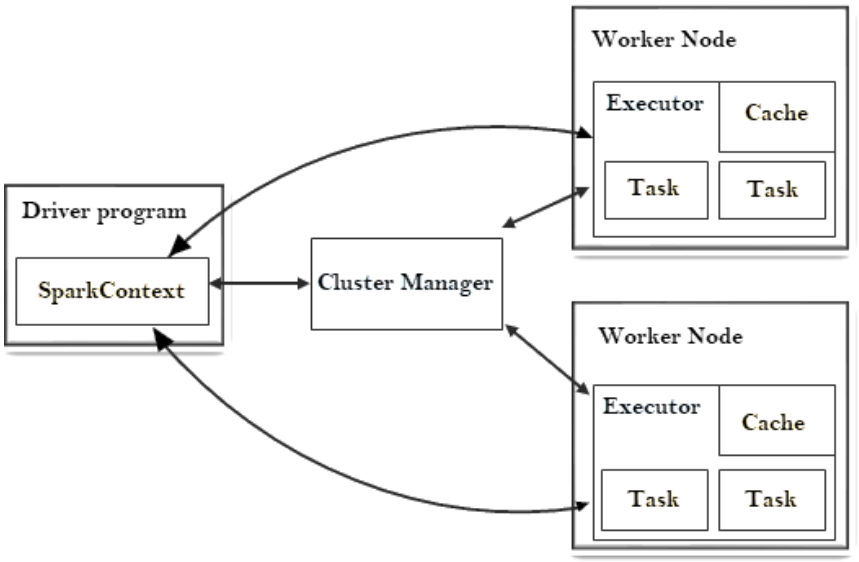

Figure 3.3 An Architecture of Apache Spark

Spark's architecture is based on a master-worker model and consists of several key components:

1. **Driver Program**: The driver program is the main control program that coordinates the execution of Spark applications. It contains the user's SparkContext, which represents the connection to the Spark cluster and coordinates the execution of various operations on the cluster.
2. **Cluster Manager**: Spark can run on various cluster managers like Apache Mesos, Hadoop YARN, or its built-in standalone cluster manager. The cluster manager allocates resources across applications and manages the execution of tasks on worker nodes.
3. **Worker Nodes**: Worker nodes are the machines in the cluster that execute the actual Spark tasks. Each worker node runs an executor process, which is responsible for executing tasks and storing data in memory or on disk.
4. **Executor**: Executors are worker processes responsible for executing tasks on a particular node in the cluster. They load data into memory, execute transformations and actions, and return results to the driver program.
5. **RDDs (Resilient Distributed Datasets)**: RDDs are the fundamental abstraction in Spark that represents distributed collections of data. RDDs are immutable, partitioned collections of objects that can be processed in parallel across a cluster. RDDs support two types of operations: transformations, which create a new RDD from an existing one, and actions, which return results to the driver program.

3.2.2.2 Implementation on Spark

In Spark, every task consists of a series of operators that work with a set of data. By using these operators, a Directed Acyclic Graph (DAG) can be created. This DAG is optimized by combining and rearranging operators as needed and whenever it is feasible. Spark's code base is compact, and the system is separated into multiple layers that operate independently of one another. Every layer has certain responsibilities.

1. Spark employs a modified Scala interpreter as its first layer for code interpretation.

2. Upon user input in the Spark console, which triggers RDD creation and operator application, Spark generates an operator graph.

3. When executing an action such as 'collect', Spark activates the DAG Scheduler, which receives the graph submission. This scheduler dissects the operator graph into distinct map and reduce stages.

4. Stages are comprised of tasks corresponding to partitions of input data. The DAG scheduler optimizes the graph through operator pipelining, consolidating multiple map operators into a single stage. This optimization greatly influences Spark's performance, yielding a set of optimized stages.

5. Following classification into Map or Reduce stages, as discussed in Hadoop, the stages are relayed to the Task Scheduler. The Task Scheduler, functioning independently of stage dependencies, initiates task launches through the cluster manager (Spark Standalone/YARN/MESOS).

6. Tasks are executed on the Slave by the Worker. Each job prompts the initiation of a new Java Virtual Machine (JVM). The Worker possesses awareness limited to the code it receives for execution.

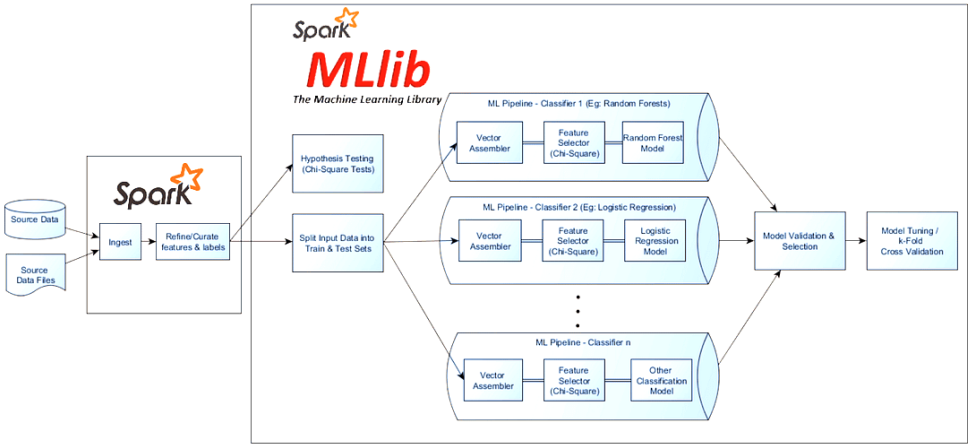

Figure 3.4 Apache Spark Architecture in Feature Selection

3.2.2.3 Resilient Distributed Dataset (RDD)

RDD is the core data structure in Spark, providing fault-tolerant distributed data storage and processing capabilities. RDDs offer the following key features:

- **Immutable**: RDDs are immutable, meaning that once created, they cannot be modified. However, transformations can be applied to create new RDDs derived from existing ones.
- **Partitioned**: RDDs are divided into logical partitions, which are distributed across nodes in the cluster. Partitions are the unit of parallelism, allowing Spark to process data in parallel across multiple nodes.
- **Resilient**: RDDs are resilient to failure, meaning that if a partition of an RDD is lost due to a node failure, Spark can reconstruct the lost partition by recomputing it from the original data lineage.
- **Lazily Evaluated**: Transformations on RDDs are lazily evaluated, meaning that Spark does not compute the results immediately. Instead, transformations are recorded as a lineage of transformations, and Spark optimizes the execution plan before executing the computation.
- **Fault-Tolerant**: RDDs provide fault tolerance through lineage information. Spark tracks the lineage of each RDD, enabling it to reconstruct lost partitions in the event of node failures.

In summary, Apache Spark's architecture is designed to provide high performance, fault tolerance, and scalability for distributed data processing applications. The RDD abstraction forms the foundation of Spark's processing model, enabling efficient and resilient distributed data processing across large clusters of machines. With its powerful

abstractions and flexible programming interface, Spark has become a popular choice for building a wide range of big data applications and analytics pipelines.

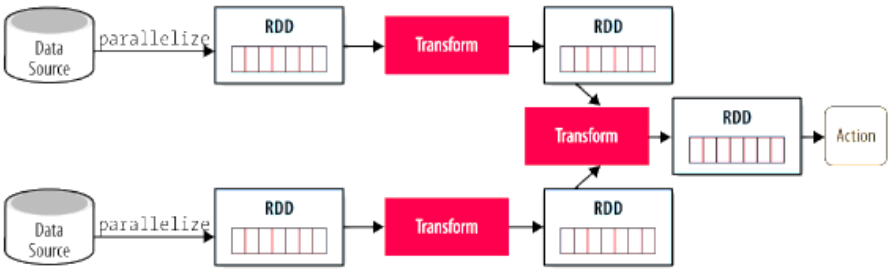

Figure 3.5 Resilient Distributed Dataset in Apache Spark

3.3 Machine Learning

Machine learning, an offshoot of computer science, emerged from the study of pattern recognition and computational learning theory within artificial intelligence. It delves into crafting and analyzing algorithms capable of learning from data and making predictions. These algorithms construct models based on sample inputs to facilitate data-driven forecasts or decisions, diverging from rigid program instructions. Machine learning intersects closely with computational statistics, a field specializing in prediction-making, and draws heavily from mathematical optimization for methodological and practical applications.

In computing, machine learning finds utility across tasks where explicit algorithm design and programming prove impractical. Its applications span various domains, including spam filtering, optical character recognition (OCR), search engines, and computer vision. Though sometimes equated with data mining, machine learning typically emphasizes predictive analytics over exploratory data analysis. Notably, machine learning and pattern recognition are perceived as complementary aspects of the same field. In industrial settings, machine learning methodologies may be termed predictive analytics or predictive modeling.

During training, a portion of the data is reserved for evaluation to assess the model's accuracy when presented with new data, resulting in a versatile model adaptable to diverse datasets. Researchers highlight three primary functions of machine learning systems: descriptive, wherein the system interprets data to explain past occurrences; predictive, wherein it anticipates future events based on data; and prescriptive, wherein it offers suggestions for action based on data analysis.

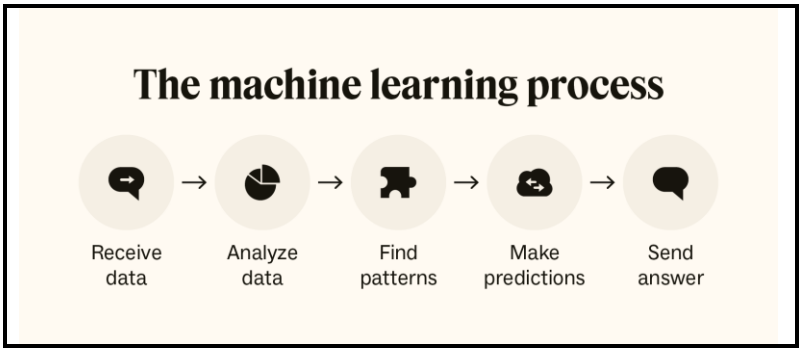

Figure 3.6 The Machine Learning Process

3.3.1 Different Types of Machine Learning

Classical machine learning is commonly classified based on how algorithms enhance their predictive accuracy, employing four fundamental approaches: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. The selection of algorithmic methods by data scientists depends on the nature of the data they aim to predict.

- **Supervised learning** involves data scientists providing algorithms with labeled training data, specifying both input and output variables for correlation assessment.
- **Unsupervised learning** entails algorithms training on unlabeled data, scanning through datasets to identify meaningful connections. The data used for training and the predictions/recommendations generated are predetermined.
- **Semi-supervised learning** combines aspects of supervised and unsupervised learning, wherein algorithms are predominantly fed labeled training data but are also allowed to explore the dataset independently to develop their understanding.
- **Reinforcement learning** is employed by data scientists to teach machines multi-step processes governed by well-defined rules. Algorithms are programmed to execute tasks and receive positive or negative cues as they progress, albeit largely determining their course of action autonomously.

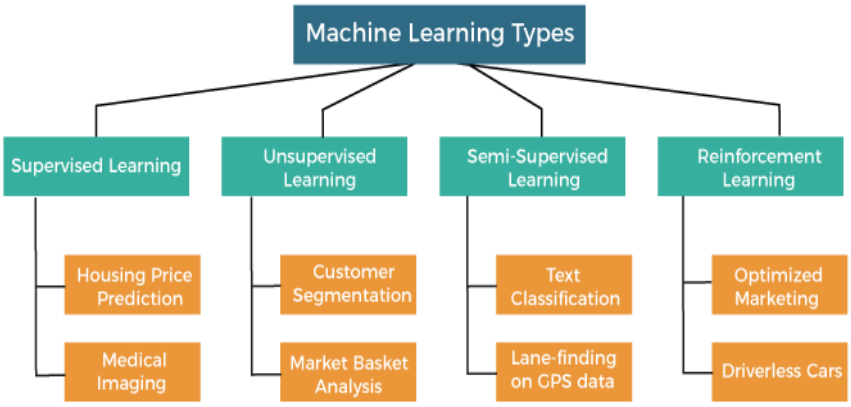

Figure 3.7 The Machine Learning Techniques

3.3.2 Machine Learning Processes in Data Analytics

The machine learning process encompasses a series of steps aimed at developing predictive models and extracting insights from data. This process involves data preparation, model selection, training, evaluation, and deployment. Below is an overview of the machine learning process with explanations for each stage:

1. **Problem Definition and Data Collection**:
   - Define the problem to be solved and establish clear objectives. Determine whether the problem is classification, regression, clustering, or another type of task.
   - Collect relevant data from various sources, ensuring it's representative, accurate, and sufficient for the intended analysis.
2. **Data Preprocessing and Exploration**:
   - Clean the data by handling missing values, outliers, and inconsistencies. Perform data normalization or standardization to ensure consistency in scale and distribution.
   - Explore the dataset through descriptive statistics, visualizations, and correlation analyses to understand the relationships between variables and identify patterns.
3. **Feature Engineering and Selection**:
   - Engineer new features or transform existing ones to capture relevant information and improve model performance.
   - Select the most informative features using techniques such as filter methods, wrapper methods, or embedded methods to reduce dimensionality and computational complexity.

4. **Model Selection and Training**:
    - Choose appropriate machine learning algorithms based on the problem type, dataset characteristics, and performance requirements.
    - Split the dataset into training, validation, and test sets to train and evaluate the model. Apply techniques like cross-validation to assess model performance robustness.
    - Train the selected models using the training data, adjusting hyperparameters and optimization strategies to minimize error and improve generalization.
5. **Model Evaluation and Performance Metrics**:
    - Evaluate model performance using appropriate metrics such as accuracy, precision, recall, F1-score, mean squared error (MSE), or area under the ROC curve (AUC).
    - Compare the performance of different models and select the one that best meets the defined objectives and requirements.
6. **Model Interpretability and Explainability**:
    - Interpret model predictions and understand the factors influencing the outcomes. Utilize techniques such as feature importance analysis, SHAP (SHapley Additive explanations), or LIME (Local Interpretable Model-agnostic Explanations) for model explainability.
    - Ensure that the model's decisions align with domain knowledge and stakeholder expectations, enhancing trust and transparency.
7. **Hyperparameter Tuning and Optimization:**
    - Fine-tune model hyperparameters using techniques like grid search, random search, or Bayesian optimization to improve performance and generalization.
    - Optimize model architecture, regularization parameters, learning rates, and other hyperparameters to achieve better convergence and prevent overfitting.
8. **Model Deployment and Monitoring:**
    - Deploy the trained model into production environments, integrating it with existing systems or applications for real-time inference or batch processing.
    - Monitor model performance and drift over time, retraining or updating the model as necessary to maintain accuracy and relevance.
    - Implement robust logging, error handling, and version control mechanisms to ensure scalability, reliability, and maintainability of deployed models.

9. **Iterative Improvement and Feedback Loop:**

- Continuously monitor and evaluate model performance in production, soliciting feedback from users and stakeholders to identify areas for improvement.
- Iterate through the machine learning process, incorporating new data, refining models, and adapting strategies to evolving requirements and business objectives.

By following these steps, organizations can leverage the power of machine learning to extract valuable insights, automate decision-making processes, and drive innovation across various domains and industries.

3.4 Feature Selection

The process of selecting a subset of pertinent features or variables from the original set to be used in the construction of a model is known as feature selection, variable selection, or attribute selection. There are frequently more features than needed in real-world datasets, and some of them might not have a major impact on the model's ability to predict outcomes. Overfitting, noise introduction, and increased computational complexity can all result from these superfluous or redundant features. Feature selection tries to increase interpretability, decrease training time, boost the performance of the model, and enable better generalization to unknown data by limiting the selection to the most informative features.

The importance of feature selection in machine learning cannot be overstated. Here are some key reasons why feature selection is essential:

- **Improved Model Performance**: By focusing only on relevant features, feature selection helps improve the predictive accuracy and generalization capability of machine learning models.
- **Reduced Overfitting**: Including irrelevant features can cause the model to overfit to the training data, resulting in poor performance on unseen data. Feature selection mitigates overfitting by eliminating unnecessary complexity from the model.
- **Enhanced Interpretability**: Models with fewer features are easier to interpret and understand. Feature selection enables the identification of the most influential variables, leading to better insights into the underlying relationships in the data.
- **Computational Efficiency**: Reducing the number of features can significantly decrease the computational resources required for model training and inference, making the process more efficient.

3.4.1 Filter Methods

Filter methods evaluate the relevance of features based on their intrinsic properties, independent of the chosen machine learning algorithm. These methods typically involve statistical tests or heuristics to rank features according to their relevance to the target variable. Commonly used filter methods include:

- **Univariate Feature Selection**: This method evaluates each feature independently based on statistical tests such as chi-square test, ANOVA, or mutual information. Features with the highest scores are selected for further analysis.
- **Correlation-Based Feature Selection**: Correlation analysis measures the linear relationship between features and the target variable. Features with high correlation coefficients with the target variable are retained, while highly correlated features with each other are removed to reduce redundancy.
- **Information Gain and Entropy**: Information gain measures the reduction in uncertainty about the target variable after observing a feature. Features with high information gain are considered more relevant for classification tasks.

3.4.2 Wrapper Method

Wrapper methods select features by directly evaluating the performance of a machine learning algorithm using subsets of features. These methods typically involve a search strategy to explore different feature subsets and evaluate their performance. Common wrapper methods include:

- **Forward Selection**: Forward selection starts with an empty set of features and iteratively adds features that improve the model's performance the most until no further improvement is observed.
- **Backward Elimination**: Backward elimination begins with the full set of features and removes the least significant feature in each iteration until the model's performance deteriorates.
- **Recursive Feature Elimination (RFE)**: RFE recursively removes the least important features based on their coefficients or feature importance scores until the desired number of features is reached.

3.4.3 Embedded Method

Embedded methods integrate feature selection into the model training process, where feature selection occurs simultaneously with model training. These methods typically use regularization techniques to penalize the coefficients of irrelevant features or incorporate feature selection as part of the model architecture. Common embedded methods include:

- **Lasso (Least Absolute Shrinkage and Selection Operator)**: Lasso regression penalizes the absolute size of the regression coefficients, effectively shrinking coefficients of irrelevant features to zero and performing feature selection automatically.
- **Elastic Net**: Elastic Net combines the penalties of lasso and ridge regression, allowing for feature selection while handling multicollinearity among features.
- **Tree-Based Methods**: Decision tree algorithms such as Random Forest and Gradient Boosting automatically perform feature selection by selecting the most discriminative features at each node of the tree.

3.5 Evaluation Metrics for Feature Selection

Evaluation metrics serve as numerical measures utilized to evaluate the performance and efficiency of a statistical or machine-learning model. They offer valuable insights into the model's performance and aid in comparing various models or algorithms. Assessing a machine learning model involves evaluating its predictive accuracy, ability to generalize, and overall quality. Evaluation metrics offer objective standards to gauge these characteristics. The selection of evaluation metrics varies based on the problem domain, data type, and desired outcomes. Evaluating the effectiveness of feature selection methods requires appropriate performance metrics that reflect the model's predictive accuracy, complexity, and generalization capability. Common evaluation metrics for feature selection include:

- **Accuracy**: The proportion of correctly classified instances in the dataset.
- **Precision**: The ratio of true positive instances to the total number of instances predicted as positive.
- **Recall (Sensitivity)**: The ratio of true positive instances to the total number of actual positive instances.

- **F1 Score**: The harmonic means of precision and recall, balancing between precision and recall.
- **Area Under the Receiver Operating Characteristic Curve (AUC-ROC)**: Measures the trade-off between true positive rate and false positive rate across different threshold values.

3.6 Chapter Summary

Through this chapter, a range of statistical tests utilizing MapReduce and Spark frameworks are applied to pinpoint pertinent features from high-dimensional microarray data. The efficacy of these techniques is scrutinized atop a Hadoop cluster and juxtaposed with conventional systems. The identified features serve as building blocks for constructing an effective model to classify microarray data, a topic elaborated upon in subsequent chapters.

CHAPTER 4

FEATURE SELECTION OF MICROARRAY DATA USING FILTER-BASED METHODS

This chapter describes several filter-based feature selection techniques on scalable clusters. The techniques can handle and analyze distributed high-dimensional big data. These datasets are subjected to filter-based methods to identify pertinent features, which are then used for additional analysis.

4.1 Feature Selection on Microarray Data

The identification of informative genes responsible for cancer has been greatly aided in recent years by the DNA microarray technique. The curse of dimensionality problem, which reduces the value of information from a dataset and causes computational instability, is the main disadvantage of microarray data analysis. Consequently, a crucial first step in the process of effectively classifying cancer microarray datasets is the selection and extraction of pertinent features or genes [55].

Numerous researchers and practitioners have proposed numerous feature (gene) selection/extraction techniques in the literature [56]. In the meantime, assaying thousands or millions of genes at once has been made possible by recent advancements in microarray chip technology, producing a massive amount of data. Nevertheless, processing the data with standard computational power on a conventional system—where the data are stored on a standalone machine—is challenging.

Due to the massive increase in data generation and storage that has occurred in recent years, big data applications have drawn more and more attention. The challenge of extracting information from the large data pool arises from the fact that existing data mining techniques are not tailored to the new space and time constraints. Many paradigms, including MapReduce, Spark, and others, have been taken into consideration for creating scalable algorithms in order to address these issues. In the field of data science development, researchers have developed a wide range of pertinent, intelligent techniques that address imprecision, uncertainty, learning, and evolution in the formulation and resolution of computational problems [57].

Liang-jing Cai introduced the Var-CV-CHI algorithm, which builds upon the CHI algorithm by incorporating variance and coefficient of variation [58]. This study presents

a new approach to rectify the shortcomings of the CHI algorithm. The research encompasses experiments in text classification across two languages (Chinese and English), employing two distinct classifier algorithms (KNN and Naive Bayes), and analyzing two types of data distributions (balanced and unbalanced datasets). Importantly, it underscores significant enhancements in results when applying the proposed algorithm compared to those obtained using the original CHI algorithm. Noteworthy strengths of this research include its outstanding classification efficacy and its comprehensive range of experiments compared to previous studies. However, a limitation of this paper is its exclusive emphasis on the distribution of feature words, neglecting their semantic information.

The authors tackled the difficulty of choosing the right technique when there are no set standards for the latest scientific discoveries in the field. Their main goal was to conduct a robust experimental comparison to assess how feature selection procedures affect different classification algorithms. To achieve this, they explored both ranking-based feature-selection methods and innovative strategies, allowing for a comprehensive comparison. They presented thorough results from various conventional microarray datasets, covering a range of attributes and patient counts, which provide insights into the achievable outcomes through these methods [59].

The authors presented a $\chi 2$ feature selection method implemented with Apache Spark, utilizing a Scikit-learn algorithm, and evaluated its effectiveness on the Databricks platform [60]. The objective of this study is to reduce training time, improve accuracy, and address overfitting in the training data. Future endeavors will focus on exploring and integrating advanced linear algebra, feature selection techniques, and machine learning algorithms into the Apache Spark platform.

S. Bahassine introduced an improved method for Arabic text classification by employing Chi-square feature selection (ImpCHI) to enhance classification performance. In Arabic text classification, the abundance of relevant terms for each class is well understood, thus the Chi-square value is computed for each term across all classes [61]. The ImpCHI algorithm aims to achieve a balanced selection of attributes for each class. This paper's forthcoming research aims to investigate extending the concept of attribute balance across classes to other feature selection algorithms. In image analysis, a blood cancer image dataset consisting of 231 images was utilized. The DE-SVM classification exhibited an impressive accuracy of 98.55%, surpassing the conventional SVM's accuracy

of 86.96%. Moreover, while the regular NB achieved an accuracy of 95.6%, the DE-NB achieved a perfect accuracy score of 100%.

The statistical tests like t-tests, analysis of variance (ANOVA), analysis of covariance, Wilcoxon ranked sum test, linear regression, and generalized linear models such as binary logistic regression and chi-square test can be considered for features selection, but only three tests chi-square, ANOVA, and linear regression test are considered due to following advantages.

- For multiclass problem, t-test and Wilcoxon ranked sum test cannot be applied.
- For both two group and multiple group studies, its flexibility in handling data, its robustness with regard to data distribution, its ease of computation, the detailed information that can be derived from the test, and its use in studies for which parametric assumptions cannot be met.

As a result, just three tests—the chi-square, ANOVA, and linear regression tests—are taken into account when selecting features, meeting all the criteria for this research project.

4.2 Filter-based Feature Selection

Feature selection plays a crucial role in improving the machine-learning process and increasing the predictive power of algorithms. It involves selecting the most important variables while eliminating redundant and irrelevant features. The objectives of feature selection include improving prediction performance, providing faster and more cost-effective predictors, and gaining a better understanding of the underlying data-generating process. Filter-based feature selection, such as the Chi-Square, ANOVA, and linear regression method used in this study, analyzes the variance of gene expressions.

4.2.1 Chi-square Feature Selection

Feature selection is integral to enhancing the machine-learning process and bolstering the predictive capabilities of algorithms. This process entails identifying the most pertinent variables while discarding redundant and extraneous features. The aims of feature selection encompass enhancing prediction accuracy, furnishing swifter and more cost-effective predictors, and deepening comprehension of the underlying data-generating mechanisms.

In this study, filter-based feature selection, exemplified by the Chi-Square method, scrutinizes the variance of gene expressions. The Chi-square test, a statistical tool,

evaluates the independence of two events and is applicable to intricate contingency tables with multiple classes. It boasts robustness, dispensing with the need for uniform variances across study groups, and adeptly addressing class imbalances in microarray data. Moreover, it serves to assess the independence of gene variables [62].

The methodology includes the following steps:

- Given the data of two variables, it can get observed count O and expected count E.

$$E_i = \left(T_{c_i} * \frac{T_{t_i}}{\sum_{k=1}^{no:\ of\ classes} T_{c_i}} \right) \qquad \text{Equation (4.1)}$$

- T_{Ci} is the total number of values belonging to the class index i (Ci), T_{ti} is the total number of values belonging to the attribute at index i (ti) and i is 1 to the number of distinct instances.
- The modified expected value formula is calculated by multiplying the weight for individual classes as follows:

$$E_i = (T_{Ci} * T_{ti} / \sum_{k=1}^{no\ of\ classes} T_{Ci}) * T_{ti} \qquad \text{Equation (4.2)}$$

- Chi-Square measures how expected count E and observed count O deviate from each other.

$$\text{Chi-Square} = \sum_{i=1}^{no\ of\ instances} \frac{(O_i - E_i)^2}{E_i} \qquad \text{Equation (4.3)}$$

- O_i is the actual count of the instance at index i.
- Test can also be applied to a complex contingency table with several classes and as such is a very useful test in research work.
- To predict the class information for each sample in the test data, the chi-square value calculates the scores of this sample.
- To test the case that the calculated chi-square value is acceptable or not, calculated as follows.
- Calculate the Degree of freedom (D_f) and N is the total number of instances

$$D_f = N-1 \qquad \text{Equation (4.4)}$$

4.2.2 ANOVA Feature Selection

ANOVA (Analysis of Variance) feature selection is a statistical method commonly used in machine learning and data analysis to select the most relevant features from a dataset. ANOVA is particularly useful when dealing with continuous input variables and a categorical target variable [63].

Here's how ANOVA feature selection works:

1. **Calculate F-Scores**: ANOVA computes the F-score for each feature by analyzing the variance between the means of different groups and the variance within each group. It essentially compares the variation between group means to the variation within groups. A higher F-score indicates that the means of the groups are significantly different, suggesting that the feature is more relevant for predicting the target variable.
2. **Rank Features**: After calculating the F-scores for all features, they are ranked based on their scores. Features with higher F-scores are considered more relevant for predicting the target variable, while features with lower F-scores are considered less relevant.
3. **Select Features**: Based on a predefined threshold or criteria, a subset of features is selected for further analysis or model building. The threshold can be determined based on domain knowledge, experimentation, or using techniques such as cross-validation.
4. **Build Model**: Once the relevant features are selected, they can be used to build machine learning models for prediction, classification, or any other task.

ANOVA feature selection is advantageous because it considers the relationship between the input features and the target variable in terms of group means. It helps to identify the features that contribute the most to explaining the variance in the target variable and can lead to more efficient and accurate models by focusing on the most informative features. However, it's important to keep in mind that ANOVA feature selection assumes linear relationships between features and the target variable and may not perform optimally in all scenarios. Additionally, it may not capture complex interactions between features.

4.2.3 Linear Regression Feature Selection

Linear regression feature selection is a technique used to select the most relevant features from a dataset when building a linear regression model. The goal is to identify the subset of features that have the most significant impact on the prediction of the target variable [64].

Here's how linear regression feature selection works:

1. **Fit Linear Regression Model**: The process starts by fitting a linear regression model using all available features in the dataset. This initial model serves as a baseline for comparison.

2. **Evaluate Feature Importance**: After fitting the model, the importance of each feature is assessed. This can be done using various statistical metrics such as p-values, coefficients, or hypothesis tests.

- **P-values**: The p-value associated with each feature indicates the probability of observing the data given that the null hypothesis is true (i.e., the coefficient of the feature is zero). Lower p-values suggest that the feature is more statistically significant.
- **Coefficients**: The coefficients of the features in the linear regression equation represent the magnitude of their influence on the target variable. Larger coefficients indicate stronger associations with the target variable.

3. **Select Features**: Based on the evaluation of feature importance, features with high statistical significance or large coefficients are selected for inclusion in the final model. Alternatively, features with low importance may be removed from the model to improve simplicity and reduce overfitting.

4. **Refine Model**: Once the subset of features is selected, the linear regression model is re-fitted using only the selected features. This refined model is then evaluated to ensure that it meets the desired performance criteria.

Linear regression feature selection is advantageous because it provides insights into the relationships between features and the target variable in a linear context. It helps to identify the most relevant features for prediction and can improve model interpretability by focusing on the most influential variables.

However, linear regression feature selection may not capture complex nonlinear relationships between features and the target variable. In such cases, more advanced techniques like polynomial regression or machine learning algorithms may be more appropriate. Additionally, the selection of features based solely on statistical significance may overlook important interactions or correlations between variables. Therefore, it's essential to consider the specific characteristics of the dataset and the goals of the analysis when choosing a feature selection approach.

4.3 Classification Methods for Feature Selection

In most real-world classification tasks, supervised learning is necessary, where a lack of knowledge about the underlying probabilities of classes and their conditional probabilities. Each instance in such scenarios comes with a class label. Often, in these real-world settings have a limited understanding of relevant features. Hence, to reflect the domain accurately, numerous potential features are introduced. Consequently, irrelevant or redundant features may exist alongside the target concept. A relevant feature is one that is neither irrelevant nor redundant to the target concept. An irrelevant feature may not be directly linked to the target concept but influences the learning process, while a redundant feature does not contribute new information to the target concept. In many classification problems, constructing effective classifiers becomes challenging without eliminating these undesirable features, primarily due to the vast size of the data. Reducing the number of irrelevant or redundant features can significantly decrease the runtime of learning algorithms and result in a more generalized classifier. This process aids in gaining a deeper understanding of the fundamental concept behind real-world classification problems [64].

The framework for general feature selection in classification is illustrated in Figure 4.1. Feature selection primarily impacts the training phase of classification. Instead of feeding all features directly to the learning algorithm after feature generation, feature selection for classification involves initially choosing a subset of features and then passing the data with the selected features to the learning algorithm. The feature selection process can operate independently of the learning algorithm, such as with filter models, or it can iteratively assess the quality of selected features using the performance of the learning algorithms, as seen in wrapper models. Once the final set of features is determined, a classifier is trained for the prediction phase.

Classifier algorithms serve to assign labels or categories to input data based on the patterns and relationships learned from the training dataset. Each classifier has its unique characteristics and mechanisms for feature selection. Let's delve into the feature selection methods for Support Vector Machines (SVM), Random Forest, Logistic Regression, k-Nearest Neighbors (k-NN), Decision Trees, and Naive Bayes:

1. Support Vector Machines (SVM)

SVMs inherently perform feature selection by identifying the optimal hyperplane that maximizes the margin between different classes. Features that lie close to the decision boundary (support vectors) are crucial for defining the boundary and are thus selected.

SVMs can handle high-dimensional feature spaces effectively, making them robust to datasets with many features.

2. Random Forest

Random Forest employs feature selection as part of its ensemble learning process. During training, each decision tree in the forest is built using a subset of randomly selected features. Features that contribute the most to reducing impurity or increasing information gain at each split are favored. After training, feature importance can be computed based on how frequently they are used across all trees in the forest.

3. Logistic Regression

In Logistic Regression, feature selection is often done by analyzing the coefficients associated with each feature. Features with larger coefficients (in absolute terms) are considered more important in predicting the target variable. Regularization techniques such as L1 (Lasso) regularization can be employed to penalize the model for the inclusion of irrelevant features, effectively promoting feature selection during training.

4. K-Nearest Neighbors (k-NN)

K-NN does not inherently perform feature selection as part of its algorithm. However, feature selection techniques can be applied before using k-NN. For instance, feature selection methods like filter methods (e.g., correlation analysis) or wrapper methods (e.g., recursive feature elimination) can be employed to select the most relevant features before training the k-NN classifier.

5. Decision Trees

Decision Trees select features based on their ability to split the dataset into subsets that are more homogeneous concerning the target variable. Features that result in the greatest reduction in impurity (e.g., Gini impurity, entropy) are preferred for splitting. Decision Trees inherently perform feature selection during the tree-building process by selecting the most discriminative features at each node.

6. Naive Bayes

Naive Bayes classifiers assume that features are conditionally independent given the class label. As such, feature selection is less critical in Naive Bayes compared to other classifiers. However, in practice, feature selection techniques can still be applied to improve model performance and reduce computational complexity. Features that are strongly correlated with the class label are typically retained in Naive Bayes models.

Overall, while some classifiers inherently incorporate feature selection mechanisms (e.g., SVMs, Decision Trees, Random Forests), others require preprocessing

steps or additional techniques to select the most relevant features before training the model. The choice of feature selection method depends on the specific characteristics of the dataset and the performance requirements of the classification task.

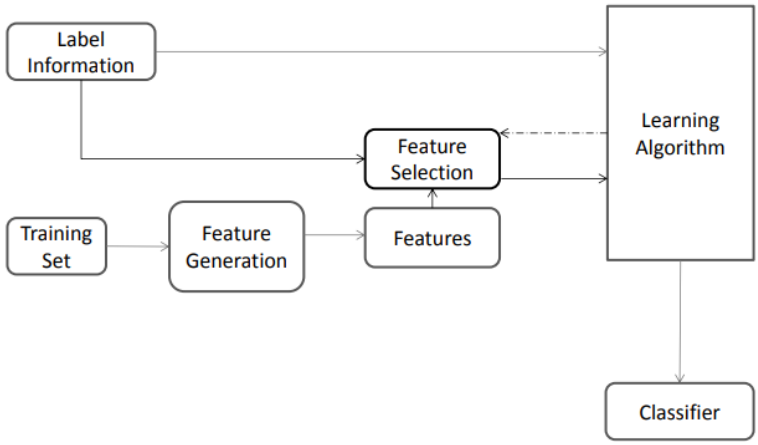

Figure 4.1 Feature Selection for Classification Framework

4.3.1 Classification over Apache Spark Framework

Apache Spark MLlib is a powerful machine learning library that provides scalable and distributed algorithms for various tasks, including classification. Apache Spark MLlib is a distributed machine learning library built on top of the Apache Spark platform. It provides a rich set of algorithms and utilities for scalable machine learning tasks, enabling users to leverage the distributed computing capabilities of Spark for large-scale data processing. Classification is a supervised learning task where the goal is to predict the categorical label of input data based on its features. It is widely used in various domains such as finance, healthcare, and marketing for tasks like spam detection, sentiment analysis, and customer segmentation [65].

To perform classification over Apache Spark using MLlib (Spark's machine learning library), user can use various algorithms such as Logistic Regression, Decision Trees, Random Forests, Gradient Boosted Trees, Support Vector Machines (SVM), and Naive Bayes.

- **Logistic Regression Classifier**: One well-liked technique for categorical response prediction is logistic regression. This particular instance of a Generalized Linear model forecasts the likelihood of the results. Binomial logistic regression and multinomial logistic regression are two methods of employing logistic regression in spark.ml to predict binary outcomes and multiclass outcomes, respectively. Logistic regression is used in Spark ML to predict multiclass outcomes using

multinomial logistic regression, or binary outcomes using binomial logistic regression. Multinomial logistic (softmax) regression facilitates multiclass classification. The conditional probabilities of the result classes, where k is a member of the set (1, 2,..., K), are modeled by the softmax function. These classifiers use a multinomial response model with an elastic-net penalty to reduce overfitting and minimize the weighted negative log-likelihood. Spark MLlib produces zero coefficients for constant nonzero columns when fitting a logistic regression model without an intercept on a dataset containing such columns.

- **Decision Tree Classifier**: A well-liked family of techniques for regression and classification are decision trees. Further details regarding the implementation of spark.ml are available in the decision tree section. Decision trees and related ensembles are popular methods for machine learning applications such as regression and classification. Decision trees are frequently employed because they can handle categorical features, capture non-linearities and feature interactions, extend to the multiclass classification scenario, and do not require feature scaling. Tree ensembles, which include boosting and random forests, are the best algorithms for problems involving classification and regression. The spark.ml implementation supports decision trees for binary and multiclass classification, regression, and both using continuous and categorical variables. The solution allows for distributed training over millions or even billions of instances by partitioning the data into rows.
- **Random Forest Classifier**: Random forests consist of ensembles of decision trees, representing one of the most successful machine learning models for both classification and regression tasks. By combining numerous decision trees, random forests effectively mitigate the risk of overfitting. Similar to decision trees, random forests accommodate categorical features, extend to multiclass classification scenarios, eliminate the need for feature scaling, and capture non-linear relationships and feature interactions. In Spark MLlib, random forests are supported for binary and multiclass classification, as well as for regression tasks, encompassing both continuous and categorical features. The implementation of random forests in Spark MLlib is built upon the existing decision tree infrastructure. For further details on trees, please refer to the decision tree guide.
- **Naive Bayes Classifiers:** represent a family of straightforward probabilistic classifiers, operating on the principles of Bayes' theorem while assuming strong

(naive) independence between each pair of features. Naive Bayes classifiers offer efficient training capabilities, requiring only a single pass over the training dataset to compute the conditional probability distribution of each feature given each label. During prediction, these classifiers apply Bayes' theorem to calculate the conditional probability distribution of each label given an observation. Many Naive Bayes classifier types are supported by MLlib, including Gaussian Naive Bayes, Bernoulli Naive Bayes, Multinomial Naive Bayes, and Complement Naive Bayes. For document classification problems, the Multinomial, Complement, and Bernoulli models are frequently used in terms of input data. Here, every observation usually stands for a document, and every feature for a phrase. For Multinomial or Complement Naive Bayes, the value of a feature might be the term frequency; for Bernoulli Naive Bayes, it could be a binary indication indicating if the term appears in the document. Multinomial and Bernoulli models require non-negative feature values. The model type can be specified using an optional parameter, with "multinomial" being the default choice. Sparse vectors are usually preferred for input feature vectors in document classification tasks. Given that the training data is used only once, caching it is unnecessary.

4.4 Chapter Summary

Spark has several benefits, which is the reason why different Spark-based classifiers are taken into consideration for high-dimensional data categorization in microarrays. This chapter presented the background theory of feature selection methods (ANOVA, Linear Regression, and Chi-square) and classification methods like (Logistic Regression, Random Forest, and Naïve Bayes).

CHAPTER 5

AN Upg-CHI BASED FEATURE SELECTION FRAMEWORK UNDER APACHE SPARK

Previous studies primarily focused on employing different filtering techniques to classify microarray data, often targeting a limited number of categories, usually only two or three. Microarray data encompass different diagnostic classifications, and achieving optimal performance metrics such as accuracy, precision, recall, and F1-score is desirable, especially when handling multiple classes, including Colon Tumor with two classes, Leukemia with three classes, Brain Tumor with four classes, and Brain Tumor with five classes. The challenge of feature selection in multiclass microarray data, aimed at addressing the curse of big dimensionality, presents a significant obstacle. Issues such as small record sizes, elevated proportions of dimension, and disparity in class are common hurdles. In microarray data, the distribution of relevant genes among classes is unknown, leading to class imbalance problems that can undermine accuracy in classification. The accuracy of classification tends to decrease quickly as the number of classes rises. A fundamental aspect of microarray data is the substantial volume of quantitative information, resulting in a large number of features relative to samples, which can lead to considerable computational costs in terms of algorithmic instability.

Dealing with the vast amount of record data poses a significant hurdle in gene classification. To deal with this matter, a method of selection feature is employed to weed out redundant traits and select the most relevant ones. This system introduces the algorithm of UpgCHI, an enhancement of the algorithm Chi-square test tailored for multiclass feature selection. UpgCHI is designed to tackle the difficulty of each class's undiscovered gene significance, improving upon the primary Chi-square method. The research leverages scalable platforms like Apache Spark for the analysis of microarray and established techniques implements with framework of Spark. Distributed and scalable frameworks like MapReduce and spark are utilized to implement machine learning techniques to examine multiclass microarray datasets with high dimensions. Consequently, the current approaches for analyzing microarray datasets have been modified for use with scalable platforms such as Apache Spark. [66].

5.1 Upgrade Chi-Square (UpgCHI) Method

In contrast to the conventional method chi-square, the proposed upgraded method chi-square algorithm in this system calculates the chi-square values for each class simultaneously. The chi-square values are calculated for each class in the microarray data where the gene expressions are unknown. A group of entity that are in the identical class are used to select the features with the highest chi-square values. The count is reached up to the user-specified threshold, and the top attributes are selected for each class according to ratio.

UpgCHI Test Algorithm

Input:

n = user threshold count

class_number = number of classes

attributes_per_class = number of attributes for each class

Output:

Selected_features = []

Step1: Define the hypothesis:

-Null Hypothesis (H0): Two variables are independent.

-Alternate Hypothesis (H1): Two variables are not independent.

For i = 1 to class_number:

Alist[i] = []

For k = 1 to attributes_per_class:

Step 2: Build a contingency table for each attribute.

contingency_table = build_contingency_table (class_i_attribute_k)

Step 3: Calculate the expected value (Ei) for each instance.

Ei = expected_value(contingency_table) according to equation (5.1)

$$E_i = \left(T_{C_i} * \frac{T_{t_i}}{\sum_{k=1}^{no:\ of\ classes} T_{C_i}} \right) \qquad \text{Equation (5.1)}$$

Step 4: Calculate the chi-square value of each instance according to equation (5.2).

chi_square_value = calculate_chi-square (contingency_table, Ei)

$$Chi\_Square = \sum_{i=1}^{no:\ of\ instances} \frac{(O_i - E_i)^2}{E_i}$$ Equation (5.2)

Step 5: Accept or Reject H0 df = degree_of_freedom(contingency_table) with alpha = 0.05 according to equation (5.3)

$$D_f = N-1$$ Equation (5.3)

if chi_square_value >= chi_square_value_from_distribution_table (df, alpha):
Reject the null hypothesis, accept the attribute, and save it.
Alist[i]. append ((class_i_attribute_k, chi_square_value))

```
# Step 6: Select the top n attributes for class i.
    Alist[i]. sort (key=lambda x: x [1], reverse=True) # Sort by chi-square value.
    Alist[i] = Alist[i] [: n]  # Select the top n attributes
# Step 7: Select the final features by using SelectByRatio.
Selected_features = SelectByRatio (Alist, n)
# SelectByRatio function.
def SelectByRatio(Alist, n):
    final_selected_features = []
    ratios = []
    for i in range (1, class_number + 1):
        class_id = i
        num_attributes = len(Alist[i])
        total_instances = sum([len(a) for a in Alist])
        ratio = (total_instances / num_attributes) * 100
        ratios.append((class_id, ratio))
    # Sort the ratios in descending order
    ratios.sort(key=lambda x: x[1], reverse=True)
    z = 0
    while len(final_selected_features) <= n:
        final_selected_features.append(Alist[z][0])
        z += 1
    return final_selected_features
```

5.2 Proposed System Architecture

The UpgCHI method operates on microarray data in which the starting expression of genes is unknown. Instead of directly using gene expressions, it computes the chi values for each class. The primary objective of the upgraded algorithm chi-square is to identify the most significant attributes. This is accomplished by computing the Chi-square value for every data point in every class. For every class c, the greatest Chi-square value (represented as max (Chi-square (i, c))) is kept as a chosen characteristic. Then, according to their Chi-square values, these chosen traits are arranged within each class. A set of top attributes is formed by selecting the top attributes, or those with the most significant values in each class.

A certain ratio value establishes how many top qualities are assigned to each class. The number of instances in each class is multiplied by 100 to determine this ratio, which is then divided by the total number of instances. Up to the user-specified threshold count, the algorithm keeps choosing characteristics for each class based on their ratio values. This method effectively finds and saves the most important characteristics for examination.

The uneven distribution of features among classes in microarray data is a common problem that can seriously hinder the classification process. Difficulties with classification may arise from improper normalization of the chosen attributes concerning their accurate representation per class. The upgrade chi algorithm is used to lessen this problem. By balancing the attribute selection between classes, this method lessens the detrimental effects of attribute dispersion and promotes a more accurate and effective classification procedure.

The proposed system employs filter-based chi-square feature selection techniques, with a specific focus on implementing the UpgCHI algorithm tailored for multiclass microarray data. UpgCHI has proven its robustness in accommodating varied data distributions, rendering it applicable across diverse scenarios. The system capitalizes on UpgCHI's strengths, offering computational efficiency, comprehensive test result insights, and adaptability to data from both binary and multiclass studies.

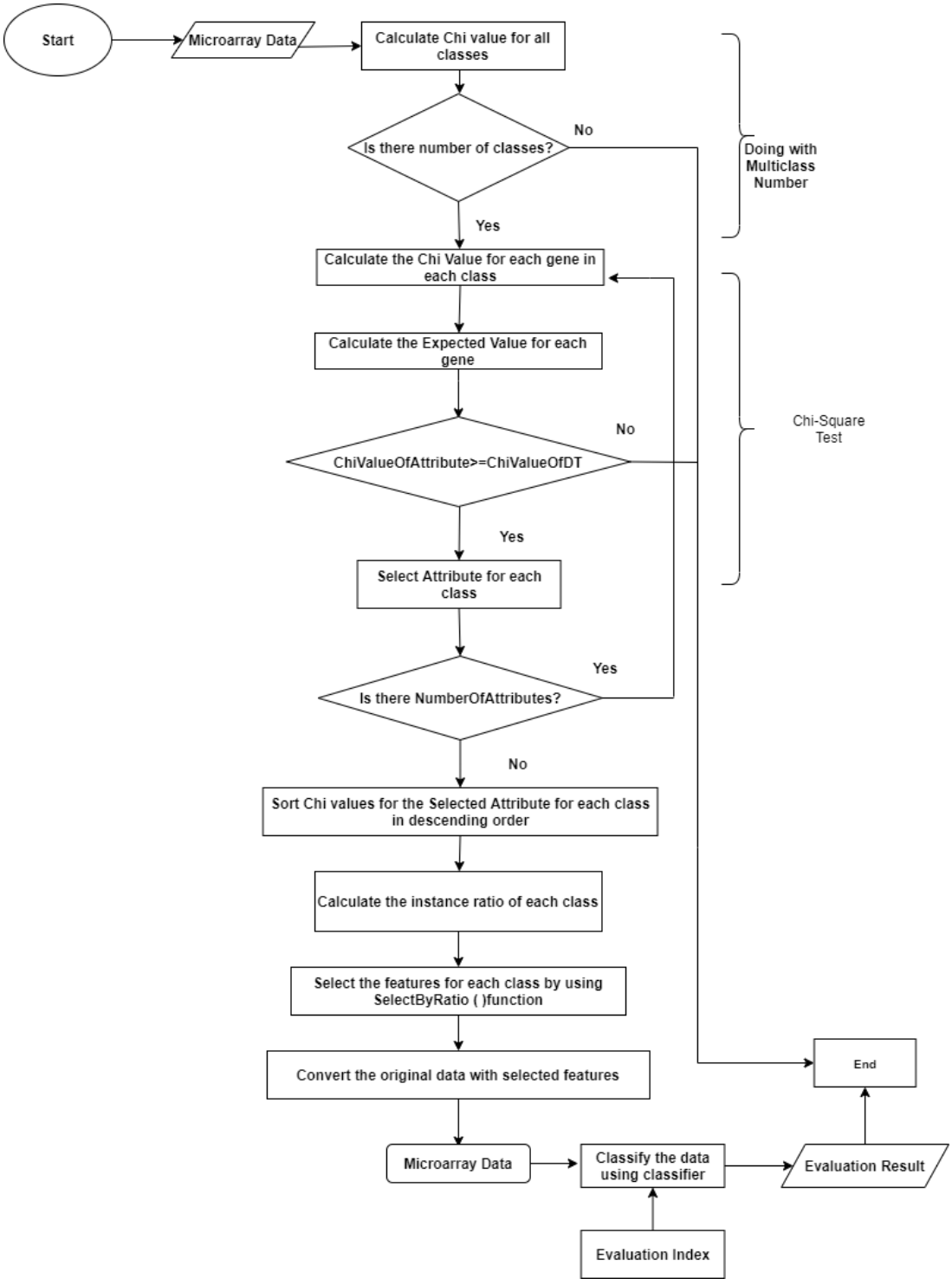

Figure 5.1 Proposed System Workflow Utilizing UpgCHI Test for Feature Selection

5.3 System Architecture on Apache Spark

The past few years have seen a significant increase in data collection and storage, which has turned attention towards Big Data applications. However, current data mining approaches are unable to keep up with the demands of rising space and time requirements, making it extremely difficult to extract significant information from this data. Novel paradigms are being investigated to create scalable algorithms to address these difficulties. This entails first identifying significant characteristics using feature selection techniques such as ANOVA, Chi-square, Linear Regression, and improved chi-square test (explained in Chapter 4) before taking into account several scalable classifiers for microarray dataset classification. By adjusting the high-dimensional data on a Hadoop cluster's data size, these methods' scalability is assessed.

The suggested algorithms' performance is assessed on Spark utilizing one master (driver) system and three slave (worker) nodes in terms of both execution time and accuracy. Next, a comparison is made between these outcomes and the outcomes from a traditional system. This chapter's main goal is to build a model that analyzes microarray data effectively. Based on performance metrics like recall, precision, F-Measure, accuracy, and time, the model's efficiency is evaluated. In short, the model should produce results with great accuracy and less time limitations. The classifier is enabled to perform as a scalable algorithm by utilizing the Spark framework, which operates in a distributed and scalable manner, to accomplish this goal.

This chapter offers and examines several classifiers built on the Spark framework, such as Random Forest (RF), Logistic Regression (LR), and Naive Bayes (NB). These classifiers are tested against a range of high-dimensional microarray datasets, and their performance is carefully assessed.

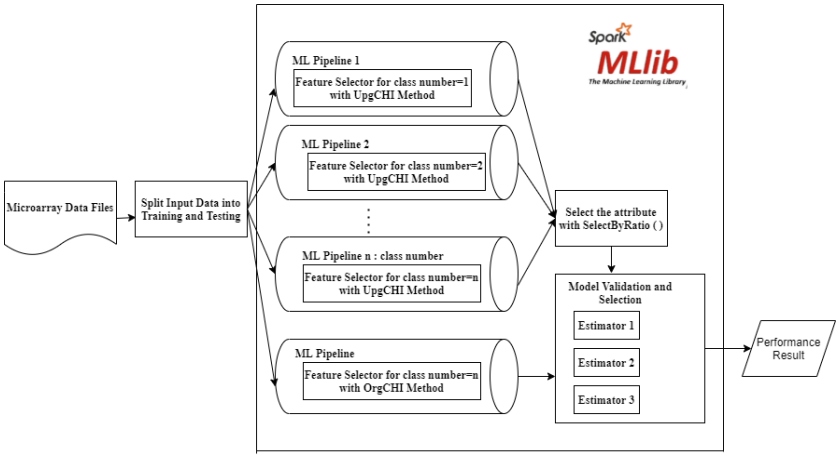

Figure 5.2 Workflow of Proposed System Implemented on Apache Spark

An estimator is an abstract description of any algorithm that is fitted or trained on data, such as a learning algorithm. As estimators, consider algorithms such as Logistic Regression. A variety of estimators are supported by Spark MLlib, such as the classifiers for logistic regression, decision trees, random forests, gradient-boosted trees, multilayer perceptron, linear support vector machines, One-vs-Rest (also called One-vs-All), Naive Bayes, and factorization machines. In this system, the three classifiers that are preferred to be used as estimators are Logistic Regression, Random Forest, and Naïve Bayes.

Logistic regression shows up as a flexible method for outcome prediction in the Spark ML setting. With binomial logistic regression or multinomial logistic regression, it makes it easier to predict binary outcomes or multiclass outcomes. Spark ML uses multinomial predictions for logistic (softmax) regression in classification jobs with several classes. Using the softmax function, this strategy involves modeling probabilities conditioned on outcome classes ($k \in 1, 2, ..., K$). Using a multinomial response model, the optimization approach involves minimizing the weighted negative log-likelihood. To avoid overfitting and improve the model's generalization skills, an elastic-net penalty is also used.

Random Forests are effective ensemble learning approaches because they use many decision trees to reduce the danger of overfitting. Random forests can handle multiclass classification, much as individual decision trees, which means feature scaling is not necessary. They are excellent at capturing intricate feature interactions and relationships. Random forests can be used in regression tasks and binary and multiclass classification tasks with Spark MLlib, which supports features in both continuous and categorical domains. Random Forests efficiently lower variance by combining predictions

from several trees, which improves performance on test data. Combining the forecasts from different trees makes the model more reliable and precise.

A type of straightforward probabilistic multiclass classifier known as a Naive Bayes classifier uses the Bayes theorem and assumes high (naive) independence between each pair of features. When trained with insufficient data, Naive Bayes classifiers perform better than other models. They are known for their quick processing, which greatly reduces the amount of time needed for both training and prediction. The performance metrics for the Spark-based logistic regression, Random Forest, and Naive Bayes classifier in the provided dataset are recall, precision, F-measure, and accuracy.

5.4 Chapter Summary

This chapter presented a number of classifiers that use the Spark framework to categorize big microarray datasets. The suggested approach works in a distributed manner on scalable clusters, improving efficiency with growing data volumes. The outcomes show that higher accuracy is attained by using the upgraded chi-square (UpgCHI) feature selection method.

CHAPTER 6
UPGRADE CHI-SQUARE FEATURE SELECTION AND SYSTEM IMPLEMENTATION

This chapter describes several scalable classifiers to classify high-dimensional data. The suggested classifiers can be integrated into any scalable cluster and operate in a distributed fashion. As the number of nodes in the cluster rises, the suggested classifiers become more scalable. To confirm the classifiers' scalability, a distributed and scalable framework named Spark is used on the Hadoop cluster's surface. The performance of these proposed classifiers is investigated with high-dimensional microarray data of different sizes.

6.1 Introduction of Proposed Works

Due to the massive rise in data collection and storage that has occurred over the past few years, big data applications have recently drawn more and more attention. The increasing space and time constraints make it difficult to extract information from these data using the data mining techniques that are already in use [2]. Scalable algorithms have been developed using the new paradigms in an attempt to overcome these obstacles.

A variety of scalable classifiers are taken into consideration to categorize the microarray datasets once the pertinent features have been chosen using feature selection techniques like ANOVA, Linear Regression, original chi test, and updated chi tests (as covered in Chapter 4). By altering the high-dimensional data on a Hadoop cluster's data size, the algorithms' scalability is evaluated. Three slave (worker) nodes and one master (driver) system are used to test the suggested algorithms' accuracy and execution times on Spark. The results are compared to those of a conventional system.

The primary goal of this chapter is to develop a model that performs an effective analysis of the microarray data. Recall, precision, F-Measure, accuracy, and other performance metrics are used to gauge how effective the model is. The distributed and scalable Spark framework is used to accomplish this purpose. Additionally, it gives the classifier knowledge of how to function as a scalable algorithm.

This chapter proposes several classifiers based on the Spark framework, including Naive Bayes, Logistic Regression, and Random Forest. The performance of the suggested classifiers is evaluated using a range of high-dimensional microarray datasets.

6.2 Results and Interpretation

This section examines the outcomes that the suggested algorithms produced using a variety of microarray datasets as input. Following feature selection, the datasets have been classified into the relevant classes using the suggested classification algorithm, which uses both a MapReduce and a Spark-based classifier. However, determining the ideal number of features needed for classification is challenging unless one has some prior knowledge about the dataset. The forward feature selection approach, which uses the highest-rated features that correspond to escalating p-values, is taken into consideration as a solution to this issue. The microarray dataset is classified using various subsets of the highest-ranked features using Spark-based classifiers, and the appropriate classification accuracies are calculated.

To validate the classifier, 5-fold cross-validation is applied to the training dataset. The performance of the final model is then evaluated using the testing dataset. The training dataset is used to train the model, and the testing dataset is used to gauge the model's performance. The model is trained using 5-fold cross-validation by varying λ and the overall performance is assessed. The overall execution details and processing efficiency of the Logistic Regression classifier using different four feature selector techniques are shown in Figure 6.1. The other Random Forest and Naïve Bayes classifiers are worked with 5-fold cross-validation and calculated the average classification accuracy. In addition, the other datasets are worked with 5-fold cross-validation by varying λ and the overall performance is assessed.

The classifier performance on different cluster nodes is investigated using a variety of case studies that employ varying datasets; these case studies are covered in the subsections that follow.

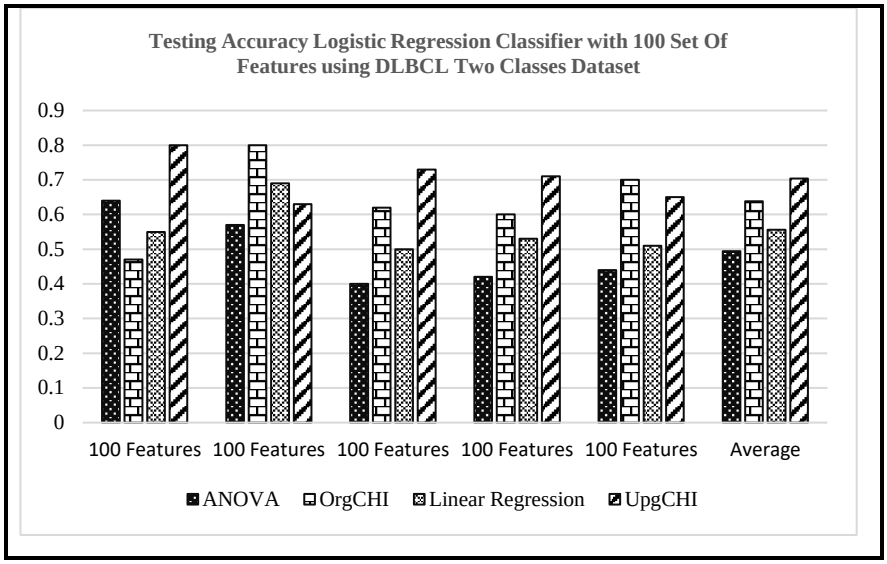

Figure 6.1 Testing Accuracy Results of Logistic Regression Classifier with 100 Features using DLBCL Two Classes Dataset

6.2.1 Result of DLBCL Two Classes Dataset

The dataset for breast cancer contains 77 samples and 5469 attributes. The performance study of several classifiers derived from various feature selection techniques is displayed in the following table. Figures 6.2 through 6.4 show the average classification accuracy of Logistic Regression, Random Forest, and Naïve Bayes classifier with 100 to 500 feature selection using ANOVA, Linear Regression, Chi-Square, and UpgCHI feature selectors.

Table 6.1 Performance Results for Logistic Regression Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.67	0.67	0.65
Linear Regression	0.7	0.7	0.7
Chi-square	0.69	0.61	0.65
UpgCHI	0.82	0.7	0.74

Table 6.2 Performance Results for Random Forest Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.57	0.58	0.57
Linear Regression	0.71	0.71	0.7
Chi-square	0.67	0.62	0.63
UpgCHI	0.78	0.72	0.74

Table 6.3 Performance Results for Naïve Bayes Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.81	0.7	0.71
Linear Regression	0.80	0.63	0.67
Chi-square	0.54	0.51	0.53
UpgCHI	0.82	0.72	0.74

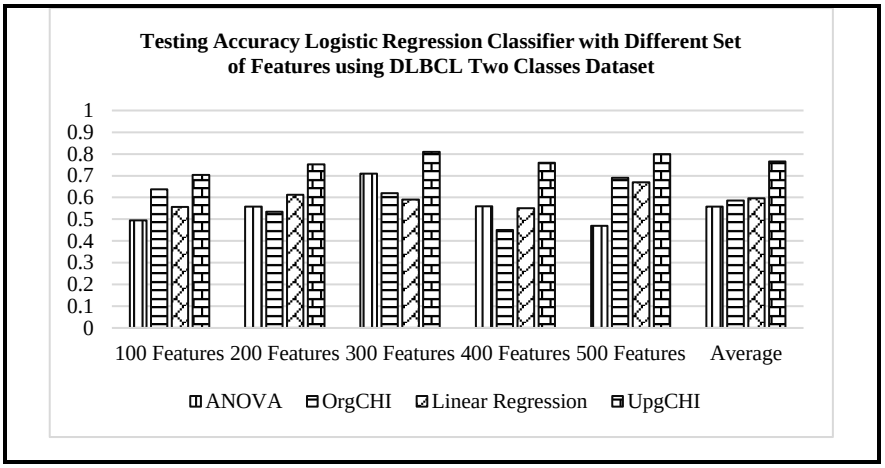

Figure 6.2 Testing Accuracy Results of Logistic Regression Classifier with Different Set of Features using DLBCL Two Classes Dataset

Figure 6.2 shows the accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using a Logistic Regression classifier on the DLBCL Dataset.

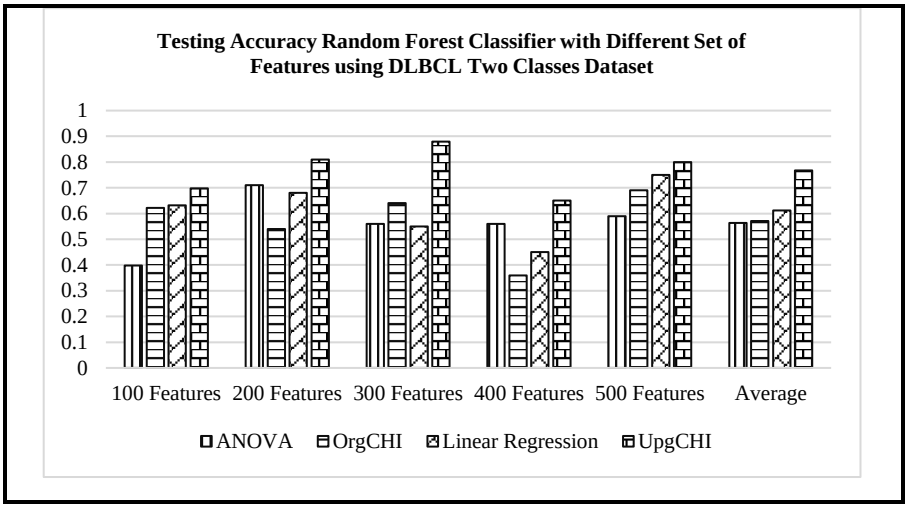

Figure 6.3 Testing Accuracy Results of Random Forest Classifier with Different Set of Features using DLBCL Two Classes Dataset

The accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using Random Forest classifier on DLBCL Dataset are shown in Figure 6.3.

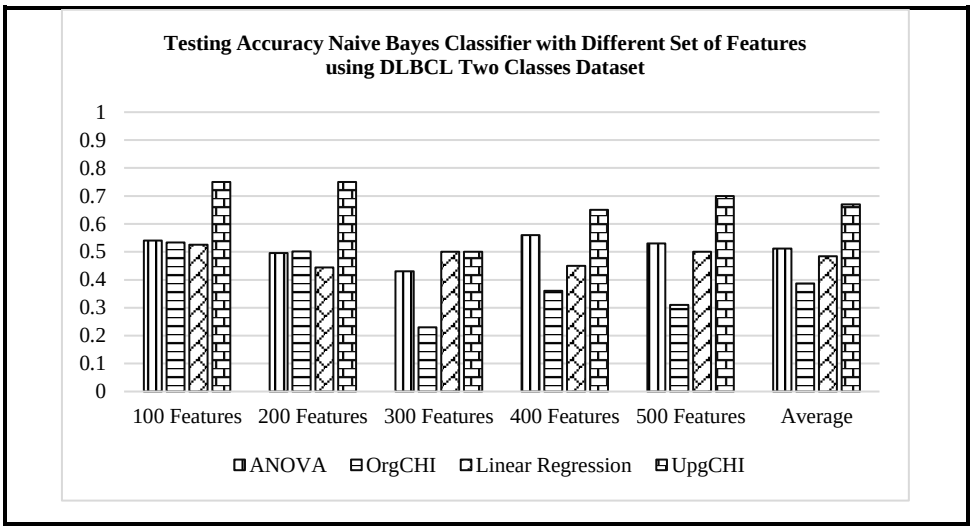

Figure 6.4 Testing Accuracy Results of Naïve Bayes Classifier with Different Set of Features using DLBCL Two Classes Dataset

The feature selector: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using Naïve Bayes classifier on DLBCL Dataset shown in Figure 6.4.

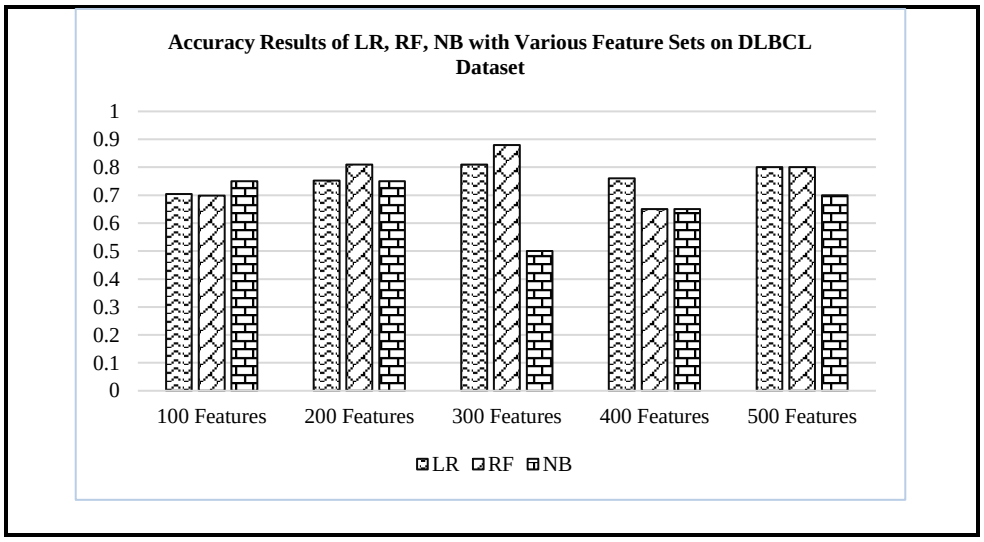

Figure 6.5 Testing Accuracy Results of Logistic Regression, Random Forest, Naïve Bayes Classifier with Various Feature Sets using DLBCL Dataset

Figure 6.5 shows the accuracy results of Logistic Regression, Random Forest, and Naïve Bayes classifier with various feature sets using DLBCL Dataset. The UpgCHI feature selection method with 300 features set and Random Forest classifier is suitable with an accuracy of 88%.

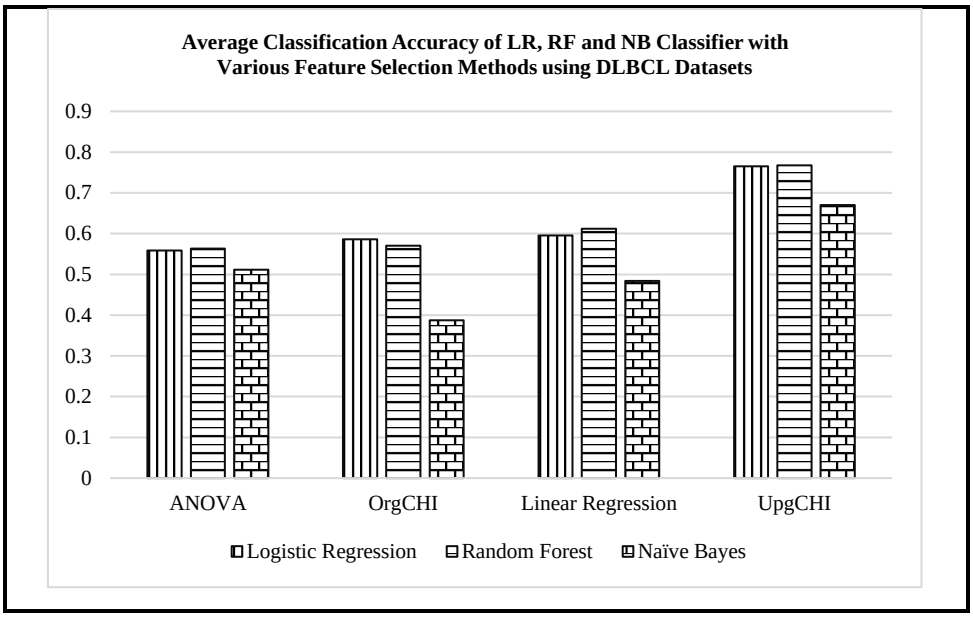

Figure 6.6 Testing Average Accuracy of LR, RF and NB classifier with Various Feature Selection Methods using DLBCL Dataset

Figure 6.6 shows the average accuracy results of Logistic Regression, Random Forest, and Naïve Bayes classifier with various feature sets using DLBCL Dataset. The UpgCHI feature selection method with the Random Forest classifier achieved an accuracy of 77%.

6.2.2 Result of Leukemia Three Classes Dataset

The dataset for breast cancer contains 72 samples and 7129 attributes. The performance study of several classifiers derived from various feature selection techniques is displayed in the following table. Figures 6.7 through 6.9 show the average classification accuracy of Logistic Regression, Random Forest, and Naïve Bayes classifier with 100 to 500 feature selection using ANOVA, Linear Regression, Chi-Square, and UpgCHI feature selectors.

Table 6.4 Performance Results for Logistic Regression Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.63	0.6	0.63
Linear Regression	0.65	0.6	0.61
Chi-square	0.89	0.8	0.85
UpgCHI	0.92	0.91	0.91

Table 6.5 Performance Results for Random Forest Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.54	0.5	0.54
Linear Regression	0.76	0.7	0.72
Chi-square	0.73	0.68	0.7
UpgCHI	0.75	0.7	0.73

Table 6.6 Performance Results for Naïve Bayes Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.68	0.63	0.64
Linear Regression	0.8	0.7	0.74
Chi-square	0.84	0.73	0.78
UpgCHI	0.9	0.81	0.85

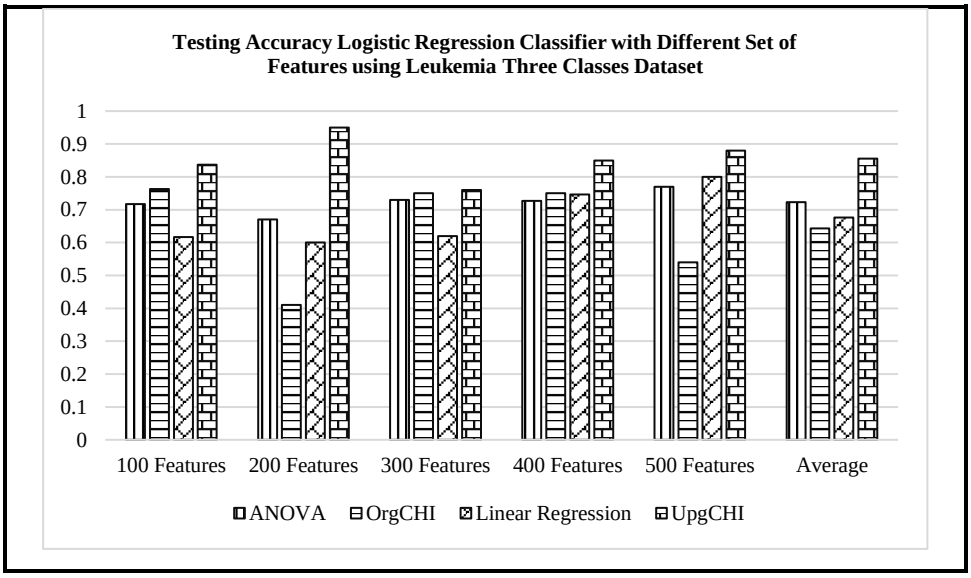

Figure 6.7 Testing Accuracy Results of Logistic Regression Classifier with Different Set of Features using Leukemia Three Classes Dataset

Figure 6.7 shows the accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using the Logistic Regression classifier on the Leukemia Dataset.

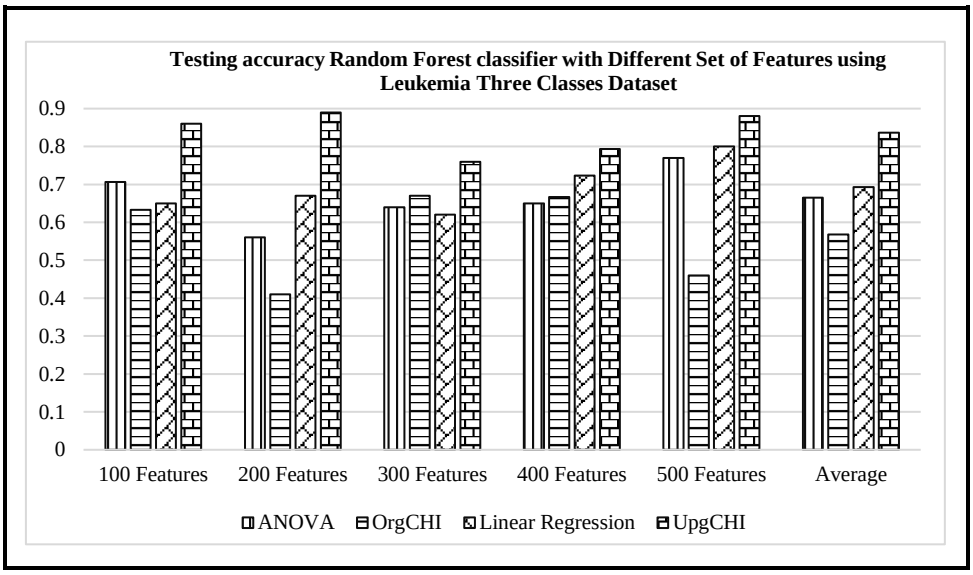

Figure 6.8 Testing Accuracy Results of Random Forest Classifier with Different Set of Features using Leukemia Three Classes Dataset

The accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using Random Forest classifier on DLBCL Dataset are shown in Figure 6.8.

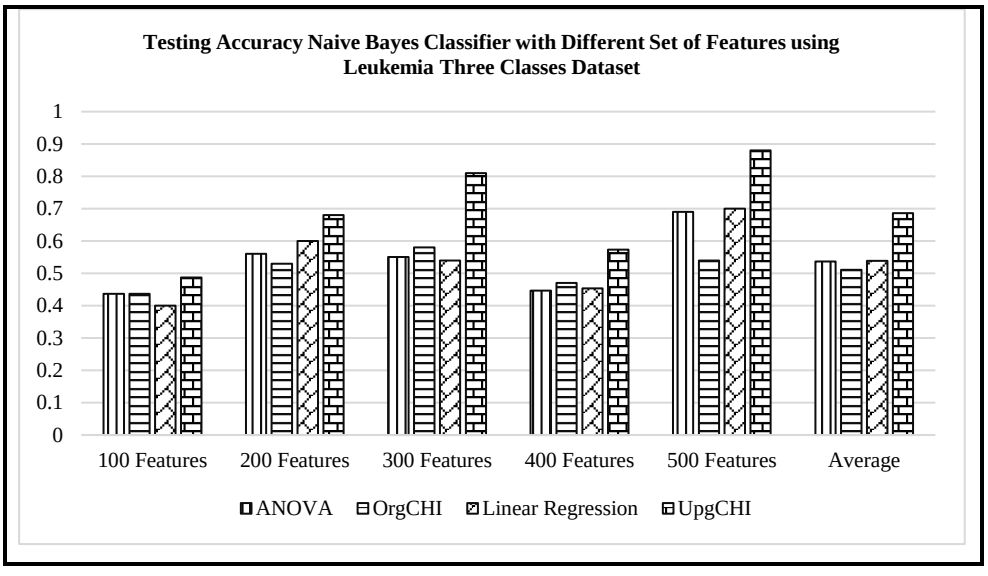

Figure 6.9 Testing Accuracy Results of Naive Bayes Classifier with Different Sets of Features using Leukemia Three Classes Dataset

The feature selector: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using Naïve Bayes classifier on the Leukemia Dataset shown in Figure 6.9.

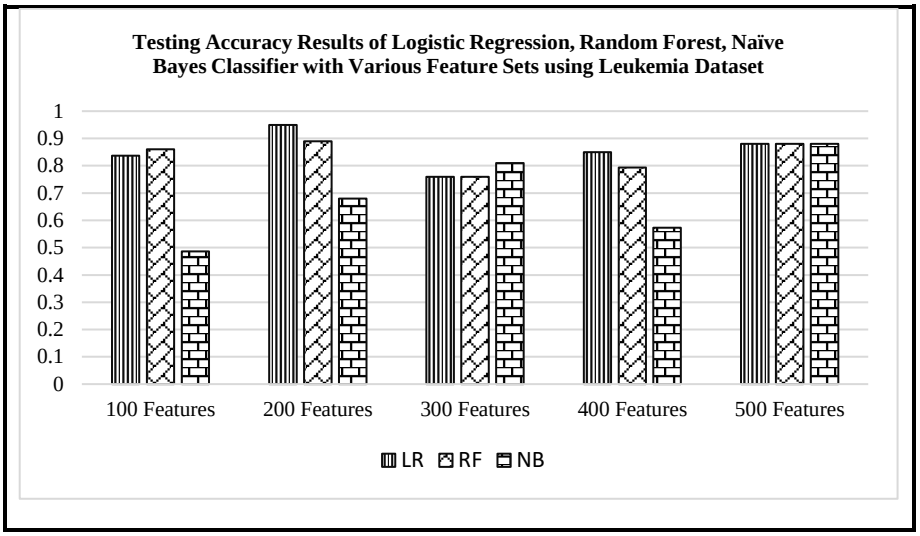

Figure 6.10 Testing Accuracy Results of Logistic Regression, Random Forest, Naïve Bayes Classifier with Various Feature Sets using Leukemia Dataset

Figure 6.10 shows the accuracy results of Logistic Regression, Random Forest, and Naïve Bayes classifier with various feature sets using DLBCL Dataset. The UpgCHI feature selection method with 200 features set and a Logistic Regression classifier is suitable with an accuracy of 95%.

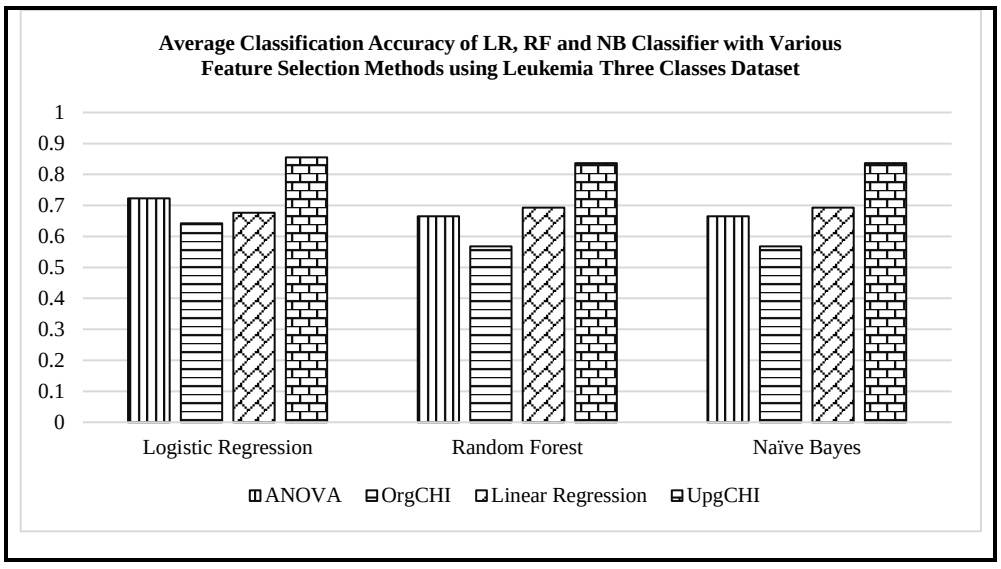

Figure 6.11 Testing Accuracy of LR, RF, and NB classifier with Various Feature Selection Methods using Leukemia Three Classes Dataset

Figure 6.11 shows the average accuracy results of the Logistic Regression, Random Forest, and Naïve Bayes classifier with various feature sets using the Leukemia three classes dataset. The UpgCHI feature selection method with the Logistic Regression classifier achieved an accuracy of 85%.

6.2.3 Result of Brain Tumor Four Classes Dataset

The Brain Tumor dataset has 50 samples and 10367 attributes. The performance measure of the different classifiers derived from various feature selection techniques is displayed in the following Tables. The classifier's accuracy either stays steady after reaching its peak or starts to decline. To circumvent the issue of the curse of dimensionality, the microarray datasets are analyzed through the usage of different classifiers, and the classifier's various performance parameters are assessed.

Table 6.7 Performance Results for Logistic Regression Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.47	0.42	0.43
Linear Regression	0.34	0.3	0.34
Chi-square	0.5	0.3	0.36
UpgCHI	0.64	0.6	0.61

Table 6.8 Performance Results for Random Forest Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.55	0.44	0.44
Linear Regression	0.53	0.53	0.53
Chi-square	0.46	0.35	0.37
UpgCHI	0.57	0.51	0.52

Table 6.9 Performance Results for Naïve Bayes Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.37	0.3	0.33
Linear Regression	0.47	0.39	0.41
Chi-square	0.44	0.5	0.49
UpgCHI	0.57	0.5	0.54

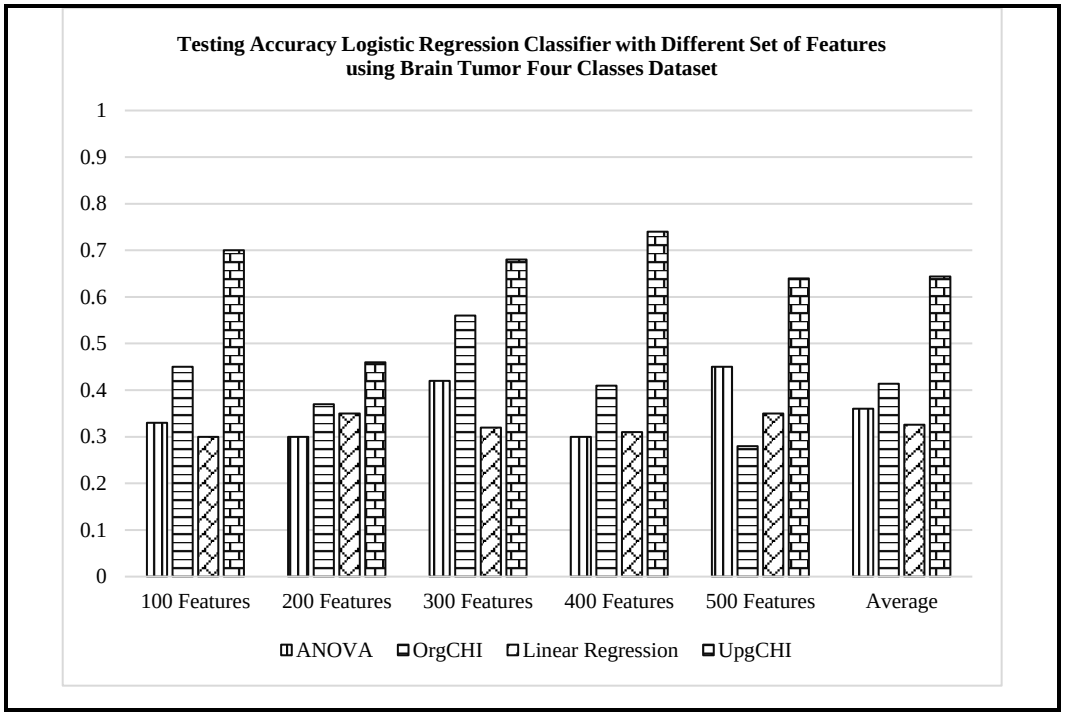

Figure 6.12 Testing Accuracy Results of Logistic Regression Classifier with Different Set of Features using Brain Tumor Four Classes Dataset

Figure 6.12 shows the accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using a Logistic Regression classifier on the Brain Tumor Dataset.

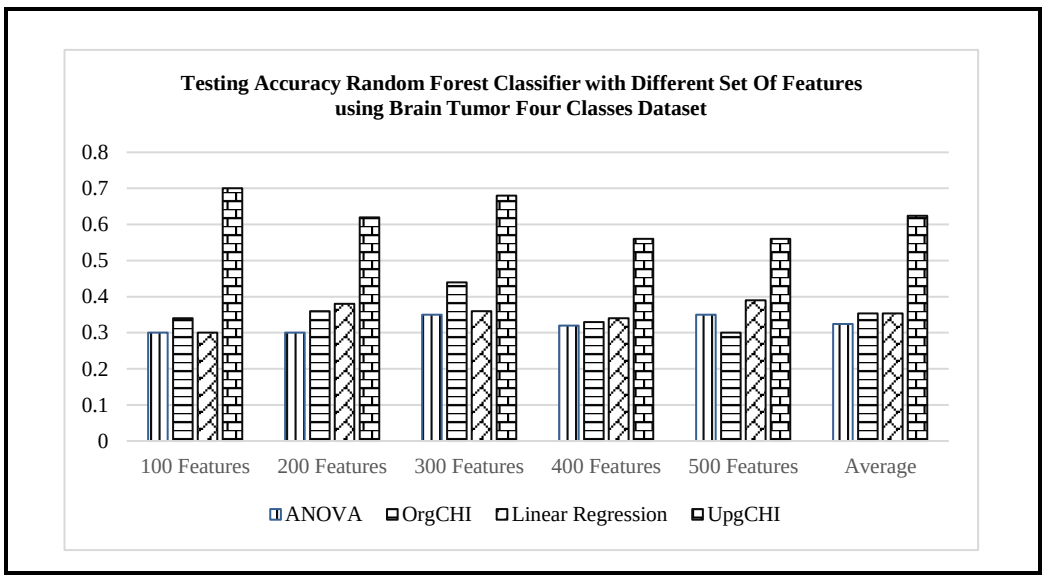

Figure 6.13 Testing Accuracy Results of Random Forest Classifier with Different Set of Features using Brain Tumor Four Classes Dataset

The accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using the Random Forest classifier on the Brain Tumor Dataset are shown in Figure 6.13.

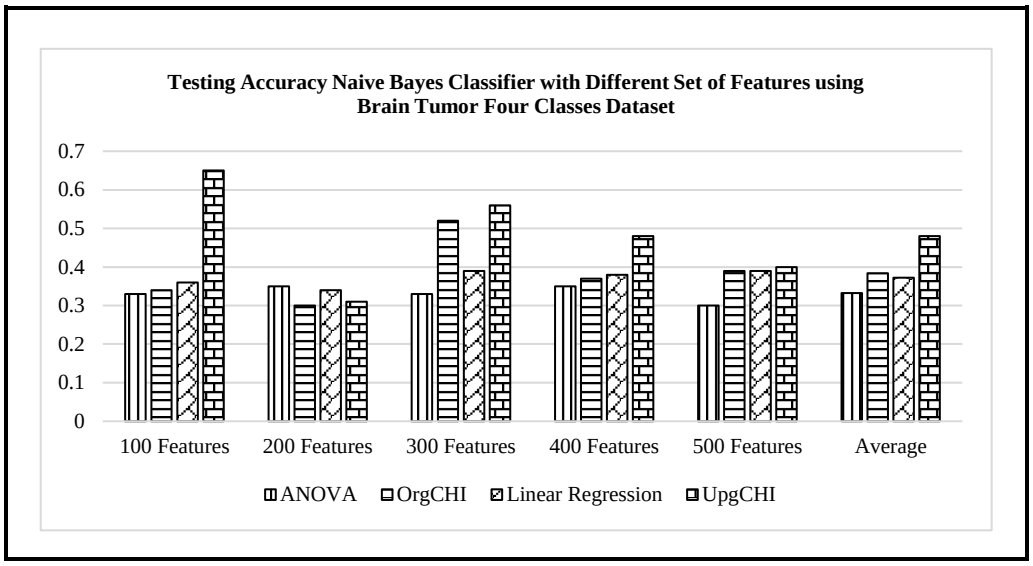

Figure 6.14 Testing Accuracy Results of Naive Bayes Classifier with Different Set of Features using Brain Tumor Four Classes Dataset

The accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using the Naïve Bayes classifier on the Brain Tumor Dataset are shown in Figure 6.14.

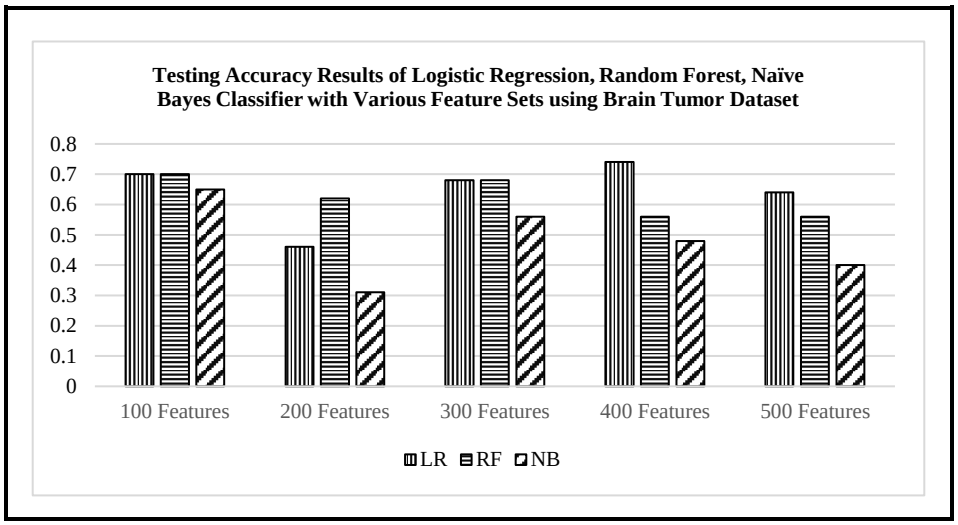

Figure 6.15 Testing Accuracy Results of Logistic Regression, Random Forest, Naïve Bayes Classifier with Various Feature Set using Brain Tumor Dataset

Figure 6.15 shows the accuracy results of Logistic Regression, Random Forest, and Naïve Bayes classifier with various feature sets using Brain Tumor Dataset. The UpgCHI feature selection method with 400 features set and a Logistic classifier is suitable with an accuracy of 74%.

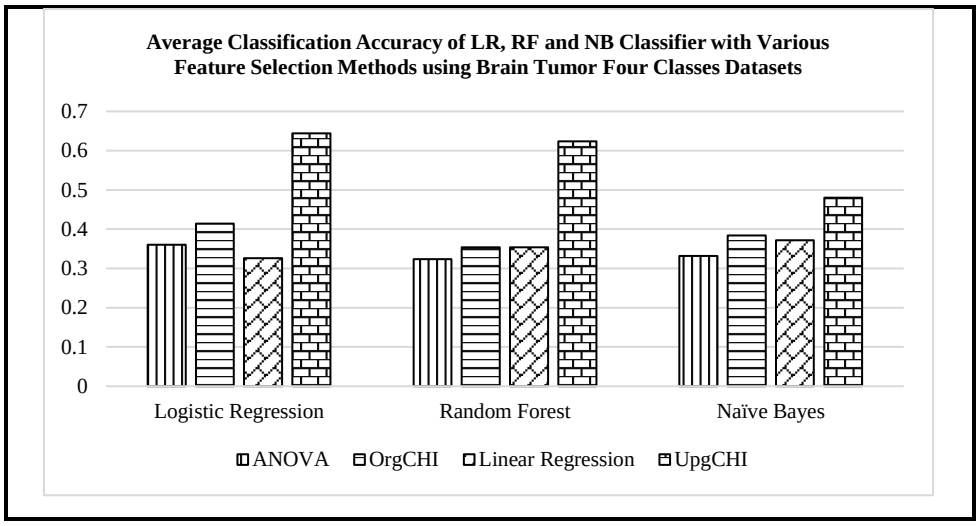

Figure 6.16 Testing Accuracy of LR, RF, and NB classifier with Various Feature Selection Methods using Brain Tumor Four Classes Dataset

Figure 6.16 shows the average accuracy results of Logistic Regression, Random Forest, and Naïve Bayes classifier with various feature sets using DLBCL Dataset. The UpgCHI feature selection method with a Logistic classifier achieved an accuracy of 64%.

6.2.4 Result of Leukemia Four Classes Dataset

The dataset for breast cancer contains 72 samples and 7129 attributes. The performance study of several classifiers derived from various feature selection techniques is displayed in the following table. Figures 6.17 through 6.19 show that the average classification accuracy of Logistic Regression, Random Forest and Naïve Bayes classifier with 100 to 500 feature selection using ANOVA, Linear Regression, Chi-Square and UpgCHI feature selectors.

Table 6.10 Performance Results for Logistic Regression Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.69	0.76	0.72
Linear Regression	0.77	0.8	0.77
Chi-square	0.68	0.66	0.61
UpgCHI	0.9	0.94	0.97

Table 6.11 Performance Results for Logistic Regression Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.5	0.52	0.51
Linear Regression	0.61	0.6	0.6
Chi-square	0.4	0.41	0.4
UpgCHI	0.9	0.88	0.88

Table 6.12 Performance Results for Naïve Bayes Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.55	0.64	0.59
Linear Regression	0.71	0.73	0.71
Chi-square	0.51	0.5	0.48
UpgCHI	0.97	0.83	0.87

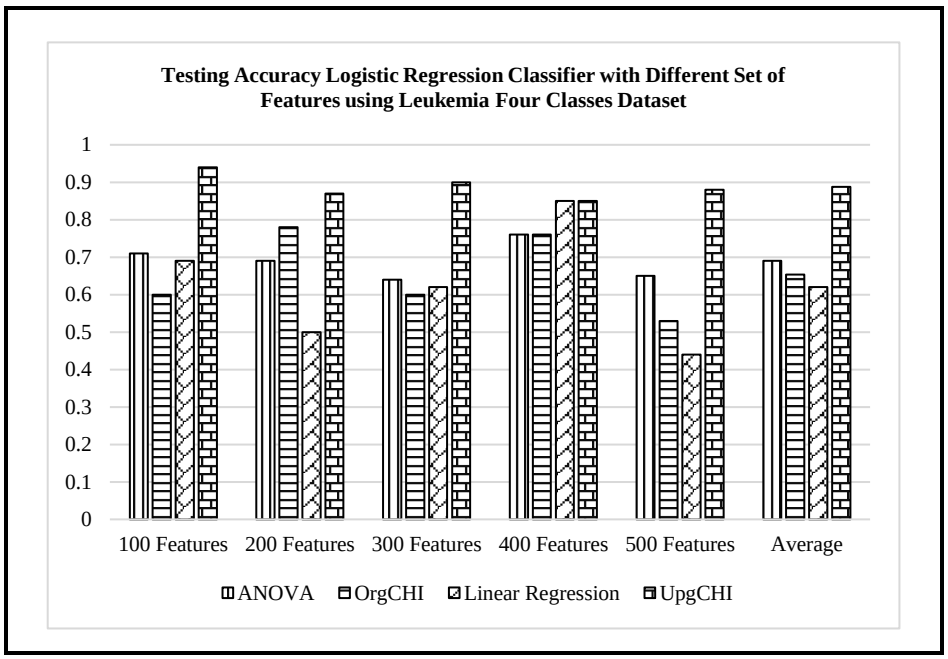

Figure 6.17 Testing Accuracy Results of Logistic Regression Classifier with Different Set of Features using Leukemia Four Classes Dataset

Figure 6.17 shows the accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using the Logistic Regression classifier on the Leukemia Dataset.

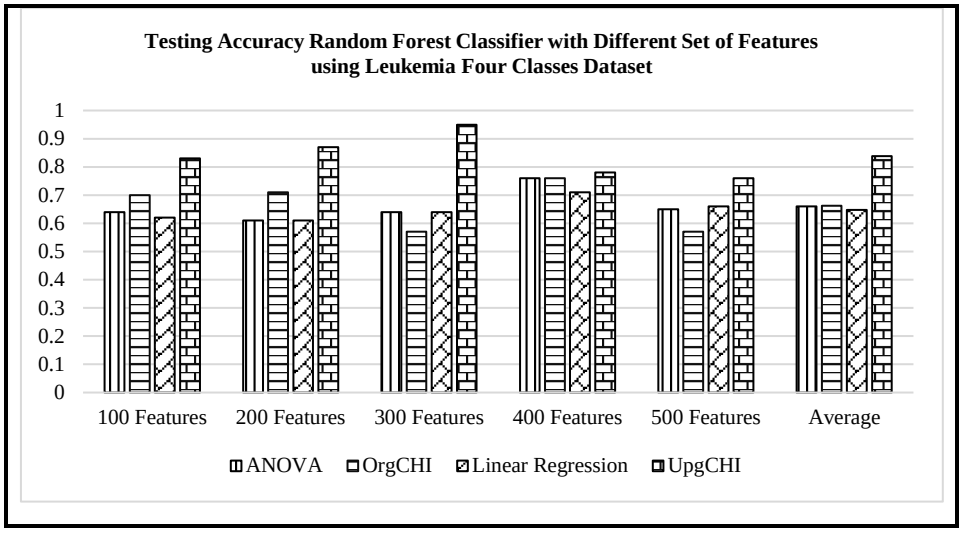

Figure 6.18 Testing Accuracy Results of Random Forest Classifier with Different Set of Features using Leukemia Four Classes Dataset

The accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using the Random Forest classifier on the Leukemia Dataset are shown in Figure 6.18.

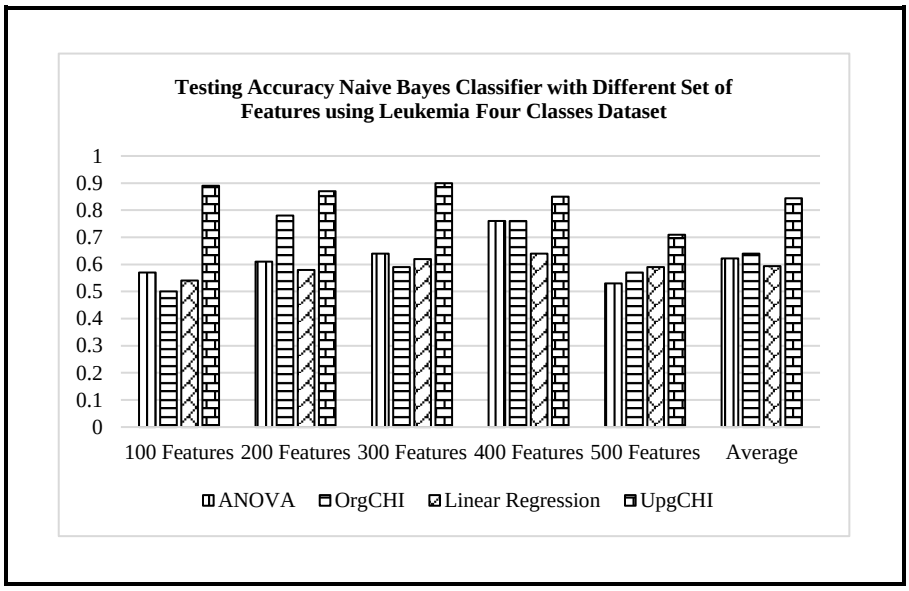

Figure 6.19 Testing Accuracy Results of Naive Bayes Classifier with Different Set of Features using Leukemia Four Classes Dataset

The accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using the Naïve Bayes classifier on the Leukemia Dataset are shown in Figure 6.19.

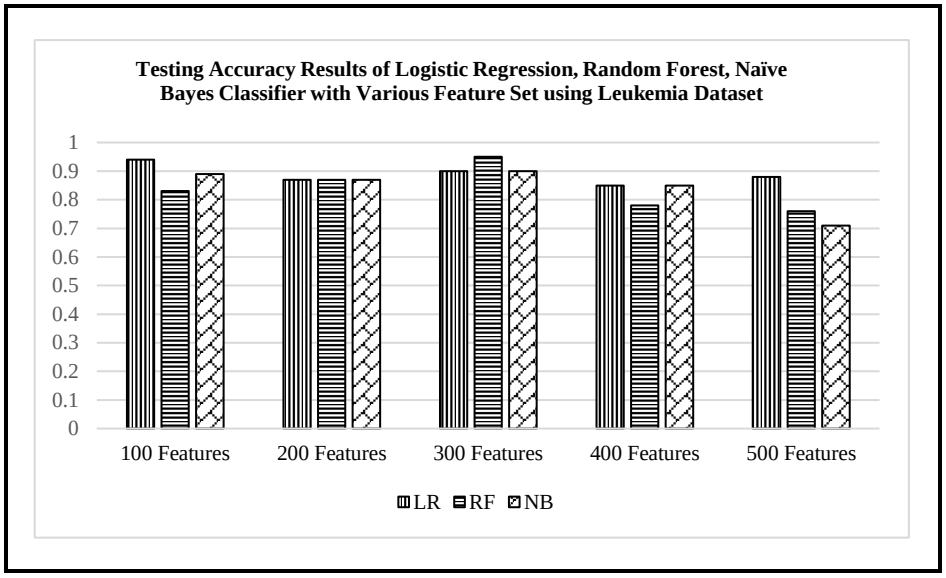

Figure 6.20 Testing Accuracy Results of Logistic Regression, Random Forest, Naïve Bayes Classifier with Various Feature Sets using Leukemia Dataset

Figure 6.20 shows the accuracy results of Logistic Regression, Random Forest, and Naïve Bayes classifier with various feature sets using DLBCL Dataset. The UpgCHI feature selection method with 300 feature sets and Random Forest classifier is more suitable with an accuracy of 95%.

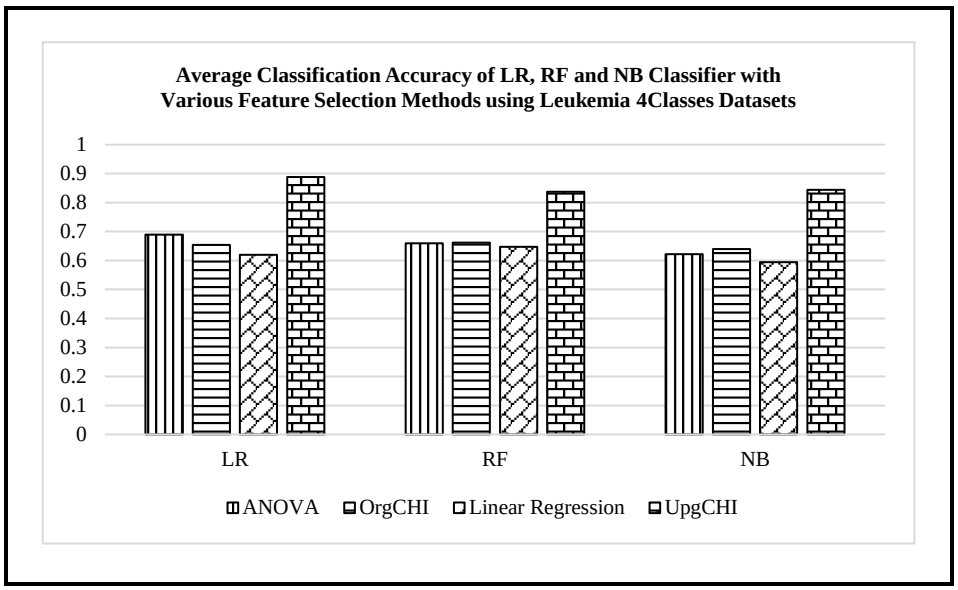

Figure 6.21 Testing Accuracy of LR, RF, and NB Classifier with Various Feature Selection Methods using Leukemia Four Classes Dataset

Figure 6.21 shows the average accuracy results of Logistic Regression, Random Forest, and Naïve Bayes classifier with various feature sets using the Leukemia Dataset. The UpgCHI feature selection method with the Logistic Regression classifier achieved an accuracy of 88%.

6.2.5 Result of Brain Tumor Five Classes Dataset

There are 5920 attributes and 90 samples in the Brain Tumor dataset. The performance measure of the different classifiers derived from various feature selection techniques is displayed in the following Tables. The performance study of several classifiers derived from various feature selection techniques is displayed in the following table. Figures 6.22 through 6.24 show the average classification accuracy of Logistic Regression, Random Forest, and Naïve Bayes classifier with 100 to 500 feature selection using ANOVA, Linear Regression, Chi-Square, and UpgCHI feature selectors.

Table 6.13 Performance Results for Logistic Regression Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.36	0.31	0.34
Linear Regression	0.5	0.3	0.3
Chi-square	0.45	0.36	0.39
UpgCHI	0.56	0.5	0.53

Table 6.14 Performance Results for Random Forest Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.35	0.32	0.34
Linear Regression	0.4	0.3	0.3
Chi-square	0.56	0.54	0.53
UpgCHI	0.7	0.55	0.61

Table 6.15 Performance Results for Naïve Bayes Classifier

Algorithms	Precision	Recall	F1-score
ANOVA	0.41	0.36	0.38
Linear Regression	0.36	0.34	0.34
Chi-square	0.58	0.54	0.57
UpgCHI	0.68	0.62	0.63

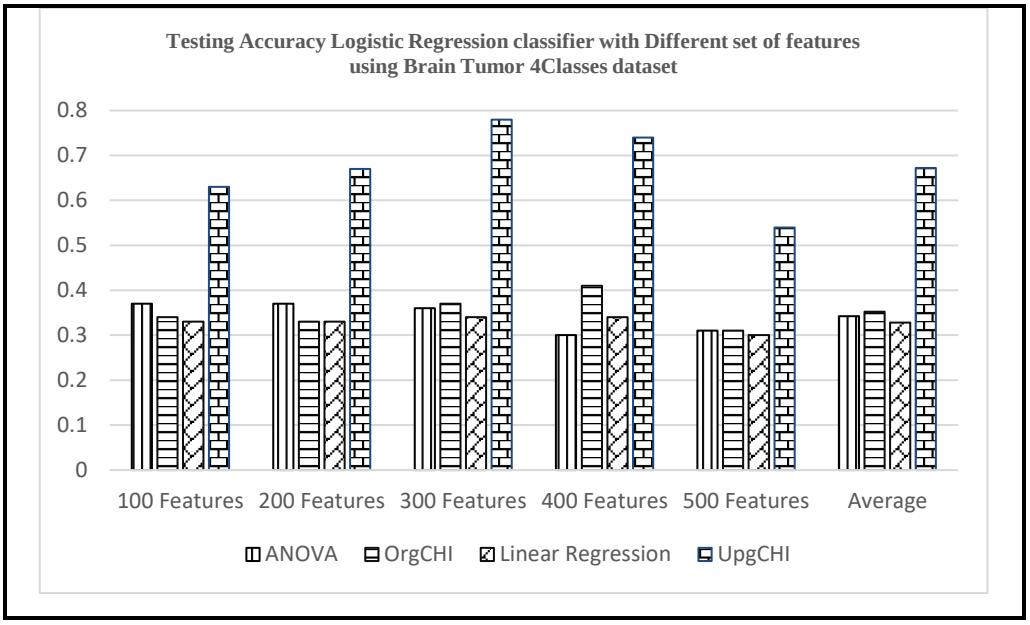

Figure 6.22 Testing Accuracy Results of Logistic Regression Classifier with Different Set of Features using Brain Tumor Five Classes Dataset

The accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression and ANOVA with 100 to 500 features using Logistic Regression classifier on Brain Tumor Dataset are shown in Figure 6.22.

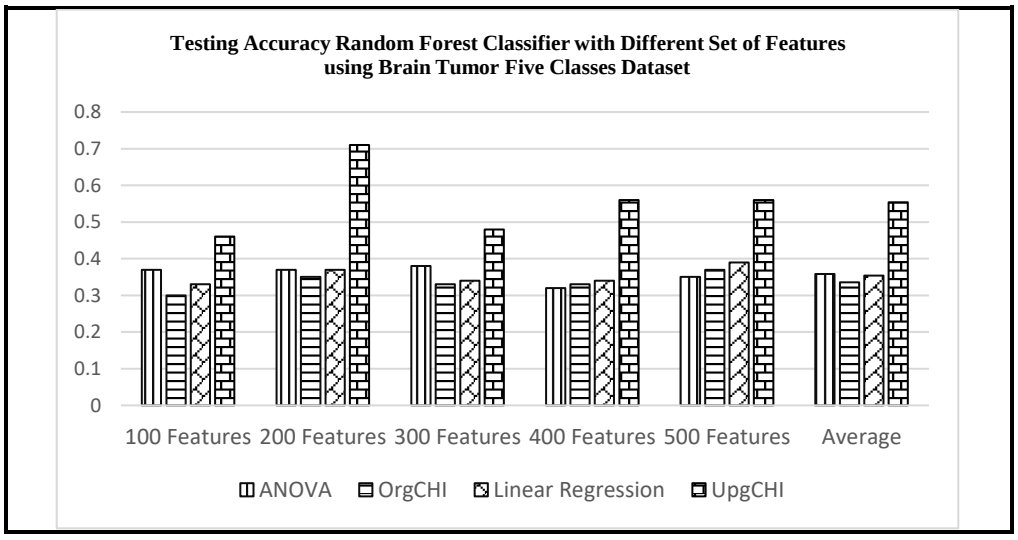

Figure 6.23 Testing Accuracy Results of Random Forest Classifier with Different Set of Features using Brain Tumor Five Classes Dataset

The accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression, and ANOVA with 100 to 500 features using the Random Forest classifier on the Brain Tumor Dataset are shown in Figure 6.23.

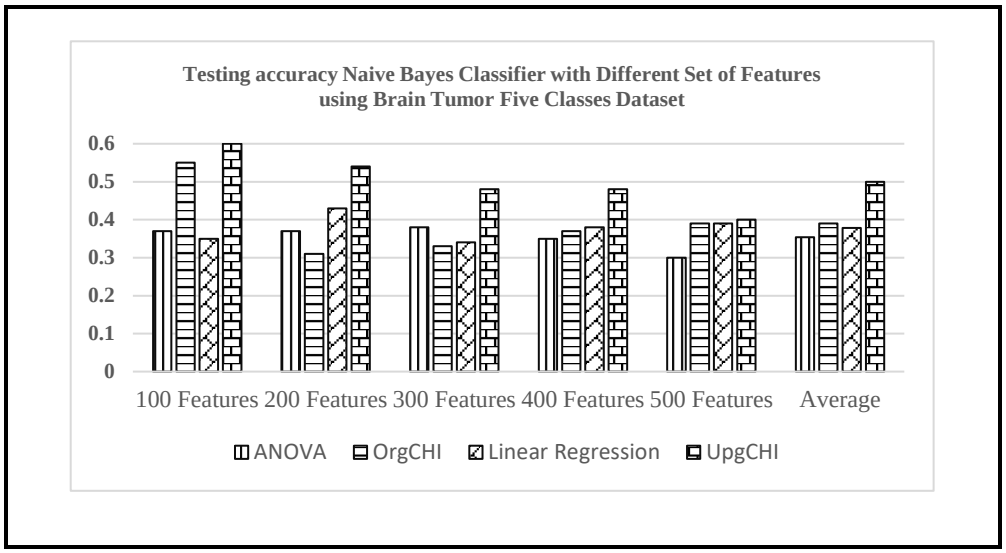

Figure 6.24 Testing Accuracy Results of Naive Bayes Classifier with Different Set of Features using Brain Tumor Five Classes Dataset

The accuracy results of four feature selection methods: UpgCHI, Chi-square, Linear Regression and ANOVA with 100 to 500 features using Naïve Bayes classifier on Brain Tumor Dataset are shown in Figure 6.24.

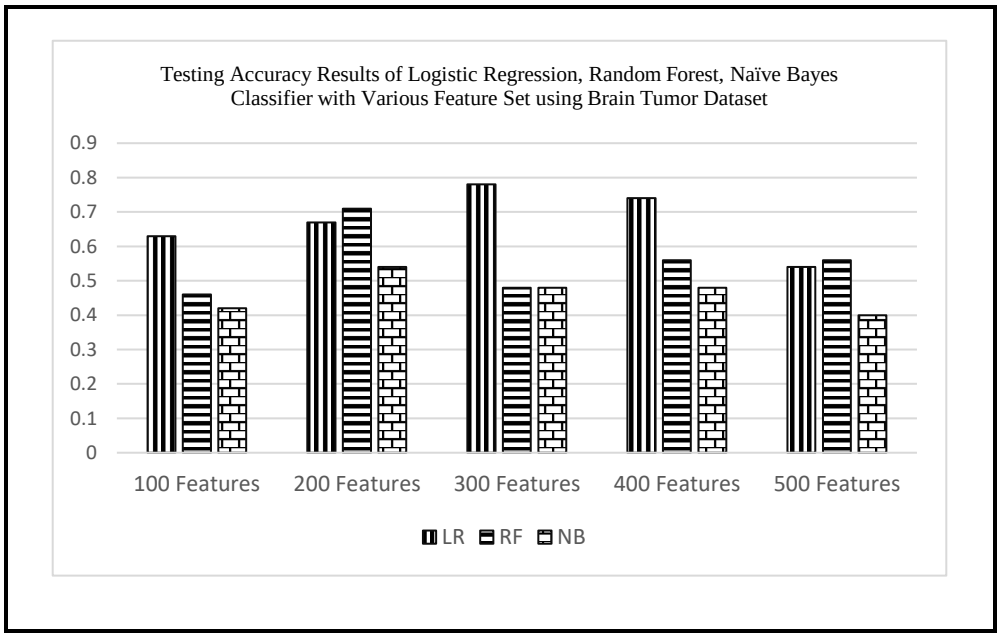

Figure 6.25 Testing Accuracy Results of Logistic Regression, Random Forest, Naïve Bayes Classifier with Various Feature Sets using Brain Tumor Dataset

Figure 6.25 shows the accuracy results of Logistic Regression, Random Forest, and Naïve Bayes classifier with various feature sets using Brain Tumor Dataset. The UpgCHI feature selection method with 300 features set and a Logistic Regression classifier is suitable with an accuracy of 78%.

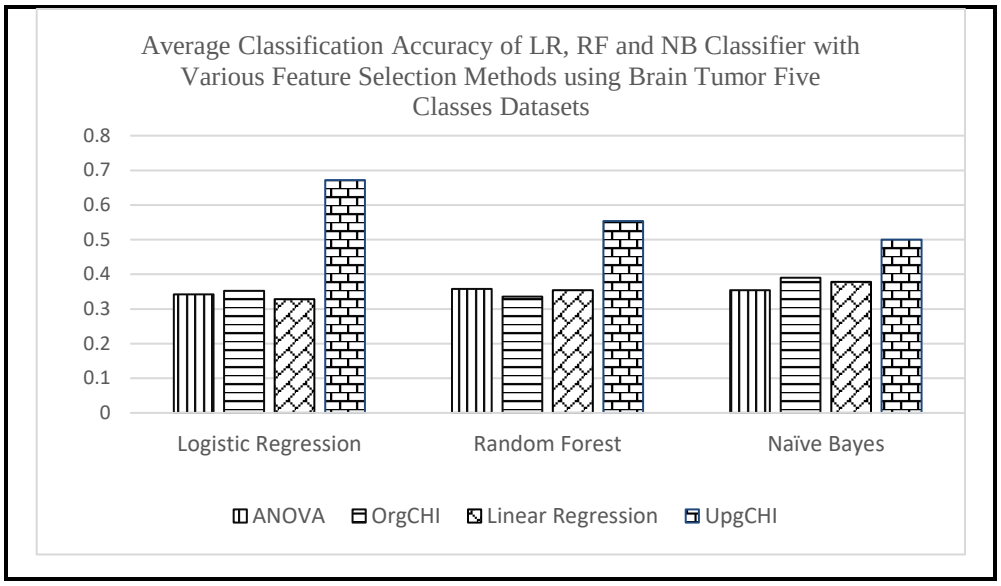

Figure 6.26 Testing Accuracy of LR, RF and NB classifier with Various Feature Selection Methods using Brain Tumor Five Classes Dataset

Figure 6.26 shows the average accuracy results of Logistic Regression, Random Forest, and Naïve Bayes classifier with various feature sets using Brain Tumor Dataset. The UpgCHI feature selection method with the Logistic Regression classifier achieved an accuracy of 67%.

Feature selection models based on statistical tests and a classifier can obtain comparative results for classification of microarray datasets, to categorize the cancer-causing genes into their respective classes. A comparative analysis is carried out to choose a feature selection model with a classifier that provides better classification accuracy. the comparative analysis of classification accuracy of the proposed Logistic Regression, Random Forest, and Naïve Bayes classifier with different feature selection models using various microarray datasets. From the obtained results, it can be inferred that among the various feature selection models used in permutation with a Logistic Regression classifier, UpgCHI provides better accuracy in the case of Leukemia four class datasets are shown in Figure 6.27.

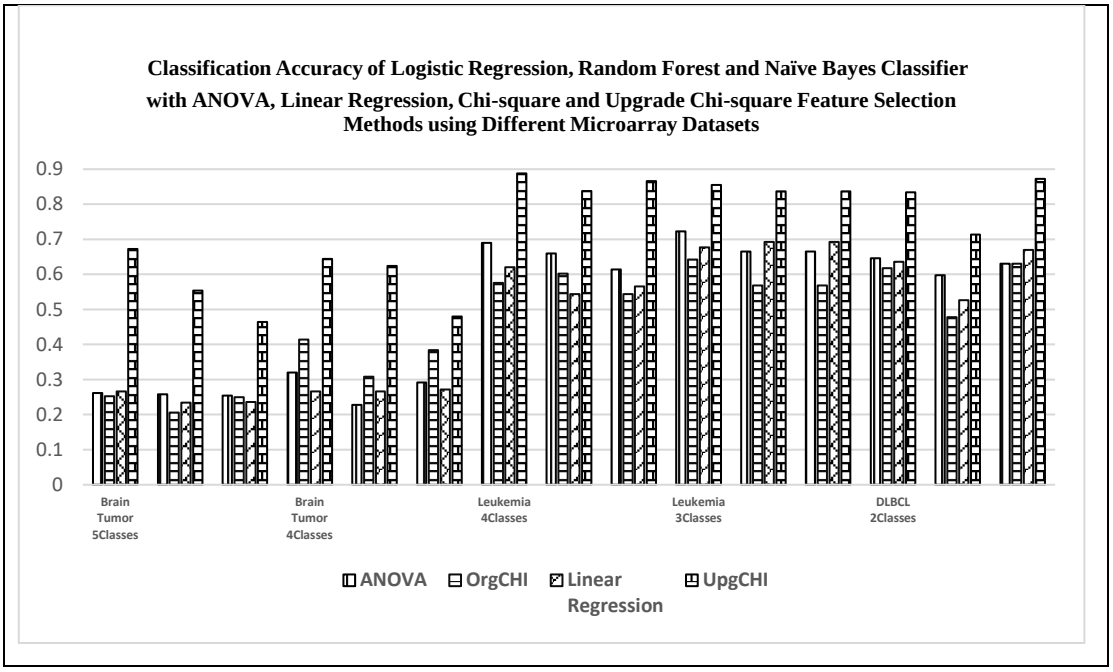

Figure 6.27 Classification accuracy of Logistic Regression, Random Forest and Naïve Bayes classifier with ANOVA, Linear Regression, Chi-square and UpgCHI Feature Selection Methods using Different Microarray Datasets

6.3 Chapter Summary

In this chapter, various types of feature selection methods based on Spark frameworks are implemented to select the relevant features from the microarray high-dimensional data. The performance of the classifier for various datasets is evaluated by varying the number of features. The results show that the UpgCHI algorithm provides better performance on scalable frameworks in terms of classification accuracy, precision, recall and F-1score.

CHAPTER 7
CONCLUSION AND FUTURE WORK

This research analyzes the multiclass microarray gene data sets in the bioinformatic areas with the two-phase procedure that includes feature selection and classification. The proposed UpgCHI algorithm has emphasized the first step of feature selection and implemented using five microarray datasets like DLBCL two classes, Leukemia three classes, Brain Tumor four classes and Brain Tumor five classes. An upgraded chi-square selects the top number of attributes belonging to each class by the ratio values of their gene's records. So, the proposed algorithm has solved the problem of class imbalance and improved the accuracy of multiclass microarray gene analysis. In the second phase, three classifiers: Logistic Regression (LR), Random Forest, and Naïve Bayes are used to classify the multi-class of microarray data on the scalable framework Spark. To evaluate the proposed system, different numbers of attributes (100, 200, 300, 400, and 500) are selected using different using different multiclass microarray datasets. The proposed system is evaluated based on the proposed UpgCHI feature selection method with three classifiers using five multiclass microarray datasets.

When the Logistic Regression classifier is implemented by using the DLBCL two classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 70%, 75%, 81%, 76%, and 80% respectively. When the Random Forest classifier is implemented by using DLBCL two classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 69%, 81%, 88%, 65%, and 80% respectively. When the Naïve Bayes classifier is implemented by using DLBCL two classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 75%, 75%, 50%, 65%, and 70% respectively. According to DLBCL two classes of datasets with Random Forest classifier, when UpgCHI is used as a feature selection method, where the top 300 features are sufficient to achieve a high accuracy of 88%.

When the Logistic Regression classifier is implemented by using the Leukemia three classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 83%, 95%, 76%, 85%, and 88% respectively. When the Random Forest classifier is implemented by using Leukemia three classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high

accuracy of 86%, 89%, 76%, 79%, and 88% respectively. When the Naïve Bayes classifier is implemented by using the Leukemia three classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 48%, 68%, 81%, 57%, and 88% respectively. According to Leukemia three classes of datasets with Logistic Regression classifier, when UpgCHI is used as a feature selection method, where the top 200 features are sufficient to achieve a high accuracy of 95%.

When the Logistic Regression classifier is implemented by using Brain Tumor four classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 70%, 46%, 68%, 74%, and 64% respectively. When the Random Forest classifier is implemented by using Brain Tumor four classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 70%, 62%, 68%, 56%, and 56% respectively. When the Naïve Bayes classifier is implemented by using the Brain Tumor four classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 65%, 31%, 56%, 48%, and 40% respectively. According to Brain Tumor four classes of datasets with Logistic Regression classifier, when UpgCHI is used as a feature selection method, where the top 400 features are sufficient to achieve a high accuracy of 74%.

When the Logistic Regression classifier is implemented by using Leukemia four classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 94%, 87%, 90%, 85%, and 88% respectively. When the Random Forest classifier is implemented by using Leukemia four classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 83%, 87%, 95%, 78%, and 76% respectively. When the Naïve Bayes classifier is implemented by using the Leukemia four classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 89%, 87%, 90%, 85%, and 71% respectively. According to Leukemia four classes datasets with Random Forest classifier, when UpgCHI is used as a feature selection method, where top 300 features are sufficient to achieve a high accuracy of 95%.

When the Logistic Regression classifier is implemented by using Brain Tumor five classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 63%, 67%, 78%, 74%, and 54% respectively. When the Random Forest classifier is implemented by using Brain Tumor five classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 46%, 71%, 48%, 56%, and 56% respectively. When the Naïve Bayes classifier is

implemented by using Brain Tumor five classes dataset, the top 100, 200, 300, 400, and 500 gene features are the best selections to achieve high accuracy of 42%, 54%, 48%, 48%, and 40% respectively. According to Brain Tumor five classes of datasets with Logistic Regression classifier, when UpgCHI is used as a feature selection method, where the top 300 features are sufficient to achieve a high accuracy of 78%.

A comparative analysis is carried out to choose a feature selection model with a classifier that provides a better classification accuracy of the UpgCHI method with an accuracy of 88% and the other methods of original chi-square with 57%, ANOVA with 69%, and Linear Regression with 62%. From the obtained results, it can be inferred that among the various feature selection models used with the Logistic Regression classifier, UpgCHI provides better accuracy in the case of the Leukemia four classes dataset.

7.1 Advantages and Limitations of the System

One of the advantages is that the microarray contains a huge amount of information. As the data size of microarray becomes huge in nature, various machine learning techniques on scalable platforms are employed to analyze this data efficiently. The processing efficiencies of these models on Spark are much greater than that on conventional systems. It is also observed that the overall accuracy (average) of these proposed methods on Spark is increased by approximately 88% than the conventional system. Therefore, the efficiency of processing the dataset using Spark is increased by approximately 88%.

However, complicated illnesses like breast cancer continue to pose the biggest risk to human survival. The expansion of statistical techniques and microarray data sets has opened up new avenues for the diagnosis, prognosis, and management of many illnesses. Microarray data analysis relies on two fundamental technologies: feature selection and classification. They are both essential for identifying genes and diagnosing illnesses. Restricted to the features of microarray data, numerous conventional approaches in this domain still require additional study to address their shortcomings.

The primary features of microarray data, along with their key problems for researchers conducting microarray data analysis, include small sample size, high complexity, and class imbalance. Because each array will yield enormous amounts of data, analyzing the findings takes a long time. The outcomes might not always be measurable and can be too difficult to understand. Reproducible outcomes are not always achieved. The cost of the technology is prohibitive. Relative concentration is indirectly measured by

the arrays. It is frequently challenging to design arrays so that homologous DNA/RNA sequences do not bind to the same probe on the array, particularly for complicated mammalian genomes. In this system, the limitation is to analyze only the two to five classes of microarray data.

7.2 Future Work

Using a distributed and scalable framework, this research analyzes microarray high-dimensional data, opening up some intriguing research avenues. It is well recognized that one of the main problems with analyzing high-dimensional data is the curse of dimensionality.

Consequently, a variety of feature selection/extraction techniques can be used to process high-dimensional big data using scalable frameworks like Spark. After selecting and extracting features, different machine learning classifiers based on deep learning, neuro-fuzzy techniques, classifier ensembles, kernel methods, decision trees, etc., can be applied to the same Spark framework and their effectiveness can be evaluated to classify the high-dimensional big data. Real-time analytics can also be added to the task via Storm or Spark streaming.

AUTHOR'S PUBLICATIONS

[p1] Lwin May Thant and Sabal Phyu, “Real Time Big Data Analytics for Feature Selection on Apache Spark”, the 12th International Conference on Future Computer and Communication (ICFCC), February 26-28, 2020

[p2] Lwin May Thant and Sabal Phyu, “Impact of Normalization Techniques in Microarray Data Analysis”, 2021 IEEE Conference on Computer Applications (ICCA).

[p3] Lwin May Thant and Tin Zar Thaw, “Feature Selection and Classification for Microarray Data using Upgrade Chi-square Test”, Indian Journal of Computer Science and Engineering, Volume 14 Issus 1, Jan-Feb, 2024.

BIBLIOGRAPHY

[1] J. Fan, F. Han, and H. Liu, “Challenges of big data analysis," National science review, vol. 1, no. 2, pp. 293–314, 2014

[2] Muhammad Umer Sarwar, Muhammad Kashif Hanif, Ramzan Talib, Awais Mobeen, and Muhammad Aslam, “A Survey of Big Data Analytics in Healthcare," International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 8, No. 6, 2017.

[3] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, “Efficient machine learning for big data: A review," Big Data Research, vol. 2, no. 3, pp. 87–93, September 2015.

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

[5] R. M. de Moraes and L. Martínez, “Computational intelligence applications for data science," Knowledge-Based Systems”, vol. 87, no. C, pp. 1–2, 2015.

[6] W. Ayadi, M. Elloumi, and J. K. Hao, “Bimine+: an efficient algorithm for discovering relevant biclusters of dna microarray data," Knowledge-Based Systems, vol. 35, pp. 224–234, November 2012.

[7] A. T. Islam, B.-S. Jeong, A. G. Bari, C.-G. Lim, and S.-H. Jeon, “Mapreduce based parallel gene selection method," Applied Intelligence, vol. 42, no. 2, pp. 147–156, 2015.

[8] S. Wang, I. Pandis, D. Johnson, I. Emam, F. Guitton, A. Oehmichen, and Y. Guo, “Optimising parallel correlation matrix calculations on gene expression data using mapreduce," BMC bioinformatics, vol. 15, no. 1, pp. 351–359, November 2014.

[9] Q. He, F. Zhuang, J. Li, and Z. Shi, “Parallel Implementation of Classification Algorithms Based on MapReduce.” Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 655–662. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16248-0_89

[10] I. Triguero, S. del Río, V. López, J. Bacardit, J. M. Benítez, and F. Herrera, “ Rosefw-rf: the winner algorithm for the ecbdl'14 big data competition: an extremely imbalanced big data bioinformatics problem," Knowledge-Based

Systems, vol. 87, pp. 69–79, June 2015.

[11] S. Li, T. Li, Z. Zhang, H. Chen, and J. Zhang, “Parallel computing of approximations in dominance-based rough sets approach," Knowledge-Based Systems, vol. 87, pp. 102–111, May 2015.

[12] J. Qian, P. Lv, X. Yue, C. Liu, and Z. Jing, “Hierarchical attribute reduction algorithms for big data using mapreduce," Knowledge-Based Systems, vol. 73, pp. 18–31, January 2015.

[13] K. E. Lee, N. Sha, E. R. Dougherty, M. Vannucci, and B. K. Mallick, “Gene selection: a Bayesian variable selection approach," Bioinformatics, vol. 19, no. 1, pp. 90–97, June 2003

[14] Y. Peng, W. Li, and Y. Liu, “A hybrid approach for biomarker discovery from microarray gene expression data for cancer classification," Cancer informatics, vol. 2, p. 301, February 2006.

[15] L. Wang, F. Chu, and W. Xie, “Accurate cancer classification using expressions of very few genes," IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), vol. 4, no. 1, pp. 40–53, Jan–Mar 2007.

[16] K. Deb and A. Raji Reddy, “Reliable classification of two-class cancer data using evolutionary algorithms," BioSystems, vol. 72, no. 1, pp. 111–129, November 2003.

[17] J. C. H. Hernandez, B. Duval, and J.-K. Hao, “A genetic embedded approach for gene selection and classification of microarray data," in Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Springer, 2007, pp. 90–101.

[18] A. Osareh and B. Shadgar, “Machine learning techniques to diagnose breast cancer," in 5th International Symposium on Health Informatics and Bioinformatics (HIBIT), Belek, Antalya, Turkey. IEEE, April 2010, pp. 114–120

[19] A. Bharathi and A. Natarajan, “Cancer classification of bioinformatics data using ANOVA," International Journal of Computer Theory and Engineering, vol. 2, no. 3, pp. 369–373, June 2010.

[20] K.-l. Tang, W.-j. Yao, T.-h. Li, Y.-x. Li, and Z.-W. Cao, “Cancer classification from the gene expression profiles by discriminant kernel-pls," Journal of Bioinformatics and Computational Biology, vol. 8, no. supp01,

pp. 147–160, December 2010.

[21] P. A. Mundra and J. C. Rajapakse, “Gene and sample selection for cancer classification with support vectors based t-statistic," Neurocomputing, vol. 73, no. 13, pp. 2353–2362, May 2010.

[22] E. Bonilla Huerta, B. Duval, and J.-K. Hao, “A hybrid lda and genetic algorithm for gene selection and classification of microarray data," Neurocomputing, vol. 73, no. 13, pp. 2375–2383, August 2010.

[23] H. Liu, L. Liu, and H. Zhang, “Ensemble gene selection for cancer classification," Pattern Recognition, vol. 43, no. 8, pp. 2763–2772, August 2010.

[24] S.-W. Zhang, D.-S. Huang, and S.-L. Wang, “A method of tumor classification based on wavelet packet transforms and neighborhood rough set," Computers in Biology and Medicine, vol. 40, no. 4, pp. 430–437, April 2010.

[25] T. Abeel, T. Helleputte, Y. Van de Peer, P. Dupont, and Y. Saeys, “Robust biomarker identification for cancer diagnosis with ensemble feature selection methods," Bioinformatics, vol. 26, no. 3, pp. 392–398, November 2010.

[26] D. A. Salem, A. Seoud, R. Ahmed, and H. A. Ali, “Mgs-cm: A multiple scoring gene selection technique for cancer classification using microarrays," International Journal of Computer Applications, vol. 36, no. 6, December 2011.

[27] C.-P. Lee and Y. Leu, “A novel hybrid feature selection method for microarray data analysis," Applied Soft Computing, vol. 11, no. 1, pp. 208–213, January 2011.

[28] D. Mishra and B. Sahu, “Feature selection for cancer classification: a signal-to-noise ratio approach," International Journal of Scientific & Engineering Research, vol. 2, no. 4, pp. 1–7, April 2011.

[29] P. Maji and S. Paul, “Rough set based maximum relevance-maximum significance criterion and gene selection from microarray data," International Journal of Approximate Reasoning, vol. 52, no. 3, pp. 408–426, 2011.

[30] X. Sun, Y. Liu, D. Wei, M. Xu, H. Chen, and J. Han, “Selection of

interdependent genes via dynamic relevance analysis for cancer diagnosis," Journal of biomedical informatics, vol. 46, no. 2, pp. 252–258, April 2012.

[31] X. Sun, Y. Liu, M. Xu, H. Chen, J. Han, and K. Wang, “Feature selection using dynamic weights for classification," Knowledge-Based Systems, January 2012.

[32] W.-C. Yeh, Y.-M. Yeh, C.-W. Chiu, and Y. Y. Chung, “A wrapper-based combined recursive orthogonal array and support vector machine for classification and feature selection," Modern Applied Science, vol. 8, no. 1, p. p11, December 2013.

[33] P. Guo, Y. Luo, G. Mai, M. Zhang, G. Wang, M. Zhao, L. Gao, F. Li, and F. Zhou, “Gene expression profile-based classification models of psoriasis," Genomics, vol. 103, no. 1, pp. 48–55, January 2013.

[34] A. Osareh and B. Shadgar, “An efficient ensemble learning method for gene microarray classification," BioMed research international, vol. 2013, July 2013.

[35] S. Hengpraprohm, “Ga-based classifier with snr weighted features for cancer microarray data classification," vol. 1, no. 1, pp. 29–33, 2013.

[36] S Prajapati, H Das, & M.K Gourisaria, “Feature selection using differential evolution for microarray data classification,” Discover Internet Things 3, 12 (2023). https://doi.org/10.1007/s43926-023-00042-5.

[37] Kim J, Yoon Y, Park HJ, Kim YH, “Comparative study of classification algorithms for various DNA microarray data,” Genes. 2022;13(3):494.

[38] Das, H., Naik, B., & Behera, H. S, “An experimental analysis of machine learning classification algorithms on biomedical data,” In Proceedings of the 2nd International Conference on Communication, Devices and Computing: ICCDC 2019 (pp. 525-539). Springer Singapore.

[39] Abdullah, M. N., Yap, B. W., Sapri, N. N. F. F., & Wan Yaacob, W. F, “Multi-class Classification for Breast Cancer with High Dimensional Microarray Data Using Machine Learning Classifier,” In Data Science and Emerging Technologies: Proceedings of DaSET 2022 (pp. 329-342). Singapore: Springer Nature Singapore.

[40] Hamla H, Ghanem K, “A Comparative Study of Filter Feature Selection Methods on Microarray Data,” In: 12th International Conference on Information Systems and Advanced Technologies “ICISAT 2022” Intelligent Information, Data Science and Decision Support System. Cham: Springer International Publishing; 2023. p. 186–201.

[41] Das H, Naik B, Behera HS, “A Jaya algorithm-based wrapper method for optimal feature selection in supervised classification,” J King Saud Uni-Comp Inform Sci. 2022;34(6):3851–63.

[42] Padhi BK, Chakravarty S, Naik B, Pattanayak RM, Das H. RHSOFS, “Feature selection using the rock hyrax swarm optimization algorithm for credit card fraud detection system,” Sensors. 2022;22(23):9321

[43] Silaich, S., & Gupta, S, “Feature Selection in High Dimensional Data: A Review,” In Third Congress on Intelligent Systems: Proceedings of CIS 2022, Volume 1 (pp. 703-717). Singapore: Springer Nature Singapore.

[44] Das, H., Naik, B., & Behera, H. S, “An experimental analysis of machine learning classification algorithms on biomedical data,” In Proceedings of the 2nd International Conference on Communication, Devices and Computing: ICCDC 2019 (pp. 525-539). Springer Singapore.

[45] Liang-jing Cai. Shu Lv, and Kai-bo Shi “Application of an Improved CHI Feature Selection Algorithm”, 13 May 2021.

[46] Nicole Dalia Cilia, Claudio De Stefano, Francesco Fontanella, Stefano Raimondo and Alessandra Scotto di Freca “An Experimental Comparison of Feature-Selection and Classification Methods for Microarray Datasets”, 10 March 2019.

[47] Kim-Anh Lê Cao, Agnès Bonnet, Sébastien Gadat “Multiclass classification and gene selection with a stochastic algorithm”, 6 Jul 2012.

[48] Mohamed Nassar, Haidar Safa, Alaa Al Mutawa, Ahmed Helal, Iskander Gaba “Chi-Squared Feature Selection over Apache Spark”, June 10–12, 2019.

[49] Y. F. Qiu, W. Wang, and D. Y. Liu, “CHI feature selection method based on variance,” Application Research of Computers, vol. 29, pp. 1304–1306, 2012.

[50] Chien-Pang Lee and Yungho Leu “A novel hybrid feature selection method for microarray data analysis”, 01 January 2011.

[51] T. White, Hadoop: The definitive guide. " O'Reilly Media, Inc.", 2012.

[52] D. Borthakur, “The hadoop distributed file system: Architecture and design," Hadoop Project Website, vol. 11, no. 2007, pp. 1–21, 2007.

[53] A. C. Murthy, V. K. Vavilapalli, D. Eadline, J. Niemiec, and J. Markham, Apache Hadoop YARN: Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2. Pearson Education, 2013.

[54] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, January 2008.

[55] M. Flores, T. Hsiao, Y. Chiu, E. Chuang, Y. Huang, and Y. Chen, “Gene regulation, modulation, and their applications in gene expression data analysis." Advances in bioinformatics, vol. 2013, pp. 360 678–360 678, January 2013.

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

[57] R. M. de Moraes and L. Martínez, “Computational intelligence applications for data science," Knowledge-Based Systems, vol. 87, no. C, pp. 1–2, 2015.

[58] Liang-jing Cai. Shu Lv, and Kai-bo Shi “Application of an Improved CHI Feature Selection Algorithm”, 13 May 2021.

[59] ND Cilia, C De Stefano, F Fontanella, S Raimondo “An Experimental Comparison of Feature-Selection and Classification Methods for Microarray Datasets”, 10 March 2019.

[60] Mohamed Nassar, Haidar Safa, Alaa Al Mutawa, Ahmed Helal, Iskander Gaba “Chi Squared Feature Selection over Apache Spark”, June 10–12, 2019.

[61] Bahassine, Said, Abdellah Madani, Mohammed Al-Sarem, and Mohamed Kissi. "Feature selection using an improved Chi-square for Arabic text classification." Journal of King Saud University-Computer and Information

Sciences 32, no. 2 (2020): 225-231.

[62] Sikri, Alisha, N. P. Singh, and Surjeet Dalal. "Chi-Square Method of Feature Selection: Impact of Pre-Processing of Data." International Journal of Intelligent Systems and Applications in Engineering 11, no. 3s (2023): 241-248.

[63] Saheed Yakub, Micheal Arowolo and Sozan Abdulsalam. “A Feature Selection Based on One Way ANOVA for Microarray Data Classification.” Al-Hikmah Journal of Pure & Applied Sciences Vol.3 (2016): 30-35

[64] Jiliang Tang, Salem Alelyani and Huan Liu. “Feature Selection for Classification: A Review”.

[65] The Apache Spark Software Foundation, “Apache Spark Documentation” https://spark.apache.org/docs/latest/ml-classification-regression.html

[66] Lwin May Thant, and Tin Zar Thaw “Feature Selection and Classification For Microarray Data Using Upgrade Chi-Square Test” Indian Journal Of Computer Science And Engineering, Volume 15 Issue 1 Jan-Feb 2024