Analysis of Microarray Data using Machine Learning Techniques on Scalable Platforms

Mukesh Kumar

Department of Computer Science and Engineering
National Institute of Technology Rourkela

Analysis of Microarray Data using Machine Learning Techniques on Scalable Platforms

Dissertation submitted in partial fulfillment

of the requirements of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Mukesh Kumar

(Roll Number: 513CS1001)

based on research carried out

under the supervision of

Prof. Santanu Kumar Rath

May, 2017

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Department of Computer Science and Engineering
National Institute of Technology Rourkela

May 23, 2017

Certificate of Examination

Roll Number: *513CS1001*
Name: *Mukesh Kumar*
Title of Dissertation: *Analysis of Microarray Data using Machine Learning Techniques on Scalable Platforms*

We the below signed, after checking the dissertation mentioned above and the official record book (s) of the student, hereby state our approval of the dissertation submitted in partial fulfillment of the requirements of the degree of *Doctor of Philosophy* in *Computer Science and Engineering* at *National Institute of Technology Rourkela*. We are satisfied with the volume, quality, correctness, and originality of the work.

Santanu Kumar Rath
(Principal Supervisor)

Suchismita Chinara
(Member, DSC)

Pabitra Mohan Khilar
(Member, DSC)

Bidyadhar Subudhi
(Member, DSC)

Swapan Bhattacharya
(External Examiner)

Durga Prasad Mohapatra
(Chairperson, DSC)

Durga Prasad Mohapatra
(Head of the Department)

Department of Computer Science and Engineering
National Institute of Technology Rourkela

Prof. Santanu Kumar Rath
Professor

May 23, 2017

Supervisor's Certificate

This is to certify that the work presented in the dissertation entitled *Analysis of Microarray Data using Machine Learning Techniques on Scalable Platforms* submitted by *Mukesh Kumar*, Roll Number 513CS1001, is a record of original research carried out by him under my supervision and guidance in partial fulfillment of the requirements of the degree of *Doctor of Philosophy* in *Computer Science and Engineering*. Neither this dissertation nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

_____
Santanu Kumar Rath

Dedication

This thesis is dedicated to my family.

For their endless love, support and encouragement.

Mukesh Kumar

Declaration of Originality

I, *Mukesh Kumar*, Roll Number *513CS1001* hereby declare that this dissertation entitled *Analysis of Microarray Data using Machine Learning Techniques on Scalable Platforms* presents my original work carried out as a doctoral student of NIT Rourkela and, to the best of my knowledge, contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of NIT Rourkela or any other institution. Any contribution made to this research by others, with whom I have worked at NIT Rourkela or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the sections ‘‘Reference’’. I have also submitted my original research records to the scrutiny committee for evaluation of my dissertation.

I am fully aware that in case of any non-compliance detected in future, the Senate of NIT Rourkela may withdraw the degree awarded to me on the basis of the present dissertation.

May 23, 2017
NIT Rourkela

Mukesh Kumar

Acknowledgment

I would like to express my gratitude to Prof. Santanu Kumar Rath, my doctoral programme supervisor for believing in my ability and allowing me to work in the challenging domain of Big data and Machine Learning. His profound insights have enriched my research work. The flexibility of work, he has offered me has deeply encouraged me producing this research work. He is always being a source of support and motivation for bringing out the quality of work. He has been very supportive more than a professor and extended parental guidance during my research work.

I am very much indebted to the Doctoral Scrutiny Committee (DSC) members Prof. S. K. Jena, Prof. B. Subudhi, Prof. P. M. Khilar, and Prof. S. Chinara for their valuable suggestions for my research work. Also, I am thankful to all the Professors and faculty members of the department for their support, advice, and encouragement during my course. I do acknowledge the academic resources that I have received from NIT Rourkela. I would like to thank the administrative and technical staff members of the Computer Science Department for support.

I would like to extend my gratitude to my wonderful collaborators outside NIT, J. Gugaliya, R. Gore, and Nandkumar LB. for all their support and guidance during my internships at ABB, Bangalore. Working with them was a good experience that I have achieved.

There are many people who influenced me and helped me to shape my mathematical skills in the first place. I am immensely indebted to Md. Azizul Haque, my math teacher in middle school and high school. His support and passion made me confident in my abilities.

Special thanks to my invaluable seniors and friends at NIT Rourkela, who made these years very enjoyable; a part of my life that I will surely miss later: Lov, Subhrakanta, Jagganath, Ashish, Abinash, Shashank, Ransingh, Nitish, Suman, Sangharatna, and Saurabh. I have learnt and earned much from your support and maturity. Thank you all, and I expect such relation in future also.

Finally, yet most importantly, I am very much grateful to my family, my elder brother Mr. Ramesh Yadav, and my mom, Mrs. Kamalpati Devi. I am very blessed to have you in my life. Thanks for believing in me, loving me unconditionally, and supporting me through all my endeavors. I lost my father, Late. R. N. Yadav, when I was 24, a trauma. Papa, it is beyond words how grateful I am for your selfless love, your dedication to our family and their comfort. You raised me, taught me priceless lessons, and gave me all I could wish for

still loved, still missed and very dear, I dedicate this dissertation to your memory and to the rest of our family.

May 23, 2017 *Mukesh Kumar*
NIT Rourkela Roll Number: 513cs1001

Abstract

Microarray-based gene expression profiling has been emerged as an efficient technique for classification, diagnosis, prognosis, and treatment of cancer disease. Frequent changes in the behavior of this disease, generate a huge volume of data. The data retrieved from microarray cover its veracities, and the changes observed as time changes (velocity). Although, it is a type of high-dimensional data which has very large number of features rather than number of samples. Therefore, the analysis of microarray high-dimensional dataset in a short period is very much essential. It often contains huge number of data, only a fraction of which comprises significantly expressed genes. The identification of the precise and interesting genes which are responsible for the cause of cancer is imperative in microarray data analysis. Most of the existing schemes employ a two phase process such as feature selection/extraction followed by classification.

Our investigation starts with the analysis of microarray data using kernel based classifiers followed by feature selection using statistical t-test. In this work, various kernel based classifiers like Extreme learning machine (ELM), Relevance vector machine (RVM), and a new proposed method called kernel fuzzy inference system (KFIS) are implemented. The proposed models are investigated using three microarray datasets like Leukemia, Breast and Ovarian cancer. Finally, the performance of these classifiers are measured and compared with Support vector machine (SVM). From the results, it is revealed that the proposed models are able to classify the datasets efficiently and the performance is comparable to the existing kernel based classifiers.

As the data size increases, to handle and process these datasets becomes very bottleneck. Hence, a distributed and a scalable cluster like Hadoop is needed for storing (HDFS) and processing (MapReduce as well as Spark) the datasets in an efficient way. The next contribution in this thesis deals with the implementation of feature selection methods, which are able to process the data in a distributed manner. Various statistical tests like ANOVA, Kruskal-Wallis, and Friedman tests are implemented using MapReduce and Spark frameworks which are executed on the top of Hadoop cluster. The performance of these scalable models are measured and compared with the conventional system. From the results, it is observed that the proposed scalable models are very efficient to process data of larger dimensions (GBs, TBs, etc.), as it is not possible to process with the traditional implementation of those algorithms.

After selecting the relevant features, the next contribution of this thesis is the scalable

implementation of the proximal support vector machine classifier, which is an efficient variant of SVM. The proposed classifier is implemented on the two scalable frameworks like MapReduce and Spark and executed on the Hadoop cluster. The obtained results are compared with the results obtained using conventional system. From the results, it is observed that the scalable cluster is well suited for the Big data. Furthermore, it is concluded that Spark is more efficient than MapReduce due to its an intelligent way of handling the datasets through Resilient distributed dataset (RDD) as well as in-memory processing and conventional system to analyze the Big datasets.

Therefore, the next contribution of the thesis is the implementation of various scalable classifiers base on Spark. In this work various classifiers like, Logistic regression (LR), Support vector machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), Artificial Neural Network (ANN), and Radial basis function network (RBFN) with two variants hybrid and gradient descent learning algorithms are proposed and implemented using Spark framework. The proposed scalable models are executed on Hadoop cluster as well as conventional system and the results are investigated. From the obtained results, it is observed that the execution of the scalable algorithms are very efficient than conventional system for processing the Big datasets.

The efficacy of the proposed scalable algorithms to handle Big datasets are investigated and compared with the conventional system (where data are not distributed, kept on standalone machine and processed in a traditional manner). The comparative analysis shows that the scalable algorithms are very efficient to process Big datasets on Hadoop cluster rather than the conventional system.

***Keywords*: *Big data*; *Microarray*; *Machine Learning*; *Hadoop*; *MapReduce*; *Spark*; *Statistical test*; *Classification*; *Feature Selection*; *Kernel Function*.**

Contents

List of Figures

List of Tables

List of Acronyms

ANN Artificial neural network, page 12
AVA All versus all, page 93
CDF Cumulative distribution function, page 35
CV Cross validation, page 19
DAG Directed acyclic graph, page 6
DNA Deoxyribonucleic acid, page 30
ELM Extreme learning machine, page 30
FIS Fuzzy inference system, page 32
fn false negative, page 61
fp false positive, page 61
HDFS Hadoop Distributed File System, page 2
HPC High performance computing, page 54
JVM Java virtual machine, page 6
KFIS Kernel Fuzzy Inference System, page 30
KNN K-Nearest neighbor, page 12
KSC Kernel subtractive clustering, page 41
LMS Least mean square, page 41
LR Logistic regression, page 12
LS-SVM Least square support vector machine, page 71
MR MapReduce, page 67
mRNA Messenger ribonucleic acid, page 1
NB Naive Bayes, page 12
OVA One versus all, page 93
PNN Probabilistic neural network, page 12
PSVM Proximal support vector machine, page 71
RBFN Radial basis function network, page 12
RDD Resilient distributed dataset, page 62
RVM Relevance vector machine, page 30
sANOVA Scalable ANOVA, page 62
SC Subtractive clustering, page 41
sf-ANN Spark based ANN, page 92
sf-KNN Spark based KNN, page 92
sf-LoR Spark based LR, page 92
sf-NB Spark based NB, page 92
sf-RBFN Spark based RBFN, page 92
sf-SVM Spark based SVM, page 92
sFriedman Scalable Friedman test, page 64
sKruskal-Wallis Scalable Kruskal-Wallis, page 63
sPSVM Scalable proximal support vector machine, page 71
SVM Support vector machine, page 12
tn true negative, page 61
tp true positive, page 61
YARN Yet Another Resource Negotiator, page 3

Chapter 1

Introduction

Microarray technology is a new and efficient approach to extract data of biomedical relevance for a wide range of applications. In cancer research, it provides high-throughput and valuable insights into differences in an individual's tumor as compared with constitutional DNA, mRNA expression, and protein expression. The advent of microarray technology has helped scientists to analyze expression levels of thousands of genes in a single experiment, as described in Appendix A [1]. One of the key components of microarray is the volume of quantitative information that it creates, which makes it as a high-dimensional data where number of features are very large with respect to samples. As a result, the major challenge in this field is handling, interpreting and utilization of this data. The high-dimensional data generally brings forth the problem of curse of dimensionality that hinders the useful information of dataset and brings noise accumulation, spurious correlations, and incidental homogeneity. It combined with large samples sizes that create various issues such as heavy computational cost and algorithmic instability [2]. To avoid this problem, feature (gene) selection/extraction methods play an essential role in high-dimensional (microarray) data analysis, which are characterized as the procedure of distinguishing and removing irrelevant features from the training data, so that the learning algorithm (classifiers) concentrates just on those aspects of the training data useful for analysis and future prediction. There are various machine learning methodologies explored in the area of bioinformatics (typically, microarray data) to analyze these datasets [3–7]. These algorithms consume a lot of time to analyze and explore large datasets on a conventional system with standard computational abilities. To counter this problem, the concept of distributed computing has been adopted, wherein the data is distributed on various systems in a cluster, and is processed using various parallel processing paradigm. Recently, Big data applications are increasingly becoming the focus of attention because of huge increase in data generation and storage. Extracting information from the Big data becomes a challenge, because current data mining techniques are not adapted to the new space and time requirements [8]. To overcome these challenges, many open source frameworks like MapReduce, Spark, MPI, etc. have been considered to develop scalable algorithms [9]. The researchers have accomplished a significant relevance of intelligent techniques to the development of data science for dealing with imprecision, uncertainty, learning, and

evolution, in posing and solving computational problems [10–18]. This thesis uses the two distributed frameworks such as MapReduce and Spark for the implementation of machine learning techniques, which are used to analyze the high-dimensional microarray datasets. These frameworks are described as follows.

1.1 Distributed Computing

Distributed computing techniques such as grid computing and cluster computing have been in use to provide better performance for different data centric applications for many years. Most of these works are based on a message-passing model, while systems that run on parallel algorithms, such as graphics processing units (GPUs), are based on shared-memory models. The efficient operation of these machines calls for an interface, that enables them to access shared file systems as well as maintains proper communication with other machines in the grid/cluster. Thus, network bandwidth becomes a bottleneck for such machines. The key intuition in distributed systems is the concept of bringing ‘‘compute to data” rather than ‘‘taking data to the compute” which happens in a traditional system. And it happens that moving ‘‘compute to data” is far more cheaper than moving ‘‘data to compute” when the size of the data exceeds a certain size. This is exactly why the buzzword Big data was coined.

1.1.1 Hadoop Framework

Hadoop is an open-source software framework that provides storage and processing of large datasets in a distributed fashion across clusters of commodity hardwares [19]. It has been designed to store data across all nodes (servers) in a cluster in a distributed manner by dividing files into smaller entities called blocks. This enables each node to work according to the principle of proximity, i.e., process the data, which is available locally, leading to fewer network transmissions. In addition to this, it provides a low-cost and scalable architecture designed to scale up from a single server to thousands of servers. Its design enables detection and handling of failures at the application layer, thus providing service even in error-prone systems. There are three core components of the Hadoop framework included in the package, which is managed and maintained by the Apache Software Foundation. They are as follows:

1.1.1.1 Hadoop Distributed File System (HDFS)

This is a Java-based distributed file system that stores different types of data without prior organization, unlike relational databases. Though similar to existing distributed systems, there are significant differences. It can be deployed on low-cost hardware commodities, and its architecture allows for speedy recovery from failure. It consists of a name node and several data nodes. The name node maintains the directory tree of all files in the file system and tracks where, across the cluster, files are stored. It does not store these files itself. In addition, the files stored in a data node are replicated on other data nodes; thereby allowing

speedy recovery whenever a node fails. When a file is written, the name node updates the directory tree and then sends the file to a data node. The data node further sends this file to other data nodes for replication [20]. Figure 1.1 shows the architecture of HDFS.

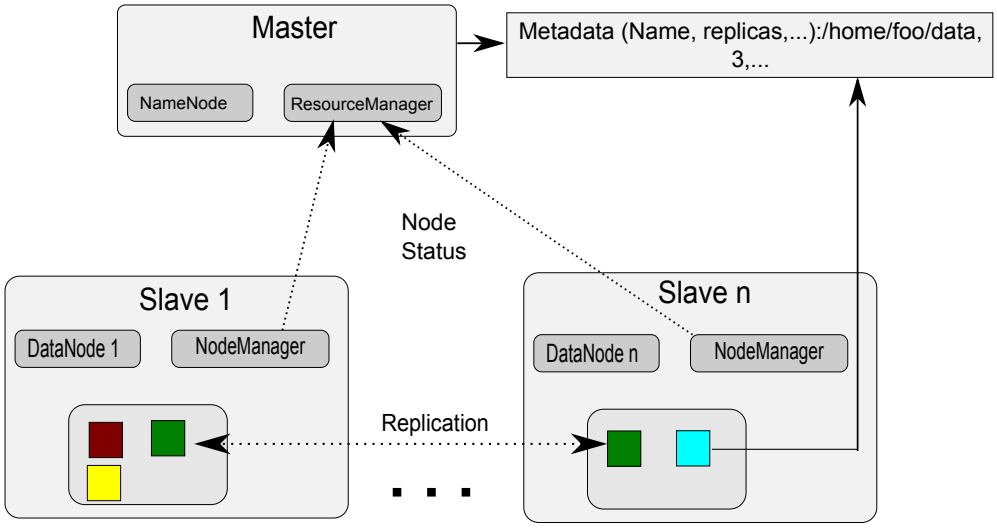


Figure 1.1: HDFS Architecture

1.1.1.2 Yet Another Resource Negotiator (YARN)

This is a resource management framework for scheduling and handling resource requests from distributed applications. YARN combines a central resource manager that manages the way applications use Hadoop system resources with node manager agents that monitor processing operations in individual cluster nodes. In case a node fails, it reschedules the job to some other node [21].

1.1.1.3 MapReduce

This is a programming model, developed by Google, for processing large datasets in a distributed manner on clusters of commodity hardware [22]. It consists of 3 steps:

- **Map step:** Each node applies the ‘‘map( )” function to local data, and writes the output to temporary storage with a key value associated with the output.
- **Shuffle step:** the nodes redistribute the data based on the output keys (produced by the map function) in such a manner that all the data belonging to one key are located on the same node.
- **Reduce step:** Each node processes each group of output data, per key, in parallel, and the output is then written to the HDFS.

Figure 1.2 shows the architecture of MapReduce and its life cycle, and it is taken from `http://hadooper.blogspot.in/`.

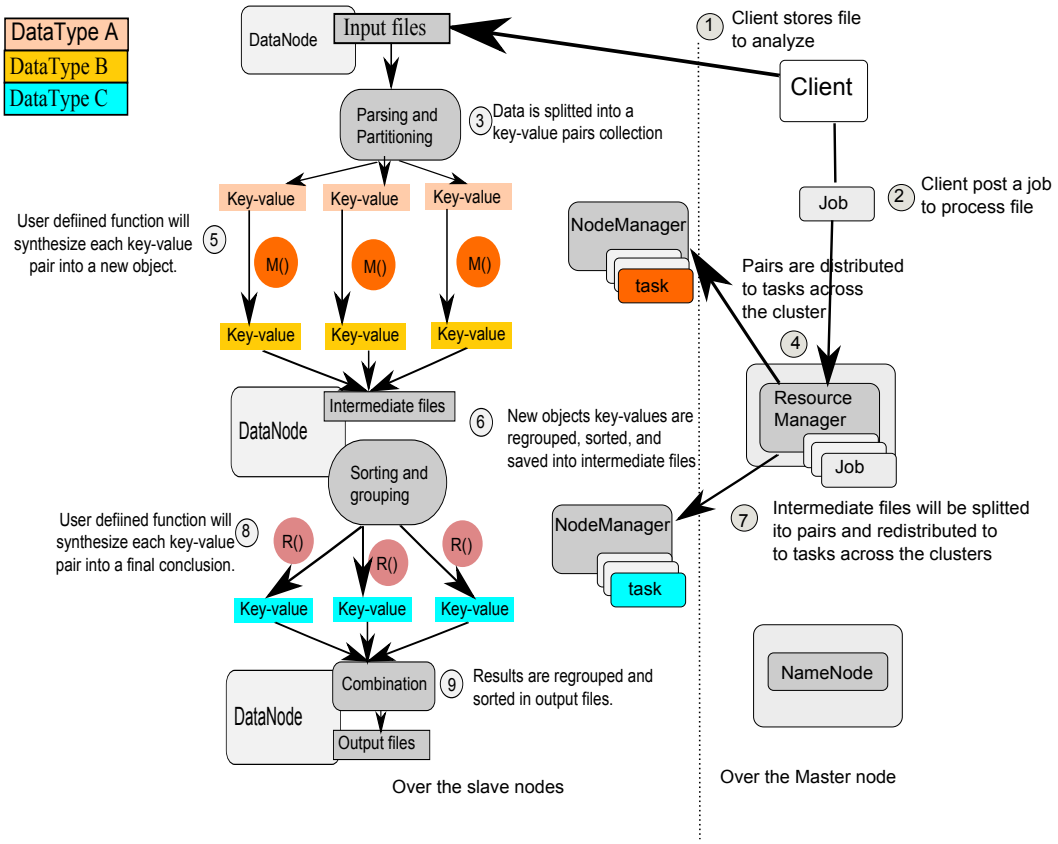


Figure 1.2: Life cycle of MapReduce

1.1.2 Execution on Hadoop using MapReduce

To run algorithms in the Hadoop framework, they are divided into two phases, i.e., map and reduce. In the map phase, mappers read a line from the file (stored as blocks in the node). The mappers process the input and calculate the required output (e.g., F-values in ANOVA or distances in sf-KNN, (section 3.3)). These values, along with their corresponding keys ('Feature ID' in the case of a feature set), are written to an intermediate file. These are then sorted, shuffled and sent to the reducers. The reducers process the input and then write the obtained results into the HDFS. The working principle of MapReduce is explained in Figure 1.2.

For instance, if the input file is 550 MB in size and the HDFS has a block size of 128 MB, when the dataset (input file) is uploaded into HDFS, it is divided into five ($\lceil 550/128 \rceil = 5$) blocks. The first four blocks will be of size 128 MB each, and the last block will retain the remaining size of the file, 38 MB. Further assuming that the split size is equal to the block size, i.e., corresponding to five splits, five mappers are formed. Mappers are executed on all data nodes simultaneously and write their output to an intermediary file in the HDFS. The reducers perform further operations on the obtained results from the mappers, and write the final result into the HDFS.

1.2 Spark Architecture and Resilient Distributed Dataset

Apache Spark is a cluster computing framework for large scale data processing which aims to be fast and general purpose engine. It is the extension of the most popular data flow model MapReduce, and developed so that interactive and iterative computation can be efficiently supported which are very hard to implement with Hadoop MapReduce. Spark was specifically designed for use cases that reuse a working set of data across parallel operations.

1.2.1 Spark architecture

Spark is an open source processing engine, which uses directed acyclic graph and its own data structure i.e., Resilient Distributed Dataset (RDD) to provide speed and analytics [23]. Spark runs on top of existing Hadoop infrastructure to provide enhanced and additional functionality. Spark architecture consists of one driver program and multiple worker nodes. The driver system is responsible for creation of RDD and Spark Context object, while the worker nodes are involved in parallel transformation of RDDs. Transformation and Action are the two operations supported by RDD. Spark provides the option of using other cluster managers like YARN and MESOS along with its own standalone cluster manager. Figure 1.3 shows Spark architecture, which includes Cluster manager, Driver node and Worker nodes . The components of Spark are described as follows:

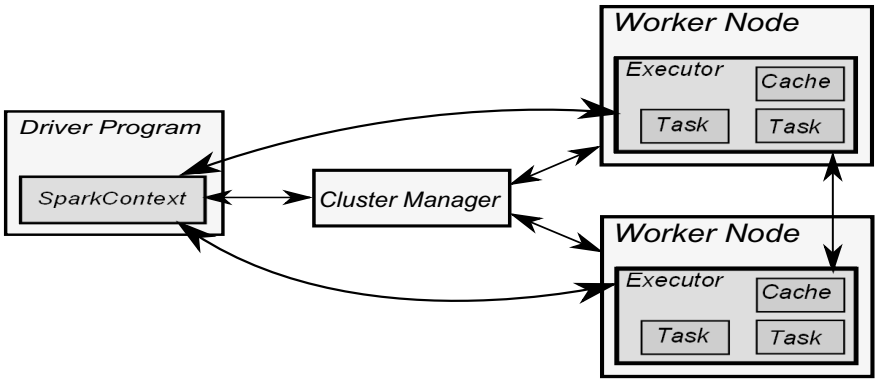


Figure 1.3: Spark architecture in cluster mode

- Spark driver: It is a client side application that creates the Spark Context.
- Spark Context: It talks to Spark driver and cluster manager to launch the Spark executers.
- Cluster manager: Cluster managers are like YARN, MESOS, etc.
- Executers: Spark worker bees.

Figure 1.4 represents the cycle of RDD *Transformation* and *Action*.

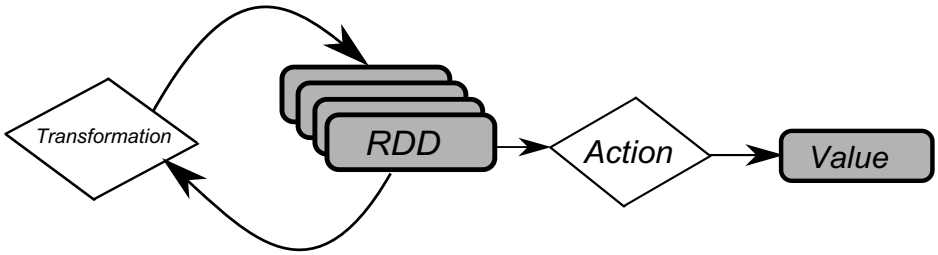


Figure 1.4: RDD Transformations and Actions

1.2.2 Execution on Spark

All jobs in Spark constitute a chain of operators and manipulate a set of data. These operators are used to fabricate a Directed Acyclic Graph (DAG). Optimization of this DAG is done by rearrangement and combination of operators as and when required, and if possible. Spark has a small code base and the system is divided in various layers (independent of each other). Each layer has some obligations.

1. The first layer is the interpreter, Spark uses a modified Scala interpreter.
2. When the user enters the code in Spark console (thus leading to creation of RDD's and application of operators), Spark constructs an operator graph.
3. When the user runs an action (like collect), the DAG Scheduler is invoked and the Graph is submitted to it. This scheduler splits the operator graph into map and reduce stages.
4. A stage consists of tasks based on input data's partitions. Pipelining of operators is done by the DAG scheduler in order to optimize the graph. For example many map operators can be docketed in one stage. This optimization is key to Spark's performance. We get a set of stages as the final result of a DAG scheduler.
5. The stages, which are classified as Map or Reduce (discussed in Hadoop) are then passed on to the Task Scheduler. The task scheduler launches tasks via cluster manager (Spark Standalone/YARN/MESOS). The task scheduler isn't aware about dependencies among stages.
6. The Worker executes the tasks on the Slave. A new java virtual machine (JVM) is started per JOB. The worker has knowledge only about the code that gets passed to it.

1.3 Time Complexity Analysis

In this section, the time taken for execution of an algorithm processed with either 'MapReduce' or 'Spark' on a Hadoop cluster and a conventional system is discussed.

As discussed above, the time taken for execution on a Hadoop cluster is equivalent to the sum of the maximum time consumed by each mapper (t_m), each reducer (t_r), and the time taken for communication between the data nodes as well as name node (t_c), i.e.,

$$T_{Hadoop} = t_m + t_r + t_c \tag{1.1}$$

Let μ be the total number of data entries (or feature sets) in a file, which are distributed among n data nodes so that the total number of entries per mapper is μ/n. Let the time taken to execute one entry be (tf_m), so that

$$t_m = (\mu/n) * tf_m \tag{1.2}$$

In the reducer, let the time taken per entry be (tf_r). Therefore, the total time taken by a reducer would be (assuming r reducers)

$$t_r = (\mu/r) * tf_r \tag{1.3}$$

The data node communicates with the name node only after executing the whole batch of entries assigned to it. Assuming that it takes t_{cd} for communication between the name node and the data node, the total time taken for communication is computed using the following equation:

$$\begin{aligned} t_c = &\quad n * tc_d + r * tc_d \\ = &\quad (n + r) * tc_d \end{aligned} \tag{1.4}$$

Replacing (1.1) by (1.2), (1.3), and (1.4), the time elapsed on the Hadoop cluster (T_{Hadoop}) is

$$\begin{aligned} T_{Hadoop} = &\quad ((\mu/n) * tf_m) + ((\mu/r) * tf_r) + (n + r) * tc_d \\ = &\quad (\mu/n) * (tf_m/n + tf_r/r) + (n + r) * tc_d \end{aligned} \tag{1.5}$$

In a conventional system ($conv$), all μ entries are stored as a single entity on one system. Hence, the time taken for complete execution on a conventional system, i.e., T_{conv} is

$$T_{conv} = t_{m1} + t_{r1} \tag{1.6}$$

where,

$$t_{m1} = \mu * (tf_m) \tag{1.7}$$

$$t_{r1} = \mu * (tf_r) \tag{1.8}$$

Thus,

$$\begin{aligned} T_{conv} = & \quad t_{m1} + t_{r1} \\ = & \quad \mu * (tf_m + tf_r) \end{aligned} \tag{1.9}$$

For the sake of analysis, it is assumed that the time for communication, which depends on network quality, is constant. In the case of small datasets, there will be a limited number of blocks; as a result, they will be distributed over fewer systems, i.e., the value of n and r will be small (n, r=1 for sizes less than one block size). Thus, Equations 1.5 and 1.9 show that, in the case of small datasets, the time consumed by a Hadoop cluster is greater than that of a conventional system. In the case of large datasets, the data are divided into more blocks. Thus, in the case of large datasets, the time taken by a Hadoop cluster will always be less than that of a conventional system. To confirm the above inferences, various microarray datasets of different dimensions are considered for classification, which is discussed in the subsequent chapters.

1.4 Research Motivation

Big data tend to have high-dimensionality and massive sample size. These two properties raise three unique disputes: (i) High dimensionality leads to noise aggregation, illegitimate correlations, and non-essential homogeneity; (ii) High dimensionality united with massive sample size creates issues such as hefty computational cost and algorithmic imbalance; (iii) The humongous samples in Big data are typically collected from multiple sources at varied time points using dissimilar technologies and then aggregated. This leads to heterogeneity issues, experimental fluctuations, and statistical biases, due to which we need to formulate adaptive and vigorous procedures.

The microarray is a type of high-dimensional Big data, which has gigantic number of features with respect to the number of samples. It would help physicians for early diagnosis and treatment of various diseases. The constraints of microarray analysis are:

- It contains huge amount of data, only a fraction of which comprises of significantly expressed genes.
- Curse of dimensionality that obstructs the profitable information of dataset and leads to computational imbalance.
- It is a type of semi-structured data with a high volume. To store and process this type of data in a finite time is very tedious task.
- The identification of the precise and interesting genes which are responsible for the cause of cancer is essential.

1.5 Research Objectives

The two main aims of high-dimensional data analysis are to formulate efficient methods that can correctly anticipate the future observations and simultaneously perceive the relationship between the features. Furthermore, due to massive sample size, Big data gives rise to two additional goals, i.e., to empathize heterogeneousness and commonality across different sub-populations. Particularly, Big data gives assurances for: (i) Exploring the invisible structures of every sub-population of the data, which is traditionally impractical (sometimes, for small samples they are considered as 'outliers'); (ii) Eliciting significant common properties across various sub-populations despite extensive individual variations.

The objectives of this work are to:

- Store and process the high-dimensional (microarray) data (categorized as Big data due to its volume, velocity, and variety) using Hadoop (HDFS, MapReduce, and Spark) platform.
- Extract optimal number of significantly expressed features (genes) using feature selection method based on MapReduce and Spark frameworks.
- Develop an efficient classifier based on MapReduce and Spark, in order to classify the dataset in a short time.
- Improve the time complexity of the models (i.e., feature selector as well as classifier).

1.6 Thesis Contribution

The contribution of the thesis is explained as follows:

Chapter 2 describes a brief review on the literature available for the analysis of microarray data using machine learning techniques. The survey includes various criteria chosen by the authors such as the feature selection and classification methods employed. Subsequently, the implementation of the existing feature selection methods and classifiers, which are most frequently applied to classify the microarray datasets are implemented and the results are compared using three datasets viz., Leukemia, Breast, and Ovarian cancer.

Chapter 3 proposes various kernel based classifiers like Extreme Learning Machine (ELM), Relevance Vector Machine (RVM), and Kernel Fuzzy Inference System (KFIS) with linear, polynomial, RBF, and tansig kernel functions and their performances are compared with the Support Vector Machine (SVM). As the size of datasets increases the traditional machine learning techniques are not suitable enough to process efficiently within a definite time. To mitigate these issues, distributed and scalable platform like Hadoop is considered for storage (HDFS) and processing (MapReduce and Spark) of data in a distributed manner. To validate the efficacy of the distributed and scalable systems, the

microarray high-dimensional datasets with various sizes have been considered. Hence, the existing methodologies as mentioned in the literature have been implemented on scalable platforms to analyze the microarray datasets.

Chapter 4 deals with the implementation of scalable feature selection methods like ANOVA, Kruskal-Wallis, and Friedman tests. The proposed methods are implemented with MapReduce and Spark on Hadoop cluster and their performance is measured. After selecting the relevant and significant features, various scalable classifiers are proposed in the subsequent chapters to classify the high-dimensional multi-class datasets.

In **Chapter 5** a scalable proximal support vector machine classifier has been proposed, and implemented using MapReduce and Spark frameworks. The proposed classifier is executed on Hadoop as well as conventional system and the performance is compared. From this chapter, it is concluded that Spark is more efficient than MapReduce and conventional system to analyze the datasets.

Chapter 6 proposes scalable implementation of various classifiers such as Logistic regression (LR), Support vector machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), Artificial Neural Network (ANN), Radial basis function network (RBFN) (with hybrid learning and gradient descent learning) to classify the microarray high-dimensional data. The proposed models are implemented using Spark framework on the top of Hadoop cluster, and their efficiencies are measured with the conventional system and the results are compared. The processing efficiencies of these models on Spark are much greater than that on conventional system. From the obtained results, It is concluded that to process the high-dimensional data with big sizes (GBs, TBs, etc.), the distributed and scalable cluster like Hadoop (Spark) is better choice for the researchers.

1.7 Thesis Organization

This thesis starts with an introduction to microarray high-dimensional data including the research challenges and objectives. It is organized into seven chapters where each chapter portrays the contributions specific to a domain. The layout of this thesis is given below.

Chapter 2: Literature review

The existing literature is explored covering two major domains of microarray data classification: (a) feature selection, and (b) classification. A tabular comparison of various approaches are presented along with their corresponding author. After study of literature, the most frequently used methods are implemented on the same framework with similar steps of analysis and the results are analyzed.

Chapter 3: Classification of Microarray Data using Kernel based Classifiers

This chapter deals with study of kernel based classifiers. The microarray data are preprocessed and then t-test is applied to select the relevant features. After feature selection,

various kernel based classifiers like ELM, RVM, and KFIS are proposed. The performance of the same is then evaluated and compared with SVM.

Chapter 4: Feature selection of Microarray Data using Scalable Statistical tests

In this chapter, various statistical tests like ANOVA, Kruskal-Wallis and Friedman tests are implemented on scalable frameworks like MapReduce and Spark on top of Hadoop cluster as a feature selection methods. The proposed methods are applied on various microarray data and select the relevant features from them. Finally, the performance of these methods are investigated on Hadoop cluster and compared with conventional system.

Chapter 5: Classification of Microarray Data using Scalable Proximal Support Vector Machine Classifier

In this chapter, Scalable Proximal Support Vector Machine Classifier (sPSVM) is proposed on various scalable frameworks like MapReduce and Spark and applied on microarray datasets of various dimensions. The performance of classifier is investigated with various FS methods, and comparison is made on different platforms like Hadoop cluster (MapReduce and Spark) and conventional system.

Chapter 6: Classification of Microarray Data using Various Scalable Classifiers on Spark

This chapter presents an experimental investigation of various scalable classifiers such as Logistic regression (LR), Support vector machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), Artificial Neural Network (ANN), Radial basis function network (RBFN) with hybrid learning and gradient descent learning based on Spark framework. Various high-dimensional microarray data are applied to investigate the performance of the proposed classifiers followed by feature selection (discussed in Chapter 4). The performance of proposed classifiers are then compared on various frameworks like Spark cluster and convention system.

Chapter 7: Conclusions and Future Work

This chapter presents the conclusions derived from the proposed methodologies with more emphasis on achievements and limitations. The scopes for future research are highlighted at the end.

Chapter 2

Literature Review

This chapter emphasizes on the state-of-the-art techniques involved in pattern classification system such as the feature selection/extraction, dimensionality reduction, and the design of a classifier. The review has been performed in two broad aspects of data analytics with respect to objectives of the thesis.

This chapter highlights on the research work carried out by various authors on classification of microarray datasets. The survey includes various criteria chosen by the authors such as the feature selection and classification methods employed, and the dataset used. The results on the survey work done for microarray data classification conclude that a good number of researchers and practitioners have employed statistical tests as a feature selection and the various machine learning techniques to classify the dataset.

The later part of the chapter highlights on the implementation of the existing feature selection methods and classifiers, which are most frequently applied to classify the microarray datasets.

In this chapter, different feature selection (FS) methods like T-test, F-test, Wilcoxon test, SNR, χ^2-test, Information Gain, Gini Index, and Fisher Score are used to select the genes having high relevance. The top ranked genes are used to classify the microarray data using various classifiers like Logistic regression (LR), Naive Bayes (NB), K-Nearest neighbor (KNN), Artificial neural network (ANN), Radial basis function network (RBFN), Probabilistic neural network (PNN), and Support vector machine (SVM), and the results are analyzed.

The rest of the chapter is organized as follows: Section 2.1 highlights on the related work in the field of microarray data classification. Section 2.2 presents the step wise procedure for classifying the microarray data using various classifiers. It highlights the empirical analysis, the results obtained and interpretation drawn from the existing classifiers. This section also presents a comparative analysis for gene classification of microarray data with classifiers available in literature. Section 2.3 summarizes the chapter and considers the scope for future work.

2.1 Literature review

In this section, a brief review on the work done by various authors related to microarray data is provided, and is tabulated as shown in Table 2.1. The table is subdivided into three categories such as: name of the author, the feature selection technique used by the respective author for selecting the most significant features of a sample which are responsible for causing cancer, and the last column represents the various machine learning classifiers used for classification of microarray data.

Table 2.1: Related work

Author (Year)	**Feature selection/optimization techniques applied**	**Classifier applied**
Osareh et al. (2010)	Signal-to Noise Ratio (SNR)	Support vector machine (SVM), K-nearest neighbor (KNN) and Probabilistic neural network (PNN) [24]
Dina et al. (2011)	Multiple scoring gene selection technique (MGS-CM)	SVM, KNN, Linear discriminant analysis (LDA) [25]
Wang et al. (2007)	t-test	Fuzzy Neural Network (FNN), SVM [5]
Zhang and Dend (2007)	Based Bayes error Filter (BBF)	SVM, KNN [26]
Xiyi Hang (2008)	ANOVA	Sparse Representation-SVM [27]
Bharathi and Natarajan (2010)	ANOVA	SVM [28]
Tang et al. (2010)	ANOVA	Discriminant kernel partial least square (PLS) [29]
Furey et al. (2000)	Signal to Noise Ratio	SVM [30]
Li et al. (2001)	Genetic algorithm	KNN [31]
Ben-Dor et al. (2000)	All genes, TNoM score	Nearest neighbor, SVM with quadratic kernel, and AdaBoost [32]
Nguyen et al. (2002)	Principal component analysis (PCA)	Logistic discriminant, and Quadratic discriminant [33]
Guyon et al. (2002)	RFE	SVM [34]

Mundra et al. (2010)	T-statistic, SVM based t-statistic, SVM with RFE SVM based t-statistic with RFE	SVM [35]
Lee et al. (2011)	$\chi^2 - test$	Hybrid with GA + KNN and SVM [36]
Cho et al. (2004)	SVM-RFE	Kernel KFDA [37]
Deb and Reddy (2003)		NSGA-II [6]
Lee et al. (2003)	Bayesian model	ANN, KNN, SVM [3]
Lee et al. (2003)		Multicategory SVM [38]
Paul and Iba (2004)	Probabilistic Model Building Genetic Algorithm (PMBGA)	Naive-Bayes, weighted voting classifier [39]
Huerta et al. (2006)	GA	GA/SVM [40]
Ye et al. (2004)	Uncorrelated Linear Discriminant Analysis (ULDA)	KNN [41]
Liu et al. (2005)	GA	SVM [42]
Alba et al. (2007)	GA, PSO	SVM [43]
Yu et al. (2004)	Redundancy Based Filter (RBF)	C4.5 [44]
Ding et al. (2005)	Minimum Redundancy- Maximum Relevance (MRMR) feature selection	NB, LDA, SVM, LR [45]
Cho and Won (2007)	Representative vector	Ensemble NN [46]
Yang et al. (2006)	PCA	LDA [47]
Peng et al. (2006)	Fisher ratio	NB, Decision tree J4.8, SVM [4]
Wang et al. (2006)	Information Gain (IG)	Neuro-fuzzy ensemble [48]
Pang et al. (2007)	Bootstrapping consistency gene selection	KNN [49]
Li et al. (2007)	t-test + Partial least square (PLSDR)	KNN, SVM [50]
Yue et al. (2007)		LDA [51]
Hernandez et al. (2007)	GA	SVM [7]
Li et al. (2008)	Wilcoxon test,	GA-SVM [52]
Huerta (2010)	t-test	LDA-GA [53]

Wang et al. (2005)	Correlation-based feature selection	C4.5 (J4.8), NB, SMO-CFS, SMO-Wrapper, Emerging pattern, SVM, Voting Machine, MAMA [54]
Ruiz et al. (2006)	Best incremental ranked subset (BIRS)-BIRSw, BIRSf	NB, IB, C4.5 [55]
Zhu et al. (2004)	Univariate ranking (UR) and Recursive feature elimination (RFE)	SVM, PLR [56]
Huynh et al. (2009)		SVD-Neural Network [57]
Diaz et al. (2006)	Random forest	Random forest [58]
Yeh et al. (2013)	Orthogonal array (OA)-SVM	SVM [59]
Yu et al. (2008)	SNR	PSO-SVM [60]
Chen et al. (2008)	(Fisher-ratio, Pearson Correlation, Cosine coefficient, Euclidean Distance and Spearman Correlation	Ensemble with different classifier [61]
Liu et al. (2010)	Ensemble gene selection	kNN [62]
Mishra et al. (2011)	Kmeans, SNR	SVM, KNN (Kmeans + SNR + SVM/KNN) [63]
Huerta et al. (2008)	Fuzzy Logic with BSS/WSS, t-test, Wilcoxon test	KNN [64]
Liu et al. (2005)	Entropy	Greedy, Simulated Annealing (SA) [65]
Liu et al. (2008)	(Mutual Information (MI), t-test, SNR) ICA, PCA, RP	Rotation forest [66]
Cho et al. (2004)	Derivative of kernel function	kernel Fisher discriminant analysis (KFDA) [37]
Sharma et al. (2008)	Gradient LDA	KNN [67]
Sun et al. (2012)	Dynamic relevance gene selection (DRGS)	SVM, KNN, Predictive analysis of microarray (PAM) [68]
Shim et al. (2009)	Wilcoxon test	Supervised Weighted Kernel Clustering (SWKC) SVM [69]
Hong et al. (2008)	Pearson correlation	One vs Rest SVM (OVR SVM with NB) [70]

Chen et al. (2007)	Multiple kernel support vector machine (MK-SVM-i)	Multiple kernel support vector machine (MK-SVM-ii) [71]
M. Akay (2009)	F-score	SVM [72]
Shen et al. (2009)	Suitability score	Suitability score [73]
Yu et al. (2009)	Ant colony optimization (ACO), Marker Gene selection using ACO (MMACO), SNR	SVM [74]
Sun et al. (2012)	Dynamic weighted FS (DWFS)	KNN, NB [75]
Maji et al. (2011)	Maximum relevance-max significance	SVM, KNN [76]
Zhang et al. (2010)	Wavelet packet transforms and neighborhood rough set (WPT + NRS)	SVM [77]
Abeel et al. (2010)	Ensemble SVM-RFE	SVM [78]
Berrar et al. (2002)		PNN [79]
P. Guo et al. (2013)	RFE, and SVM-RFE	ANN [80]
González et al. (2013)	Multivariate joint entropy guided by simulated annealing	SVM [81]
González- Navarro (2011)	Mutual Information (MI) using bootstrap	SVM, NB, KNN, LR [82]
Cai et al. (2009)	Mutual Information (MI)	SVM [83]
Wang et al. (2008)	Hybrid huberized support vector machine (HHSVM)	SVM [84]
Bu et al. (2007)	PCA, genetic algorithm, and the floating backward search method	support vector machine [85]
Hong et al. (2008)	gene boosting	SVM [86]
Hewett et al. (2008)	Boosting-based ranking algorithm (MDR)	SVM [87]
Yu et al. (2013)	Enseble technique	SVM [88]
Osareh et al. (2013)	ReliefF, Correlation-based filter selection (CFS), Minimum Redundancy Maximum Relevance (mRMR), and General Signal to Noise Ratio (GSNR), Fast Correlation-based Filter (FCBF)	Decision tree, Root based ensemble [89]

Hengpraprohm (2013)	SNR	GA-based classier [90]
Sejja et al. (2011)	SNR	SVM [91]
Kim et al. (2004)	Information Gain	GA-ANN [92]

Following are the observations made from the literature survey carried out:

- Numerous feature selection techniques have been employed, as it plays a crucial role in selecting most **significant features** before performing classification of the dataset.
- Majority of the authors have used SVM classifier for performing microarray data classification. However is was observed that hybrid approaches have been more frequently used.
- Out of numerous datasets available in literature survey, many authors have used dataset related to Leukemia, Ovarian and Breast cancer [93].

In this chapter, eight feature selection techniques [94] [T-test[95], F-test [96], Wilcoxon test [52, 97], SNR [24], χ^2-test [98], Information Gain [99], Gini Index [100], Fisher Score [101]] in combination with seven machine learning techniques [Logistic regression (LR) [102], Naive Bayes (NB) [102], K-Nearest Neighbor (KNN) [103], Artificial neural network (ANN) [104], Radial basis function network (RBFN) [105], Probabilistic neural network (PNN) [106], and Support vector machine (SVM) [107]] are used for carrying out the classification of samples into cancerous or non-cancerous classes for Leukemia, Ovarian, and Breast cancer datasets.

2.2 Empirical Analysis of Existing Techniques

This section presents the proposed approach for classification of microarray data, which consists of various phases:

1: Emphasizes on preprocessing the input data using various methods such as missing data imputation, and normalization.

2: Feature selection is carried out using various methods like T-test, F-test, Wilcoxon test, SNR, χ^2-test, Information Gain, Gini Index, and Fisher Score.

3: Classification is performed using different classifiers like Logistic regression (LR), Naive Bayes (NB), K-Nearest neighbor (KNN), Artificial neural network (ANN), Radial basis function network (RBFN), Probabilistic neural network (PNN), and Support vector machine (SVM) already available in the literature.

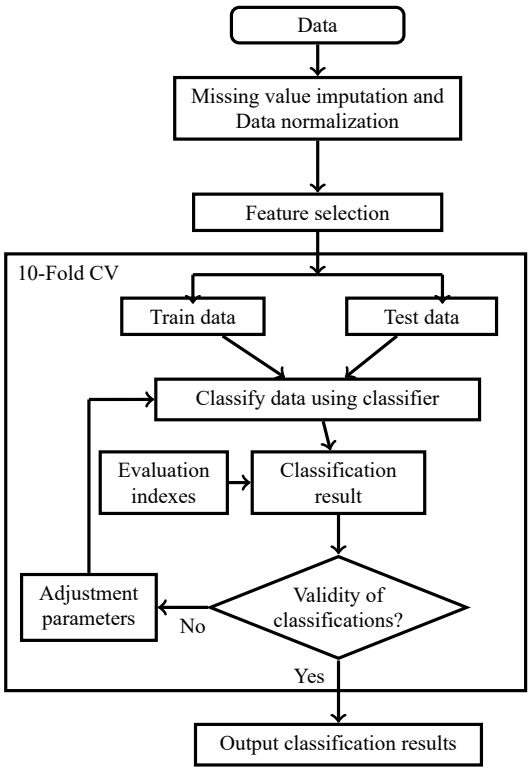


Figure 2.1: Step wise procedure for microarray classification.

The graphical representation of the proposed approach is depicted in Fig. 3.1.
The following steps give a brief description of the proposed approach.

1. **Data collection**

   The dataset for classification analysis which acts as requisite input to the models is obtained from Kent Ridge Bio-medical Dataset Repository [93].

2. **Missing data imputation and normalization of dataset**

   Missing data of a feature (gene) of microarray data are imputed by using the *mean* value of the respective feature. Input feature values are normalized over the range $[0, 1]$ using Min-Max normalization technique [108]. Let X_i be the i_{th} feature of the dataset X, and x is an element of the X_i. The normalization of the x can be calculated as:

   $$Normalized(x) = \frac{x - \min(X_i)}{\max(X_i) - \min(X_i)} \tag{2.1}$$

   where, $\min(X_i)$ and $\max(X_i)$ are the minimum and maximum values for the dataset X_i respectively. If $\max(X_i)$ is equal to $\min(X_i)$, then $Normalized(x)$ is set to 0.5.

3. **Division of Dataset**

   The dataset is divided into two categories such as: training set and testing set.

4. **Feature selection of dataset**

Different statistical methods have been applied to select the features having high relevance value and hence the curse of dimensionality issue has been reduced.

5. **Build classifier**

   Different classifiers have been built to classify the microarray dataset.

6. **Test the model**

   Model is tested using the testing dataset and then the performance of the classifier has been compared with various performance measuring criterion using cross validation (CV) technique.

2.2.1 Results and Interpretation

In this section, the obtained results are discussed for the proposed approach (Section 2.2). Three case studies viz., Leukemia [109], Breast cancer [110], and Ovarian cancer [111] microarray datasets are considered to find the classification accuracy. The performance of the classifier is assessed using 10 fold cross validation (CV) technique for the various datasets. The cross validation technique provides more realistic assessment of classifiers, which generalizes significantly to unseen data.

Since the dataset contains a very huge number of features with irrelevant information, feature selection (FS) method are applied to remove irrelevant information. Application of feature selection methods aim at finding a feature subset that has the most discriminative information from the original feature set. This helps in selecting the features (genes) which have high relevance score. The genes with low relevance score are stripped off. The various statistic methods have been used to choose genes with high relevance score.

To achieve the objectives of FS such as (a) to avoid over-fitting and improve model (classifier) performance, (b) to provide faster and more cost effective models, and (c) to gain a deeper insight into the underlying processes that generate the data, forward selection (incremental approach) method has been employed by selecting the features having high discriminative information (power) using various FS methods. Then, the top hundred features are selected in the multiples of two, i.e., 2, 4, ...,100.

After performing feature selection using above mentioned FS methods, the proposed classifiers like Logistic regression (LR), Naive Bayes (NB), K-Nearest neighbor (KNN), Artificial neural network (ANN), Radial basis function network (RBFN), Probabilistic neural network (PNN), and Support vector machine (SVM) have been applied to classify the reduced dataset.

When the samples are sequentially selected, the model designed may be over-trained or under-trained. This is because of the samples selected for training may contain either cancerous or non-cancerous data. To avoid this, every F^{th} ($F = 10$) sample is selected for

testing, and the rest of the samples are chosen as training set using Algorithm 1. Similarly, the training set is further partitioned into learning and validation sets, using Algorithm 1.

Algorithm 1 Division of dataset in F-Fold Cross Validation

Input: The array of dataset (X).
Output: Training (D) and testing (T) dataset for each fold.

1: **for** `i=1 to F` **do**
2: **for** `j=1 to F` **do**
3: $r = (i + (j-1)N)\ mod\ size(X)$
4: $T(i,j) = X(r)$
5: $D(i,j) = X - T(i,j)$
6: **end for**
7: **end for**

where i represents the partition set for the dataset (X). After partitioning the dataset, the model is selected by performing 10-fold CV as discussed in the Algorithm 2.

Algorithm 2 F-Fold Cross Validation

1: **for** `i=1 to F` **do**
2: Divide the dataset into training set D_i and testing set T_i using Algorithm 1.
3: **WRITE THE ‘‘FOR LOOP” FOR THE VARYING PARAMETER (θ) OF CLASSIFIER.**
4: **for** `j=1 to F` **do**
5: Divide the training set (D_i) into learning set (L_j) and validation set (V_j) using Algorithm 1.
6: Train the model using learning set (L_j).
7: Validate the model using validation set (V_j).
8: Calculate accuracy of the model.
9: **end for**
10: Calculate mean of accuracy of model corresponding to Parameter (θ_i).
11: **END THE ‘‘FOR LOOP” OF VARYING PARAMETER (θ).**
12: Select θ, corresponding to model having high accuracy (called $\theta^{'}$).
13: Train the model with training set (D_i) with $\theta^{'}$ and calculate accuracy.
14: Test the model with testing set (T_i) with $\theta^{'}$ and calculate accuracy.
15: **end for**

The algorithm behaves differently with different classifiers; i.e., for different classifier the number of varying parameter (θ) is different, for that a ‘for loop’ is inserted in algorithm at line number 3 and end the $for\ loop$ at line number 11. For instance, suppose ANN is used as a classifier, for that the number of hidden nodes are varied. According to that, a $for\ loop$ is placed at line number 3 (e.g. $for\ \theta = 5:5:50$ do) and ended the for loop at line number 11 ($end\ for$). Similarly, for each classifier a for loop is added, as and when required.

After performing ‘‘10-fold CV” on the dataset, the predicted values of test data are collected in each of the fold and the classification matrix is designed. This analysis has been carried out on three microarray datasets by considering varying number of feature sets.

2.2.1.1 Different datasets used

In order to perform the classification of microarray data, three datasets viz., Leukemia, Breast and Ovarian Cancer dataset from ‘‘Kent ridge biomedical” repository are used [93]. The details of the dataset in terms of two class problem (cancerous and non-cancerous) are provided in the section 4.2.

The classifiers have been run sequentially on the varying size of feature sets [2, 100] using these microarray dataset.

2.2.1.2 Logistic regression

The results of logistic regression [102] has been validated using ‘10-fold CV’ for three dataset and are tabulated in Table 2.2. The table gives the values of statistics such as the number of features (which obtained better accuracy for the respective feature selection method), accuracy obtained for training and testing phase. In the detailed analysis Table 2.2a, 2.2b, and 2.2c give the statistics for three datasets viz., Leukemia, Breast cancer and Ovarian respectively.

From the obtained result, it is observed that LR classifier with t-test as a feature selection method achieves better accuracy of 98.61% in testing phase for Leukemia dataset for six number of features. Similarly, t-test obtained better accuracy of 80.14% and 98.81% for Breast cancer and Ovarian dataset respectively when logistic regression was applied. The values highlighted in bold font indicate the highest accuracy value obtained for the particular dataset.

Table 2.2: Result of LR with various feature selection methods.

(a) Leukemia dataset

Feature selection methods	No. of features	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**6**	**63.95**	**98.61**
F-test	46	81.10	97.22
Wilkixon test	30	83.31	97.22
SNR	84	54.55	97.22
χ^2-test	48	77.64	97.22
Information Gain	46	81.10	97.22
Gini Index	2	65.36	72.22
Fisher Score	52	74.40	97.22

(b) Breast cancer dataset

Feature selection methods	No. of features	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**16**	**78.44**	**80.41**
F-test	10	81.78	79.38
Wilkixon test	8	82.71	77.32
SNR	72	64.71	77.32
χ^2-test	82	55.00	76.29
Information Gain	14	80.04	79.38
Gini Index	2	77.07	76.29
Fisher Score	8	82.71	77.32

(c) Ovarian dataset

Feature selection methods	No. of features	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**36**	**99.52**	**98.81**
F-test	42	99.74	98.81
Wilkixon test	48	100.00	98.81
SNR	90	99.70	97.63
χ^2-test	54	99.96	98.81
Information Gain	8	97.64	98.42
Gini Index	28	99.61	97.63
Fisher Score	34	97.56	97.63

2.2.1.3 Naive Bayes

The classification results obtained using Naive Bayes [102] as a classifier with various FS methods are tabulated in Table 2.3. The table gives the values of statistics such as the number of features (which obtained better accuracy for the respective feature selection method), accuracy obtained for training and testing phase. In the detailed analysis Table 2.3a, 2.3b, and 2.3c give the statistics for three datasets viz., Leukemia, Breast cancer and Ovarian respectively.

From the obtained result, it is clear that when t-test is used as a FS method, a set of 6 features are sufficient to achieve 98.71% of training accuracy and 98.61% of testing accuracy for Leukemia dataset. Similarly, all the results can be analyzed for Breast cancer and Ovarian dataset. The values highlighted in bold font indicate the high accuracy obtained for the particular dataset.

Table 2.3: Result of NB with various feature selection methods.

(a) Leukemia dataset

Feature selection methods	No. of features	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**6**	**98.71**	**98.61**
F-test	46	97.19	97.22
Wilkixon test	30	97.19	97.22
SNR	84	97.19	97.22
χ^2-test	48	97.19	97.22
Information Gain	46	97.19	97.22
Gini Index	2	69.74	72.22
Fisher Score	52	97.19	97.22

(b) Breast cancer dataset

Feature selection methods	No. of features	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**16**	**81.03**	**80.41**
F-test	10	78.18	79.38
Wilkixon test	8	78.56	77.32
SNR	72	78.51	77.32
χ^2-test	82	77.40	76.29
Information Gain	14	78.28	79.38
Gini Index	2	76.82	76.29
Fisher Score	8	78.56	77.32

(c) Ovarian cancer dataset

Feature selection methods	No. of features	Train CV Acc. (%)	Test CV Acc. (%)
T-test	36	98.78	98.81
F-test	**42**	**98.82**	**98.81**
Wilkixon test	48	98.82	98.81
SNR	90	97.65	97.63
χ^2-test	54	98.78	98.81
Information Gain	40	98.78	98.81
Gini Index	8	98.43	98.42
Fisher Score	28	97.65	97.63

2.2.1.4 Artificial neural network

ANN is a network of simulated neurons. It is inspired by the examination of central nervous system of human body. Warren in 1943 created a computational model for neural networks based on mathematical formulation and algorithms [104]. In this analysis, sigmoid function is used in hidden as well as output layer with back propagation gradient descent learning algorithm. In this study, a varying number of input nodes are considered based on the applied feature selection technique. In hidden layer, selecting the number of hidden nodes is a challenging task. To overcome this issue, varying number of hidden nodes in multiples of five in the range 5 to 50 are considered to design a network. Next, 10 fold CV is

applied to validate the obtained results. After finding the optimal no. of hidden nodes, their corresponding training and testing accuracies in each of the fold is obtained.

In the final model, the median of all the hidden nodes in each of the fold is considered as the optimal number of hidden nodes. The average of training and testing accuracy obtained in each of the fold are considered as the final output (accuracy) of the model.

The classification results obtained using ANN as a classifier with various FS methods are tabulated in Table 2.4. The table gives the values of statistics such as the number of features (which obtained better accuracy for the respective feature selection method), accuracy obtained for training and testing phase. In the detailed analysis, Table 2.4a, 2.4b, and 2.4c give the statistics for three datasets viz., Leukemia, Breast cancer and Ovarian respectively.

From the obtained results, it is clear that when t-test is used as a FS method, a set of 40 features with 5 number of hidden nodes are sufficient enough to achieve 100% training accuracy and 97.22% testing accuracy for Leukemia dataset. Similarly, the results can be analyzed for the other two datasets. The values highlighted in bold font indicate the high accuracy obtained for the particular dataset.

Table 2.4: Result of ANN with various feature selection methods.

(a) Leukemia dataset

Feature selection methods	No. of features	No. of hidden nodes	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**40**	**5**	**100.00**	**97.22**
F-test	24	5	97.18	97.22
Wilkixon test	4	5	95.83	94.44
SNR	34	5	98.61	94.44
χ^2-test	46	5	100.00	95.83
Information Gain	14	10	99.31	93.06
Gini Index	4	5	95.83	94.44
Fisher Score	46	5	100.00	95.83

(b) Breast cancer dataset

Feature selection methods	No. of features	No. of hidden nodes	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**40**	**15**	**77.97**	**78.38**
F-test	12	15	80.42	74.23
Wilkixon test	16	15	77.83	76.29
SNR	48	25	79.94	75.26
χ^2-test	52	15	77.95	75.26
Information Gain	52	15	78.43	74.23
Gini Index	4	10	77.90	77.32
Fisher Score	4	10	77.34	76.29

(c) Ovarian cancer dataset

Feature selection methods	No. of features	No. of hidden nodes	Train CV Acc. (%)	Test CV Acc. (%)
T-test	44	5	99.40	100.00
F-test	**50**	**5**	**100.00**	**100.00**
Wilkixon test	46	5	99.21	99.60
SNR	48	10	100.00	100.00
χ^2-test	50	5	99.80	100.00
Information Gain	40	5	98.41	98.82
Gini Index	35	10	96.56	97.87
Fisher Score	42	5	95.84	97.23

2.2.1.5 Radial basis function network

Radial Basis Function Network (RBFN) was first formulated by Broomhead et al. [112] and was popularized by Moody et al. [113].

There are different training schemes for updating the parameters such as the weights and center in RBF network [105]. In this analysis, the hybrid approach is applied to train the parameters of the network. To find the cluster centers, K-means clustering algorithm is used, and the weight vector is updated using gradient descent learning algorithm.

Varying number of clusters in the range from 2 to 10 are used to find the number of hidden nodes required to obtain the maximum testing accuracy. 10 fold CV is used to validate the results. Table 2.5 shows the results of RBF network classifier using various FS methods for three datasets.

Table 2.5, gives the values of statistics such as the number of features (which obtained better accuracy for the respective feature selection method), accuracy obtained for training and testing phase. In the detailed analysis Table 2.5a, 2.5b, and 2.5c give the statistics for three datasets viz., Leukemia, Breast cancer and Ovarian respectively.

From the obtained results, it is clear that when t-test is used as a FS method, a set of 40 features with 2 number of hidden nodes are sufficient enough to achieve 98.71% training accuracy and 98.61% testing accuracy for Leukemia dataset. Similarly, the results can be analyzed for the other two datasets. The values highlighted in bold font indicate the high accuracy obtained for the particular dataset.

Table 2.5: Result of RBFN with various feature selection methods.

(a) Leukemia dataset

Feature selection methods	No. of features	No. of hidden nodes (Clusters)	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**40**	**2**	**98.71**	**98.61**
F-test	10	5	95.07	93.06
Wilkixon test	42	2	98.71	98.61
SNR	22	2	98.71	94.44
χ^2-test	38	2	97.43	97.22
Information Gain	42	2	98.71	98.61
Gini Index	2	2	88.52	90.28
Fisher Score	20	2	96.62	95.83

(b) Breast cancer dataset

Feature selection methods	No. of features	No. of hidden nodes (Clusters)	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**6**	**2**	**83.82**	**83.51**
F-test	42	4	79.10	77.32
Wilkixon test	26	8	78.33	75.26
SNR	76	4	78.68	77.32
χ^2-test	2	2	72.89	71.13
Information Gain	42	4	79.10	77.32
Gini Index	2	2	72.89	71.13
Fisher Score	16	2	79.96	81.44

(c) Ovarian cancer dataset

Feature selection methods	No. of features	No. of hidden nodes (Clusters)	Train CV Acc. (%)	Test CV Acc. (%)
T-test	14	8	98.30	98.42
F-test	50	7	98.78	98.42
Wilkixon test	**38**	**6**	**98.82**	**98.42**
SNR	48	6	98.82	98.42
χ^2-test	46	6	98.78	98.42
Information Gain	38	6	98.82	98.42
Gini Index	2	2	93.29	93.28
Fisher Score	42	6	98.78	98.02

2.2.1.6 Probabilistic neural network

Probabilistic neural network (PNN) is a four layered network (as shown in Figure 2.2), consists of input, pattern, summation, and output layer and it maps any input pattern to any number of classifications [106, 114].

- First, the distance between input vector to training vectors is computed in input layer.
- The second layer transforms the input space into nonlinear space using Gaussian function, and the center is determined from the training data. The Gaussian function is determined as using Equation 2.2.

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \sum_{k=1}^{n} exp(\frac{-(x - x_k)^2}{2\sigma^2}) \tag{2.2}$$

where n, σ, x, x_k represent the number of samples from a class, smoothing parameter, testing input and k^{th} sample respectively.

- The contribution of each class of input layer is summed up in the summation layer, to produce a net output which is vector of probabilities.
- The output layer determines the classification rate.

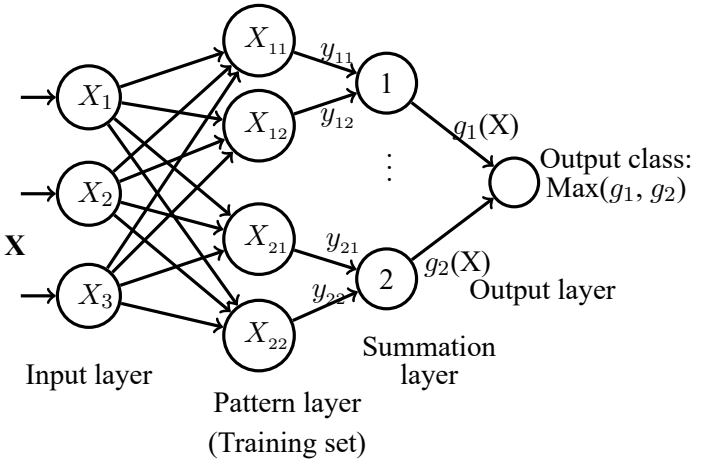


Figure 2.2: Basic structure of PNN

In PNN, 50% of cancerous and non cancerous classes are considered as input for hidden layers. Gaussian elimination is used as a hidden node function Equation 2.2. The summation layers sum contribution of each class of input patterns and produce a net output which is a vector of probabilities. The output pattern having maximum summation value is classified into respective class.

The results of PNN are evaluated by varying the value of the smoothing parameter 'σ' in the range of $(0, 1]$ using '10-fold CV'. The σ specifies the spread of activation function.

The obtained results using different datasets are evaluated on various FS methods and are shown in Table 2.6. The table gives the values of statistics such as the number of features (which obtained better accuracy for the respective feature selection method), accuracy obtained for training and testing phase. In the detailed analysis Table 2.6a, 2.6b, and 2.6c give the statistics for three datasets viz., Leukemia, Breast cancer and Ovarian respectively.

From the obtained results, it is clear that when t-test is used as a FS method, a set of 78 features with 0.2 σ value are sufficient enough to achieve 82.48% training accuracy and 84.72% testing accuracy for Leukemia dataset. Similarly, the results can be analyzed for the other two datasets. The values highlighted in bold font indicate the highest accuracy obtained for the particular dataset.

From this result it is observed that, among all the FS methods using t-test the classifier achieves better accuracy.

Table 2.6: Result of PNN with various feature selection methods.

(a) Leukemia dataset

Feature selection methods	No. of features	Best σ	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**78**	**0.2**	**82.48**	**84.72**
F-test	80	0.2	82.52	84.72
Wilkixon test	42	0.1	80.98	81.94
SNR	26	0.3	77.59	80.56
χ^2-test	46	0.1	78.45	80.56
Information Gain	36	0.1	78.69	79.17
Gini Index	2	0.2	73.45	73.61
Fisher Score	96	0.1	80.40	81.94

(b) Breast cancer dataset

Feature selection methods	No. of features	Best σ	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**6**	**0.3**	**71.79**	**73.20**
F-test	2	0.1	69.90	72.16
Wilkixon test	22	0.4	66.64	69.07
SNR	82	0.4	66.97	68.04
χ^2-test	4	0.1	70.35	72.16
Information Gain	14	0.1	67.96	68.04
Gini Index	2	0.3	69.90	72.16
Fisher Score	4	0.1	70.35	72.16

(c) Ovarian cancer dataset

Feature selection methods	No. of features	Best σ	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**34**	**0.3**	**98.82**	**98.81**
F-test	28	0.2	95.82	96.20
Wilkixon test	34	0.6	98.82	98.81
SNR	100	0.1	97.08	96.84
χ^2-test	64	0.5	98.30	98.42
Information Gain	32	0.1	98.39	98.42
Gini Index	2	0.3	90.22	89.72
Fisher Score	14	0.1	96.20	96.05

2.2.1.7 K-Nearest neighbor

The K-nearest neighbor algorithm [103] is used to classify the microarray dataset to determine the category of the cell. Here, Euclidean distance is used to measure the distance between the training and testing samples. Table 2.7 shows the results obtained using KNN classifier permuted with various FS methods using different microarray dataset. The results are evaluated by varying the number of nearest neighbor K in the range of [1,n] with a step size of 2, where n is the number of samples in the training set. From the obtained result, it is inferred that out of 50 subsets of dataset, the model with best accuracy is selected for choosing the optimal value of K.

The table gives the values of statistics such as the number of features (which obtained better accuracy for the respective feature selection method), accuracy obtained for training and testing phase. In the detailed analysis Table 2.7a, 2.7b, and 2.7c give the statistics for three datasets viz., Leukemia, Breast cancer and Ovarian respectively.

From the obtained results, it is observed that when t-test is used as a FS method, a set of 26 features with $K(=1)$ value are sufficient enough to achieve 97.19% training accuracy and 97.22% testing accuracy for Leukemia dataset. Similarly, the results can be analyzed for the other two datasets. The values highlighted in bold font indicate the highest accuracy obtained for the particular dataset. The values highlighted in bold font indicate the highest accuracy obtained for the particular dataset.

2.2.1.8 Support vector machine

In SVM [107] classifier different kernel functions such as: linear, polynomial, RBF (gaussian), and tangent sigmoid are frequently used. In this analysis, RBF kernel function

Table 2.7: Result of KNN with various feature selection methods.

(a) Leukemia dataset

Feature selection methods	No. of features	Value of 'K'	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**26**	**1**	**97.19**	**97.22**
F-test	8	3	99.12	91.67
Wilkixon test	88	3	98.21	97.22
SNR	18	1	95.95	90.28
χ^2-test	98	3	98.52	72.22
Information Gain	48	1	97.19	97.22
Gini Index	2	3	93.36	93.06
Fisher Score	54	3	97.19	97.22

(b) Breast cancer dataset

Feature selection methods	No. of features	Value of 'K'	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**6**	**7**	**80.88**	**82.47**
F-test	48	5	82.33	78.35
Wilkixon test	28	9	80.65	77.32
SNR	90	4	82.10	80.41
χ^2-test	2	10	74.58	71.13
Information Gain	66	11	81.07	80.41
Gini Index	2	10	74.58	71.13
Fisher Score	82	7	80.44	77.32

(c) Ovarian cancer dataset

Feature selection methods	No. of features	Value of 'K'	Train CV Acc. (%)	Test CV Acc. (%)
T-test	26	1	99.30	98.81
F-test	**50**	**1**	**99.39**	**98.42**
Wilkixon test	48	1	99.39	98.42
SNR	48	1	99.39	98.42
χ^2-test	100	1	99.17	98.02
Information Gain	36	1	99.35	98.42
Gini Index	26	1	99.30	98.81
Fisher Score	44	1	99.35	98.42

is used to map the input vector into high-dimensional space. The parameters of the kernel functions like $gamma$ (γ) and the penalty parameter C are selected using the grid search in the range of $[2^{-5}, 2^5]$ and $[2^{-5}, 2^5]$ respectively. The results are evaluated by varying the value of kernel parameters in the specified range using '10-fold CV'. Then the classifier is tested with different permutations of FS methods is on three datasets. The obtained accuracy has been tabulated as shown in Table 2.8 .

Table 2.8: Result of SVM with various feature selection methods.

(a) Leukemia dataset

Feature selection methods	No. of features	Best '$log_2\gamma$'	Best 'log_2C'	Train CV Acc. (%)	Test CV Acc. (%)
T-test	14	5	3	99.69	100.00
F-test	32	5	0	97.99	95.83
Wilkixon test	24	5	2	98.77	98.61
SNR	**68**	**5**	**-1**	**100.00**	**100.00**
χ^2-test	58	5	-1.5	99.69	98.61
Information Gain	62	5	-1.5	100.00	98.61
Gini Index	50	5	0	100.00	100.00
Fisher Score	64	5	-1	100.00	98.61

(b) Breast cancer dataset

Feature selection methods	No. of features	Best '$log_2\gamma$'	Best 'log_2C'	Train CV Acc. (%)	Test CV Acc. (%)
T-test	**12**	**3.5**	**-4.5**	**83.16**	**84.54**
F-test	82	5	-2	81.67	76.29
Wilkixon test	20	1	0	81.10	75.26
SNR	92	2.5	-2.5	82.81	74.23
χ^2-test	14	4	-5	83.40	81.44
Information Gain	56	2.5	-1	81.33	76.29
Gini Index	88	5	-2	81.55	77.32
Fisher Score	12	3.5	-4.5	83.16	84.54

(c) Ovarian cancer dataset

Feature selection methods	No. of features	Best '$log_2\gamma$'	Best 'log_2C'	Train CV Acc. (%)	Test CV Acc. (%)
T-test	50	5	0	99.87	100.00
F-test	50	5	0	99.87	100.00
Wilkixon test	**58**	**5**	**0**	**99.91**	**100.00**
SNR	44	5	-1	99.69	98.81
χ^2-test	68	5	-1	99.91	100.00
Information Gain	40	5	-2	99.74	99.21
Gini Index	2	3.5	2	94.99	95.26
Fisher Score	40	5	-2	99.74	99.21

From Table 2.8, it is inferred that, the number of features required corresponding to the optimal value of γ and C with their training and testing accuracy. The Table gives the values of statistics such as the number of features (which obtained better accuracy for the respective

feature selection method), accuracy obtained for training and testing phase. In the detailed analysis, Table 2.8a, 2.8b, and 2.8c give the statistics for three datasets viz., Leukemia, Breast cancer and Ovarian respectively.

From the obtained results, it is clear that when t-test is used as a FS method, a set of 14 features with $\gamma = 2^5, C = 2^3$ values are sufficient enough to achieve 99.69% training accuracy and 100.00% testing accuracy for Leukemia dataset. Similarly, the results can be analyzed for the other two datasets. The values highlighted in bold font indicate the highest accuracy obtained for the particular dataset.

2.2.1.9 Comparative analysis

In this classification analysis, emphasis was laid on designing classifier models which can obtain better classification of microarray dataset to categorize the cancer causing genes into respective classes. A two class classifier was considered, consisting of cancerous and non cancerous categories. Seven classifiers were designed with different permutation of eight feature selection methods. So, a comparative analysis was done is order to choose a better classifier among the set of designed classifiers which involves a suitable feature selection method. From the obtained results, it can be inferred that among the various classifiers used in permutation with eight different feature selection methods,results from t-test yielded better accuracy in all the designed classifier models.

2.3 Summary

Any huge dataset for a classification problem has drawbacks such as the curse of dimensionality, missing values of an attribute, presence of noise etc. To overcome these pitfalls, various machine learning techniques are successfully applied for feature selection/extraction of microarray data classification. In this chapter, an attempt was made to focus on the existing schemes available to select highly significant and relevant features in the dataset for microarray in a succinct manner. Also the work highlights the classification task carried out to classify the microarray dataset using numerous machine learning techniques. The most widely used classifiers in combination with feature selection techniques are employed for classification. The obtained results are validated by the application of 10-fold CV. On keen observation, it is revealed that features selection plays a significant role in the classification of microarray data.

The rapid growth of diseases in the present day has lead to a huge increase in the datasize with respect to different category of disease, causing enormous loss to human life. To detect a disease causing factor of a particular class at an early stage within a large space of data is very much critical. From the above analysis, it is observed that the existing platforms are not suitable to process the current scenarios of the data analytics. The complexity of classifying a particular disease into a particular class can be reduced by the use of ‘Big data’ concept.

This can be achieved through the use of techniques such as High Performance Computing (HPC), Hadoop, Spark etc. which minimizes the amount of time required for classifying a disease into particular category of a class.

From the above study, it is observed that, t-test as a feature selection method provides better results with various classifiers. In the next chapter, t-test is considered for feature selection and various kernel based classifiers are discussed.

Chapter 3

Classification of Microarray Data using Kernel based Classifiers

In this chapter, kernel based classifiers are applied to classify the microarray high-dimensional data followed by feature section using t-test. The kernel based classifiers have capability to classify the datasets which are not linearly separable. The kernel functions provides the flexibility to the linear classifier so that it can easily classify the datasets. In this study, t-test is applied to identify the precise and interesting genes which are responsible for cause of cancer. After precise identification, various classifiers like, Extreme Learning Machine (ELM), and Relevance Vector Machine (RVM), and Kernel Fuzzy Inference System (KFIS) are proposed to classify the samples in to their respective classes (i.e., cancer or normal). Further, a comparative analysis of the obtained classification accuracy, and the results of existing Support Vector Machine (SVM) available in the literature are presented. The behavior of the classifiers are analyzed using various mocroarray datasets like Leukemia, Ovarian and Breast cancer. The performance of the classifiers are measured using performance parameters such as precision, recall, specificity, F-Measure, and accuracy.

3.1 Introduction

In recent years, Deoxyribonucleic acid (DNA) microarray technique has shown great impact in determining the informative genes that cause cancer [115, 116]. The major pitfall which is persistent in microarray data is the 'curse of dimensionality' problem [117]. This problem hinders the useful information of data set and leads to computational instability. Therefore, the selection/extraction of relevant features (genes) remains imperative in the analysis of microarray data of cancer.

A good number of feature (gene) selection/extraction techniques and classifiers based on machine learning techniques have been proposed by various researchers and practitioners [3–7].

The statistical tests which can be categorized as either parametric or non-parametric can be used for feature selection method by assuming the hypotheses [118]. Based on the correctness of the hypothesis (Null hypothesis or Alternate hypothesis), the features are either selected or rejected. Further, classification of data to their respective classes is

performed.

Extreme Learning Machine (ELM) is a variant of artificial neural network (ANN) specifically, single-hidden layer feed forward networks, which recently has gained lot of popularity due to its faster learning rate when compared to conventional machine learning techniques [119].

Relevance Vector Machines (RVM) is one of the machine learning technique which has an better edge in comparison to SVM among the research community [120, 121]. RVM work flow is based on the Bayesian formulation of a linear model with an appropriate assumption that results in a sparse representation. As a result, it can be well generalized and can provide inferences at low computation cost. RVM has an identical functionality in comparison to SVM, but rather it uses a Bayesian probabilistic model for learning and performing predictions.

Fuzzy logic provides a means to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. Since the nature of data set is quite fuzzy i.e., not predictable, which in turn (data) leads to different inference. The relationship among the data and inference is unknown. The fuzzy concept has been used in this work, to study the behavior of the data (capturing human way of thinking), and also it is also possible to represent and describe the data mathematically. Further, fuzzy system has been considered because of the limited number of learning rules that needs to be learnt in the present system. The no. of free parameters to be learnt is reduced considerably, leading to efficient computation. In general, if the number of features are larger than 100, then it is suitable to use machine learning techniques rather than using statistical approaches.

If ANN is applied for the same method, designing the model would be far more challenging due to the large no. of cases. Hence coupling ANN with Fuzzy logic, will be easy to handle by inferring the rule base of the fuzzy system. In the current scenario, neuro-fuzzy networks have found to be successfully applied in various areas of analytics. Two typical types of neuro-fuzzy networks are Mamdani-type [122] and TSK-type [123]. For Mamdani-type neuro-fuzzy networks, minimum number of fuzzy implications are used in fuzzy reasoning. Meanwhile, in TSK-type neuro-fuzzy networks, the consequence of each rule is a function of various input variables. The generic adopted function for rule generation, is a linear combination of input variables and constant term. Several researchers and practitioners have reported that, using TSK-type neuro-fuzzy network achieves superior performance in network size and learning accuracy than that of Mamdani-type neuron-fuzzy networks [124]. In classic TSK-type neuro-fuzzy network, which is linear polynomial of the input variables, the system output is approximated locally by the rule of hyperplanes.

However, a linear subspace cannot describe the non-linear variations of microarray genes. Alternatively, a kernel feature space can reflect non-linear information of genes. By using the kernel trick, the data points are mapped into a higher dimensional (possibly infinite-dimensional) space [125]. Kernel trick is a mathematical technique which can be

applied to any dot product based algorithms. Whenever a dot product between two vectors is encountered, it is replaced by kernel function. This maps candidate linear algorithms into non-linear algorithms (sometimes with little effort or reformulation). Further, the transformed non-linear algorithms are the equivalent of their linear algorithm in their original feature space.

In this chapter, the following type of kernels have been used to map the function in high-dimensional space.

- **Linear:** $K(x_i, x_j) = \gamma x_i^T x_j$.
- **Polynomial:** $K(x_i, x_j) = (x_i^T x_j + b)^\gamma, \gamma > 0$.
- **Radial Basis Function (RBF):** $K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2), \gamma > 0$.
- **Tansigmoid (Tansig):** $K(x_i, x_j) = tanh(\gamma x_i^T x_j + b), \gamma > 0$.

where, γ and b are *kernel parameters*.

The choice of a kernel function depends on the problem in hand because it depends on what we are trying to model. For instance, a polynomial kernel allows feature conjunction modeling to the order of the polynomial. Radial basis function allows to pick out circles (or hyper spheres) in contrast with the linear kernel, which allows only to pick out lines (or hyperplanes). The objective behind using the choice of a particular kernel can be very intuitive and straightforward depending on what kind of information is to be extracted with respect to data.

Hence, along with the feature selection using t-statistic, ELM, RVM, and a non-linear version of FIS called kernel fuzzy inference system (KFIS), which is a kernerlized version of neuro-fuzzy system with different kernel functions are used as classifiers by applying 10-fold cross validation (CV). We have already shown the state of art simulation of existing methods in the earlier chapter, where SVM (with RBF Kernel) performed very well with better accuracy and in less time. Therefore, the motivation of this chapter is:

- To analyze the microarray data using classifier with better accuracy in a minimum processing time.
- Idea: Various classifiers with different kernels are applied to analyze the microarray dataset.

The rest of the chapter is organized as follows: Section 3.2 presents the procedure for classifying the microarray data using various proposed classifiers. Section 3.3 presents the implementation details of the proposed approach. Section 3.5 highlights on the results obtained, and the interpretation drawn from it. Section 3.6 summarizes the chapter and presents the scope for future work.

3.2 Proposed work

This section presents the proposed approach for classification of microarray data, consisting of two phases:

1) Pre-process the input data using methods such as missing data imputation, normalization, and feature selection using *t*-statistic.

2) Applying ELM, RVM and KFIS with different kernel functions as a classifier.

Figure 3.1 shows the block diagram of the proposed approach, and the brief description is as follows:

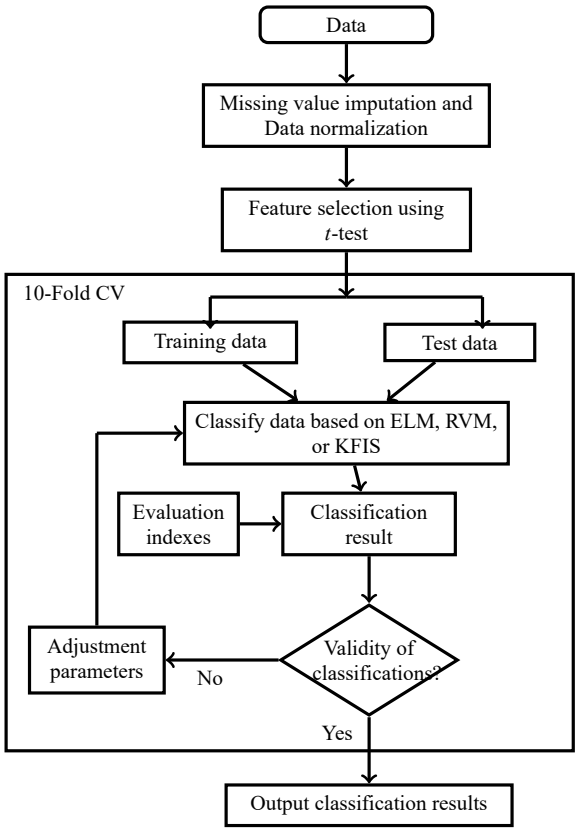


Figure 3.1: Proposed work for microarray classification.

1. **Data collection:** The data set for classification, which is the requisite input to the models is obtained from Kent Ridge Bio-medical Dataset Repository [109].

2. **Missing data imputation and normalization of dataset:**

   Missing data of a feature (gene) of microarray data is imputed by using the *mean* value of the respective feature. Input feature values are normalized over the range $[0, 1]$ using Min-Max normalization technique [126]. Let X_i be the i_{th} feature of the dataset X,

and x is an element of the X_i. The normalization of x can be formulated as:

$$Norm(x) = \frac{x - \min(X_i)}{\max(X_i) - \min(X_i)} \tag{3.1}$$

where, $\min(X_i)$ and $\max(X_i)$ represent the minimum and maximum value for the dataset X_i respectively. If in case, $\max(X_i)$ is equal to $\min(X_i)$, then $Norm(x)$ is set to 0.5.

3. **Feature selection from dataset:**

   t-test statistic has been applied to select the features having high relevance value and hence the curse of dimensionality issue has been reduced.

4. **Division of dataset:** The data set is divided into two categories such as: training set and testing set using Algorithm 1.

5. **Application of a classifier:**

   ELM, RVM, and KFIS with different kernel functions are applied to classify the microarray dataset.

6. **Testing:**

   Models are tested using the testing dataset. The performance of the classifier is measured using precision, recall, specificity, F-Measure, and accuracy parameters (discussed in Section 4.3). Also ‘‘10-fold cross validation” technique is applied to validate the model [95].

3.3 Implementation

3.3.1 Feature selection using *t*-test

Generally, the problem with microarray data are (a) ‘‘curse of dimensionality”, where numbers of features are much larger than the number of samples (b) there are so many features having very less effect on the classification result, etc. To alleviate these problems, feature selection approaches are used [94]. In this chapter, *t*-test filter approach is used to overcome the problems. Selecting features using *t*-test is to reduce the dimension of the data by finding a small set of important features which can give good classification performance, and is computed using Equation 3.2.

$$TS(i) = \frac{\bar{X}_{i1} - \bar{X}_{i2}}{s_{X_1X_2}\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{3.2}$$

$$s^2_{X_1X_2} = \frac{(n_1 - 1)s^2_{X_{i1}} + (n_2 - 1)s^2_{X_{i2}}}{n_1 + n_2 - 2} \tag{3.3}$$

where $s_{X_1X_2}$ is an estimator of the common standard deviation of the two samples $\bar{X}_{ik}$ represents the mean of feature i of class $k \in \{1, 2\}$ and s is the standard deviation.

A widely-used filter method for microarray data is to apply a univariate criterion separately on each feature, assuming that there is no interaction between features. A two-class problem test of the null hypothesis (H_0) is that the mean of two populations are equal; it means that there is no significant difference between their means, and both features are almost same. It implies that they (features) do not affect much on the classification result. Hence, these features have been discarded; and the features having significant difference between their means are accepted. Therefore, it is necessary to reject 'null hypothesis' (H_0) and accept the 'alternate hypothesis' (H_1). In other words, alternate hypothesis is accepted. Here, t-test on each feature has been applied and compared with their corresponding p-value (or the absolute values of t-statistics) for each feature as a measure of how effective it is at separating groups. In order to get a general idea of how well-separated the two groups (classes) are by each feature, the empirical cumulative distribution function (CDF) of the p-values has been plotted in Figure 3.2.

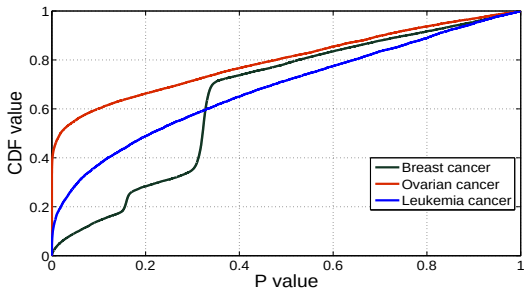


Figure 3.2: Empirical cumulative distribution function (CDF) of the p-values

The features having p-values smaller than 0.05 have strong discrimination power. Sorting these features according to their p-values (or the absolute values of the t-statistic) helps to identify some features from the sorted list.

From Figure 3.2, it is observed that

- In case of Leukemia, about 18% of features have p-values close to zero, and over 28.70% of features have p-values smaller than 0.05. Also 71.30% i.e., 5083 out of 7129 features have p-value greater than or equal to 0.05 (95% Confidence interval).
- In case of breast cancer, about 10.44% features have p-values smaller than 0.05; and 89.56% i.e., 21924 out of 24481 features have p-values greater than or equal to 0.05.
- In case of ovarian cancer, about 55.19% (8363) features have p-values smaller than 0.05; and 44.81% i.e., 6791 out of 15154 features have p-values greater than or equal to 0.05.

However, it is usually difficult to decide how many features are needed unless one has some domain knowledge or the maximum number of features that can be considered have been dictated in advance based on outside constraints. To overcome this problem, forward feature selection method is considered, in which top ranked features corresponding to their ascending p-value are identified.

3.3.2 Extreme learning machine (ELM) classifier

ELM is a variant of single-hidden layer feed forward networks (SLFNs), which recently has gained lot of popularity due to its faster learning rate when compared to conventional machine learning techniques [119]. In ELM, hidden neurons need not be tuned and these hidden neurons can be randomly generated; thus, it leads to reduction in training time which helps to 'learn' at an extremely high speed.

In conventional learning method, one must check the training data before generating the parameters for hidden neurons; whereas in case of ELM, the parameters for hidden neurons can be generated before checking the training data. Further, in ELM there is no scope to fine tune parameters such as learning rate, momentum, epochs, etc.

ELM classifier in comparison to back propagation (traditional gradient learning algorithm) overcomes several issues like local minima, improper learning rate and over fitting. It tends to reach the solutions in straightforward manner without any such trivial issues.

The basic architecture of ELM classifier is shown in Figure 3.3, which comprises of an input layer, hidden layer and an output layer similar to that of single hidden layer feed forward neural network.

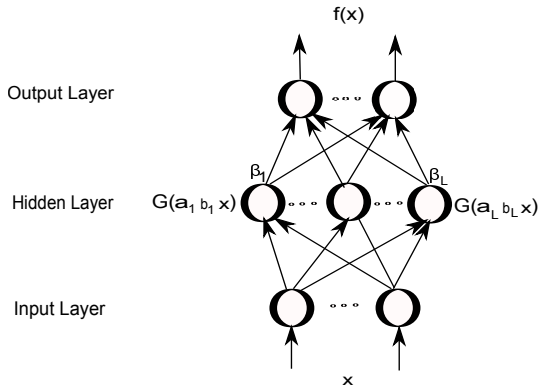


Figure 3.3: Structure of ELM

The output function $f_L(x)$ of ELM for generalized SLFNs is given below as shown in equation 3.4.

$$f_L(x) \;=\; \sum_{i=1}^{L} \beta_i G(a_i, b_i, x) \;=\; \beta.h(x) \tag{3.4}$$

where L is the number of hidden layer neurons, and β_i is the weight vector representing

the connection between the i^{th} hidden neuron and the output neuron. $G(a_i, b_i, x)$ is the output/activation function of the i^{th} hidden neuron, which can be either sigmoid, sine, radial basis, or any nonlinear piecewise continuous activation function, as ELM works for all nonlinear piecewise continuous activation function. The hidden layer mapping ($h(x)$), i.e., the output vector of hidden layer with respect to the input x can be represented as $h(x)$ = $[G(a_i, b_i, x), ..., G(a_L, b_L, x)]$.

Let $X = \{(x_i, t_i)|x_i \in \mathbf{R}^d, t_i \in \mathbf{R}^c, i = 1, 2, ..., N\}$, is training sample with L be the number of hidden neurons and $G(a, b, x)$ be the hidden neuron activation function, the output function of ELM can be mathematically formulated as shown in equation 3.5.

$$f_L(x) \;=\; \sum_{i=1}^{L} \beta_i G(a_i, b_i, x_j) = t_j \tag{3.5}$$

where $j = 1, ..., N$. Equation 3.5 can be expressed as $H\beta = T$, where,

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ \vdots & \vdots & \vdots \\ G(a_1, b_1, x_N) & \cdots & G(a_L, b_L, x_N) \end{bmatrix} \tag{3.6}$$

$$\beta \;=\; \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix} \;\; and \;\; T \;=\; \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix} \tag{3.7}$$

here, H is the hidden layer output matrix. The output weight vector β can be calculated as:

$$\beta \;=\; H^{\dagger}T \tag{3.8}$$

where, $H^{\dagger}$ is the Moore-Penrose generalized inverse of hidden layer output matrix H. In order to obtain a more stable and generalized solution, the regularization factor C can be set. This helps in making ELM more stable. After considering the factor C, the output weight vector β can be calculated using equation 3.9.

$$\beta \;=\; H^T(\frac{1}{C} + HH^T)^{-1}T \tag{3.9}$$

and the output function of ELM corresponding to output weight vector β in equation 3.9 is shown in equation 3.10.

$$f_L(x) \;=\; h(x)\beta \;=\; h(x)H^T(\frac{1}{C} + HH^T)^{-1}T \tag{3.10}$$

similarly, the output function of ELM corresponding to output weight vector β is shown in Equation 3.11.

$$f_L(x) \;=\; h(x)\beta \;=\; h(x)H^T(\frac{1}{C} + H^TH)^{-1}T \tag{3.11}$$

When hidden layer mapping matrix is known, the basic ELM classifier can be used. The solution is different when hidden layer mapping matrix is unknown.

If hidden layer mapping matrix is unknown, one can use kernel based ELM, where kernel matrix for ELM can be defined as shown below in equation 3.12.

$$\Omega_{i,j} \; = \; h(x_i).h(x_j) \; = \; k(x_i, x_j) \tag{3.12}$$

The output function $f_L(x)$ for kernel based ELM is expressed in equation 3.13.

$$f_L(x) \;\; = \;\; \begin{bmatrix} k(x, x_1) \\ . \\ . \\ . \\ k(x, x_N) \end{bmatrix} (\frac{1}{C} + \Omega_{ELM})^{-1} T \tag{3.13}$$

3.3.3 Relevance vector machine (RVM) classifier

This section briefly discusses the formulation of RVM [127]. Let $X = \{(x_i, t_i)|x_i \in \mathbf{R}^d, t_i \in \mathbf{R}^c, i = 1, 2, ..., N\}$ be the pair of input data ($\mathbf{x}_i$) with scalar valued target label (t_i). The design of RVM follows a Bayesian probabilistic model for learning and the predictions are based on the function

$$y = {}^T\Phi(\mathbf{x}) \tag{3.14}$$

where $= (\omega_0, \omega_1, \omega_2, \ldots, \omega_N)^T$ is the weight matrix, $\Phi(\mathbf{x}) = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)]^T$ is a set of basis functions, for $\phi(\mathbf{x}_n) = \{\ 1, K(\mathbf{x}_n, \mathbf{x}_1), K(\mathbf{x}_n, \mathbf{x}_2), \ldots, K(\mathbf{x}_n, \mathbf{x}_N)\ \}$ with $K(*, *)$ is the kernel function which can be of the form Gaussian, Euclidean, Laplacian, etc. The output of RVM (y) is a linear combination of weighted basis functions. The weights () are computed during training and the training samples corresponding to non-zero weights are called *relevance vectors* (RVs). The objective of learning the classifier is to predict the posterior probability of class membership for the given input $\mathbf{x}$. The linear model in Equation (3.14) is generalized by applying the logistic sigmoid function $\sigma(y) = 1/(1+e^{-y})$ to y and adopting the Bernoulli distribution to define the likelihood as

$$P(\mathbf{t}|) = \prod_{n=1}^{N} \sigma\{y(\mathbf{x}_n,)\}^{t_n}[1 - \sigma\{y(\mathbf{x}_n,)\}]^{1-t_n} \tag{3.15}$$

To obtain the marginal likelihood analytically, Mackay's iterative procedure [128] is used which is based on the Laplace's method. Let α be the vector of hyperparameters and each individual α value is associated with every weight value. For the fixed values of α, the most probable weights $\mathbf{w}$ are found, giving location of the mode of posterior distribution [127]. Since

$$p(|\mathbf{t}, \alpha) \propto P(\mathbf{t}|)p(|\alpha), \tag{3.16}$$

this is equivalent to finding the maximum of

$$\log\{P(\mathbf{t}|)p(|\alpha)\} = \sum_{n=1}^{N}[t_n \log y_n + (1-t_n)\log(1-y_n)] - \frac{1}{2}{}^T\mathbf{A} \tag{3.17}$$

over , where $y_n = \sigma\{y(\mathbf{x}_n,)\}$. Equation (3.17) is differentiated twice to obtain

$$\nabla\nabla \log p(|\mathbf{t}, \alpha)|_{\mathbf{w}} = -(\Phi^T\mathbf{B}\Phi + \mathbf{A}) \tag{3.18}$$

where $\mathbf{A} = \text{diag}(\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_N)$ and $\mathbf{B} = \text{diag}(\beta_1, \beta_2, \ldots, \beta_N)$ with $\beta_n = \sigma\{y(\mathbf{x}_n)\}[1 - \sigma\{y(\mathbf{x}_n)\}]$. The covariance matrix Σ and the posterior over weights centered at (**w**) are defined as

$$\Sigma = (\Phi^T\mathbf{B}\Phi + \mathbf{A})^{-1} \tag{3.19}$$

$$\mathbf{w} = \Sigma\Phi^T\mathbf{B}\mathbf{t} \tag{3.20}$$

Using Σ and **w**, the hyper-parameters are updated using

$$\alpha_i = \frac{\gamma_i}{\mathbf{w}_i^2} \tag{3.21}$$

where $\mathbf{w}_i$ is the i^{th} posterior weight computed using Equation (3.20), $\gamma_i \equiv 1 - \alpha_i\Sigma_{ii}$ and Σ_{ii} is the i^{th} diagonal element of Σ. Similarly, β is updated using

$$\beta = \frac{N - \sum_i \gamma_i}{\|\mathbf{t} - \Phi\mathbf{w}\|^2} \tag{3.22}$$

The convergence criteria for the above iterative procedure is defined as

$$\delta = \sum_{i=1} \alpha_i^{n+1} - \alpha_i^n \tag{3.23}$$

Re-estimation stops when $\delta < \delta_\tau$, where δ_τ is the threshold value for change of α between iterations. The training samples corresponding to $\neq 0$ are termed as relevance vectors (**R**). The weights and relevance vectors obtained from Algorithm 3 are used to find an estimate of the target value pertaining to the new input x'

$$y' = \mathbf{w}^T\phi(x') \tag{3.24}$$

3.3.4 Fuzzy inference system (FIS)

For a given universe set U of objects, a conventional binary logic (crisp) A is defined by specifying the objects of U that are member of A. In other words, the characteristic function

Algorithm 3 Training: RVM

Input: Input matrix $X = \{(x_i, t_i)|x_i \in \mathbf{R}^d, t_i \in \mathbf{R}^c, i = 1, 2, ..., N\}$, where N is the number of samples with dimension d, $\mathbf{t}$: corresponding target values
Output: $\mathbf{R}$: Relevance vectors of model, $\mathbf{w}$: weight matrix, and Generate $\Phi = \phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)$

Initialize δ_τ
Initialize α, β ▷ Initialisation of hyper-parameters
repeat
$\mathbf{A} = \mathrm{diag}(\alpha), \mathbf{B} = \mathrm{diag}(\beta)$
$\Sigma = (\Phi^T \mathbf{B} \Phi + \mathbf{A})^{-1}$
$\mathbf{w} = \Sigma \Phi^T \mathbf{B} \mathbf{t}$
$\gamma_i \equiv 1 - \alpha_i \Sigma_{ii}$
$\alpha_i = \frac{\gamma_i}{\mathbf{w}_i^2}$
$\beta = \frac{N - \sum_i \gamma_i}{\|\mathbf{t} - \Phi \mathbf{w}\|^2}$
$\delta = \sum_{i=1} \alpha_i^{n+1} - \alpha_i^n$
$\mathbf{R} = \mathbf{x}(\mathbf{w}_{\mathrm{index}})$ ▷ Training samples corresponding to non-zero weights
until $\delta < \delta_\tau$

Algorithm 4 Testing: RVM

Input: x': Testing data for classification, $\mathbf{R}$: Relevance vectors, $\mathbf{w}$: Weight matrix
Output: y': Predicted class membership.

Generate $\phi(x')$ using $\mathbf{R}$
$y' = \mathbf{w}^T \phi(x')$

of A can be written as $u_A : U \to \{0, 1\}$ for all $x \in U$.

Fuzzy sets are obtained by generalizing the concept of characteristic function to a membership function $u_A : U \to [0, 1]$ for all $x \in U$. It provides the degree of membership rather than just the binary *'is'/'is not'* a member to a set; which ensures the objects that are not clearly member of one class or another. Using crisp techniques, an ambiguous object will be assigned to one class only lending an aura of precision and definiteness to the assignment that are not warranted. On the other hand, fuzzy techniques will specify to what degree the object belongs to each class.

The TSK fuzzy model (FIS) is an adaptive rule model introduced by T. Takagi, M. Sugeno and K. T. Kang in 1984 [123]. The main objective of using TSK fuzzy model is to reduce the number of rules generated by Mamdani model[124]. In this approach, TSK fuzzy model can also be used for classifying complex and high-dimensional problems. It develops a systematic approach to generate fuzzy rules from a given input-output data set. TSK model replaces the fuzzy sets of the Mamdani rule with the function of the input variables.

3.3.5 Kernel fuzzy inference system (KFIS)

In this section, KFIS has been described which is a non-linear version of FIS. The number of rules (R), the parameters of fuzzy sets i.e., the centers (c) and the width parameters (σ)

of the corresponding membership function (in this case Gaussian) of KFIS are computed using kernel subtractive clustering technique (KSC) which is also a non-linear version of subtractive clustering (SC) and the parameters of rules are computed using least mean square (LMS) algorithm in non-linear space. The stepwise working procedure of KFIS has been depicted in Figure 3.4. The working procedure of KFIS is described as follows:

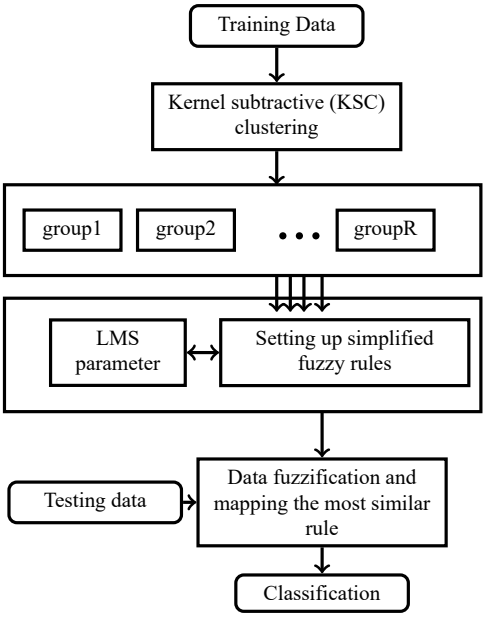


Figure 3.4: Framework of Kernel fuzzy inference system (K-FIS).

1. **Clustering:**

   To compute the parameters of the membership function i.e., centroids (c) and the width parameter (σ) and number of rules (centers), Kernel subtractive clustering (KSC) has been used on training data set (microarray). The algorithm of KSC has been described in the section 3.3.5.1.

2. **Setting up a simplified fuzzy rule base**

   - **Computation of membership function**

     Gaussian function is used as a membership function (A). The parameters such as centroid (c) and the width parameter (σ) of A have been computed using KSC and A is expressed as:

     $$A = exp\left(-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2\right) \tag{3.25}$$

   - **Generation of fuzzy rules**

     The number of fuzzy rules generated will be equal to the number of clusters formed.

3. **Estimation of parameters of rules**

   After generating fuzzy rules, the constant parameters in rules can be estimated using Least Mean Square (LMS) algorithm.

3.3.5.1 Kernel subtractive clustering (KSC)

The kernel subtractive clustering (KSC) is a non-linear version of subtractive clustering [129], here input space is mapped into non-linear space. In this algorithm, to obtain the cluster centroids and sigmas, the same parameters are used which are also used in subtractive clustering (SC) [130]. The parameters used to calculate the cluster centroid are *Hypersphere cluster radius (r_a) in data space*, *Reject ratio ($\bar{\epsilon}$)*, *Accept ratio (ϵ)*. Reject ratio ($\bar{\epsilon}$) specifies a threshold for the potential value above which the data point is definitely accepted as a cluster centroid. Accept ratio (ϵ) specifies a threshold below which the data point is definitely rejected. Squash factor (η) defines the neighborhood which will have the measurable reductions in potential value, and it can be calculated as:

$$\eta = \frac{r_b}{r_a} \tag{3.26}$$

For a given data point $x_i \subset X$ where $(1 \leq i \leq n)$, $X \in \mathbb{R}^p$ and a non-linear function ϕ: $\mathbb{R}^p \rightarrow \mathbb{H}$ maps the input to a higher (may be infinite) dimensional feature space $\mathbb{H}$. The potential value of each data point defines a measure of the data point to serve as a cluster centroid and can be calculated by using the following Equation:

$$\begin{aligned} p(x_i) &= \sum_{j=1}^{n} e^{-\alpha \|\phi(x_i)-\phi(x_j)\|^2} \\ &= \sum_{j=1}^{n} e^{-\alpha(K(x_i,x_i)-2K(x_i,x_j)+K(x_j,x_j))} \end{aligned} \tag{3.27}$$

where $\alpha = 4/r_a^2$, K is a kernel function, $\|.\|$ denotes the Euclidean distance between the data points, and r_a is a positive constant called cluster radius. The data point with highest potential is selected as the first cluster centroid by computing the potential value of individual data point. Let x_1^* be the centroid of the first cluster and p_1^* its potential value. The potential value of each data point x_i^* is revised as follows:

$$\begin{aligned} p_j(x_i) &= p_{j-1}(x_i) - p_{j-1}^* e^{-\beta \left\| \phi(x_i)-\phi(x_{j-1}^*) \right\|^2} \\ &= p_{j-1}(x_i) - p_{j-1}^* e^{-\beta(K(x_i,x_i)-2K(x_i,x_{j-1}^*)+K(x_{j-1}^*,x_{j-1}^*))} \end{aligned} \tag{3.28}$$

where $p_j^* = Max_i(p(x_i))$, $\beta = 4/r_b^2$, $r_b = \eta * r_a$ and η is a positive constant over the range $[1, 2]$. When the potentials of all data points have been revised by Equation 3.28, the

data point with the highest remaining potential is selected as the second cluster centroid. In such a manner, all the cluster centroids are selected using Algorithm 5.

Algorithm 5 Kernel subtractive clustering (KSC)

Input: The dataset X, radius $r_a, \eta, \epsilon, \bar{\epsilon}$.
Output: Optimal number of clusters, their centroid (c) and the width parameter (σ).

1: Compute the potential for each data point x_i using Equation 3.27.
2: Choose the data point x_i whose potential value is highest as a cluster centroid.
3: Discard and recompute the potential value for each x_i using Equation 3.28.
4: **if** $p_j^* > \epsilon p_1^*$ **then**
5: Accept x_j^* as a cluster center and continue.
6: **else if** $p_j^* < \bar{\epsilon} p_1^*$ **then**
7: Reject x_j^* and end the clustering process.
8: **else**
9: d_{min} = shortest of the distance between x_j^* and all previously found cluster centers.
10: **if** $\frac{d_{min}}{r_a} + \frac{p_j^*}{p_1^*} \geq 1$ **then**
11: Accept x_j^* as a cluster center and continue.
12: **else**
13: Reject x_j^* and set the potential at x_j^* to 0. Select the data point with the next highest potential as the new x_j^* and reset.
14: **end if**
15: **end if**
16: $\sigma = (r_a * (Max(X) - Min(X))/\sqrt{8.0}$

After computing the number of rules (R), the parameters of fuzzy sets and the parameters of rules are derived. To derive the rules for the KFIS, the dataset with selected features (genes) using filter approach (t-test) has been used as the input. The k^{th} rule (R^k) for the given test point x_t can be expressed as:

IF x_1 is A_1^k and x_2 is A_2^k,... and x_n is A_n^k; where x_1, x_2,... x_n are input variables and A_j^k is a fuzzy set, R^k is a linear function. The fuzzy set A_j^k uses a Gaussian function and can be computed as:

$$A_j^k = exp\left(-\frac{1}{2}\left(\frac{\phi(x_j) - \phi(c_{jk})}{\sigma_{jk}}\right)^2\right)$$

$$= exp\left(-\frac{1}{2\sigma_{jk}^2}\left(K(x_j, x_j) - 2K(x_i, c_{jk}) + K(c_{jk}, c_{jk})\right)\right) \tag{3.29}$$

THEN

$$R^k = b_0^k + \sum_{t=1}^{n} p_t^k \phi(x_t) \tag{3.30}$$

$$p_t^k = \sum_{i=1}^{m} \alpha_i \phi(x_i) \tag{3.31}$$

Consider m to be the number of training samples, and ϕ as a non-linear transformation function. The representer theorem [131, 132] states that the solution of an optimization of Equation 3.31, can be written in the form of an expansion over training pattern, (x_i is replaced by $\phi(x_i)$). Therefore, each training vector lies in the span of $\phi(x_1), \phi(x_2), ...\phi(x_m)$ and Lagrange multiplier α_i, where $i = 1, 2, ...m$ [133]. Therefore, Equation 3.30 is expressed as:

$$\begin{aligned} R^k(x_t) &= b_0^k + \sum_{t=1}^{n}\sum_{i=1}^{m} \alpha_i \phi(x_i)\phi(x_t) \\ &= b_0^k + \sum_{t=1}^{n}\sum_{i=1}^{m} \alpha_i K(x_i, x_t) \end{aligned} \tag{3.32}$$

The degree (firing strength) with which the input matches k^{th} rule is typically computed using 'and' operator:

$$\mu_k = \prod_{j=1}^{n} A_j^k \tag{3.33}$$

In this case, each rule is a crisp output. The overall output is calculated using the weighted average as shown in Equation 3.34.

$$Y = \frac{\sum_i^R \mu_i R_i}{\sum_i^R \mu_i} \tag{3.34}$$

where R is the number of rules and R_i is the i^{th} fuzzy rule where $i = 1, 2, ..., R$. For KFIS classification algorithm, the probability $\hat{y}$ of output Y can be calculated using Equation 3.35 [134].

$$\hat{y} = (1 + exp(-Y))^{-1} \tag{3.35}$$

Using the usual kernel trick, the inner product can be substituted by kernel functions satisfying Mercer's condition. Substituting the expansion of p in equation 3.31, into equation 3.30, this transformation leads to nonlinear generalization of fuzzy inference system in kernel space which can be called as kernel fuzzy inference system (KFIS).

The parameters used in the KFIS are shown in Table 3.1.

Table 3.1: Parameters of KFIS model

Parameters used	Range	Value used
Squash factor (η)	[1, 2]	1.25
Accept ratio (ϵ)	(0, 1]	0.75
Reject ratio ($\bar{\epsilon}$)	(0, 1]	0.15
Cluster radius (r_a)	(0, 1]	—

3.4 Performance Measures

This section highlights the performance parameters used for classification. Table 3.2 shows the confusion matrix which provides the statistics for the number of correct and incorrect predictions made by a classifier compared to that of the actual classifications of the samples in the test data [135, 136]. The confusion matrix is a table that shows the output and actual (target) class classification accomplished by the classifier. The corresponding measures of performance are represented in Table 3.3.

Table 3.2: Confusion matrix

		Output class	
		Classified as **neg**	Classified as **pos**
Target class	**neg**	tn	fp
	pos	fn	tp

Table 3.3: Performance parameters

Performance parameters	**Description**
$Precision = \frac{tp}{fp+tp}$	It is the degree to which the repeated measurements under unchanged conditions show the same results
$Recall = \frac{tp}{fn+tp}$	It indicates the number of the relevant items are to be identified
$F - Measure = \frac{2*Precision*Recall}{Precision+Recall}$	It combines the 'precision' and 'recall' numeric values to give a single score, which is defined as the harmonic mean of the precision and recall.
$Specificity = \frac{tn}{fp+tn}$	It focuses on how effectively a classifier identifies negative labels.
$Accuracy = \frac{tp+tn}{fp+fn+tp+tn}$	It measures the percentage of inputs in the test set that the classifier correctly labeled.

3.5 Results and Interpretation

In this section, the obtained results are discussed for the proposed work. Three case studies viz., Leukemia [109], Ovarian cancer [137] and Breast cancer [110] microarray datasets are considered to find the classification accuracy. ''10 fold cross validation (CV)" (discussed in Algorithm 2) is applied to assess the performance of the classifier, as it provides more realistic assessment of classifiers. Cross validation generalizes the model significantly to unseen data.

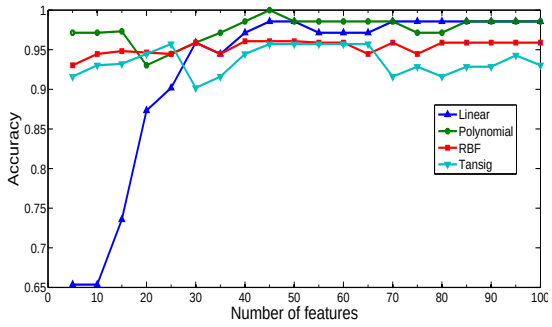
After performing feature selection using *t*-test, the classification algorithms ''ELM", ''RVM" and ''KFIS" have been applied to classify the reduced dataset. After performing ''10-fold CV", the predicted values of test data are collected in each of the fold and the confusion matrix is designed with their respective feature set using the proposed classifiers. This analysis has been carried out on three different microarray datasets by considering varying number of feature sets. The feature sets are varied in the multiple of five i.e., 5, 10, 15, 20, ...

The proposed classifiers have been implemented using various kernel functions viz., Linear, Polynomial, RBF and Tansig. The gamma (γ) and C values are selected by searching in the

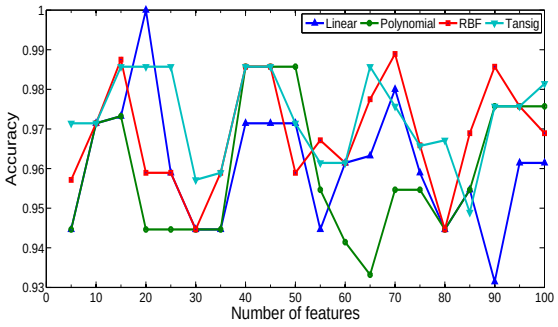
range of 2^{-5} to 2^{5}; and $r_a \in (0, 1]$ (for KFIS only) in each fold. Finally, the median value of the best γ, C, and r_a from each fold is considered as the value of γ, C, and r_a for the final model. By using these values, the performance of classifiers are evaluated on test data.

3.5.1 Case study: Leukemia cancer dataset

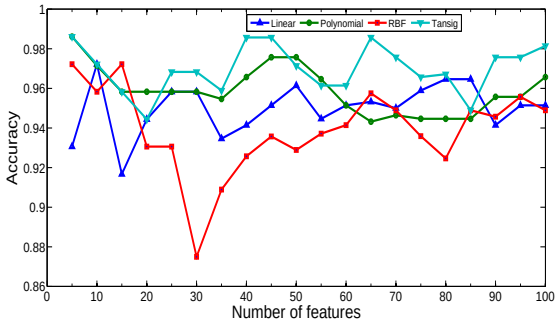
The Leukemia dataset consists of 72 samples and 7129 features (genes). The samples are categorized into two classes viz., Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML) [109]. Out of 72 samples, the dataset contains 25 AML and 47 ALL samples. The ELM, RVM, and KFIS classifiers with various kernel functions have been run sequentially on the varying size of feature sets for Leukemia dataset.



(a) Classification accuracy of ELM classifier



(b) Classification accuracy of RVM classifier



(c) Classification accuracy of KFIS classifier

Figure 3.5: Accuracy of classifiers using Leukemia dataset by varying the number of features

Figure 3.5a shows the accuracy of ELM classifier with different kernel functions by varying the number of features for Leukemia dataset. From Figure 3.5a, it is observed that maximum accuracy (minimum CV error) has been acquired when feature set with 45, 45, 40, and 45 features are selected using ELM classifier with linear, polynomial, RBF, and tansig kernel function respectively. After attaining the peak value, the accuracy of ELM classifier either remains constant or reduces from maximum accuracy. Therefore, to avoid the curse of dimensionality problem, the feature set with 45, 45, 40, and 45 features are selected using ELM classifier with linear, polynomial, RBF, and tansig kernel function respectively. The rest of the performance parameters are tabulated in Table 3.4.

Table 3.4: Performance analysis of kernel based ELM classifiers using Leukemia dataset.

kernel	Accuracy	Precision	Recall	Specificity	F-measure
Linear Function	0.9861	1.0000	0.9600	1.0000	0.9795
Polynomial Function	1.0000	1.0000	1.0000	1.0000	1.0000
Tansig Function	0.9583	1.0000	0.8800	1.0000	0.9361
Radial Basis Function	0.9583	0.8928	1.0000	0.9361	0.9433

Similarly, Figure 3.5b shows the accuracy of RVM classifier with different kernel functions by varying the number of features using Leukemia dataset. From Figure 3.5b, it is clear that maximum accuracy (minimum CV error) has been acquired when feature set with 20, 40, 15, and 15 features are selected using RVM classifier with linear, polynomial, RBF, and tansig kernel function respectively. After attaining the peak, the accuracy of RVM classifier either remains constant or reduces from maximum accuracy. Therefore, to avoid the curse of dimensionality problem, the feature set with 20, 40, 15, and 15 features are selected using RVM classifier with linear, polynomial, RBF, and tansig kernel function respectively. The rest of the performance parameters are tabulated in Table 3.5.

Table 3.5: Performance analysis of kernel based RVM classifiers using Leukemia dataset.

Kernel	Accuracy	Precision	Recall	Specificity	F-measure
Linear Function	0.9861	1.0000	0.9600	1.0000	0.9795
Polynomial Function	0.9722	1.0000	0.9200	1.0000	0.9583
Tansig Function	0.9861	1.0000	0.9600	1.0000	0.9796
Radial Basis Function	0.9722	0.9600	0.9600	0.9787	0.9600

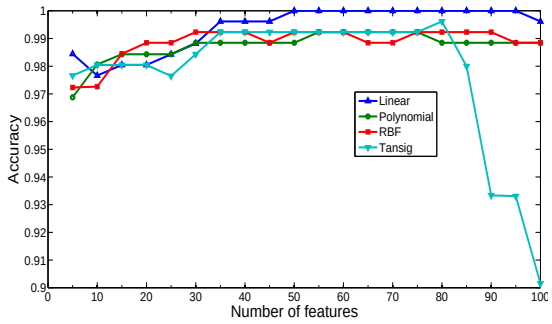
Similarly, Figure 3.5c shows the accuracy of KFIS classifier with different kernel functions by varying the number of features using Leukemia dataset. From Figure 3.5b, it is clear that maximum accuracy (minimum CV error) has been acquired when feature set with 10, 5, 5, and 5 features are selected using RVM classifier with linear, polynomial, RBF, and tansig kernel function respectively. After attaining the peak, the accuracy of KFIS classifier either remains constant or reduces from maximum accuracy. Therefore, to avoid the curse of dimensionality problem, the feature set with 10, 5, 5, and 5 features are selected using KFIS classifier with linear, polynomial, RBF, and tansig kernel function respectively. The rest of the performance parameters are tabulated in Table 3.6.

Table 3.6: Performance analysis of kernel based KFIS classifier using Leukemia dataset.

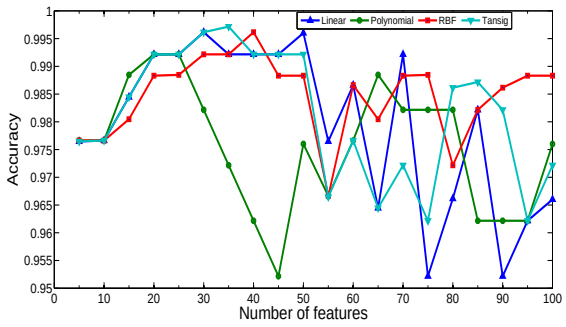
Kernel/#Features/r_a	Accuracy	Precision	Recall	Specificity	F-measure
Linear/10/0.2	0.9722	0.9600	0.9600	0.9787	0.9600
Polynomial/5/0.2	0.9861	0.9615	1	0.9787	0.9804
Tansig/5/0.2	0.9861	1	0.9600	1	0.9796
RBF/5/0.4	0.9722	0.9600	0.9600	0.9787	0.9600

3.5.2 Case study: Ovarian cancer

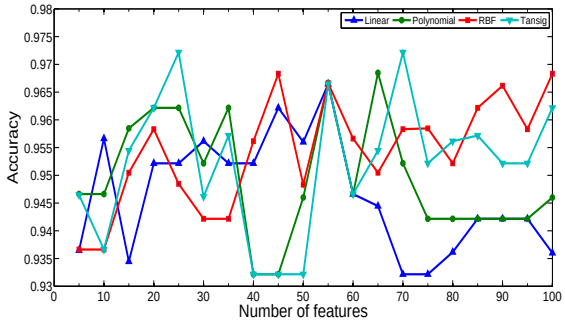
The ovarian cancer dataset consists of 253 samples and 15154 features (genes). The samples are categorized as 'cancer' and 'normal' classes. Out of 253 samples, the dataset contains 162 cancer and 91 normal samples [137].



(a) Classification accuracy of ELM classifier



(b) Classification accuracy of RVM classifier



(c) Classification accuracy of KFIS classifier

Figure 3.6: Accuracy of classifiers with various number of features using Ovarian dataset

Figure 3.6a shows the accuracy of ELM classifier with different kernel functions by varying the number of features using Ovarian dataset. From Figure 3.6a, it is clear that maximum accuracy (minimum CV error) has been acquired when feature set with 50, 55, 30, and 80 features are selected using ELM classifier with linear, polynomial, RBF, and tansig kernel function respectively. After attaining the peak value, the accuracy of ELM classifier either remains constant or reduces from maximum accuracy. Therefore, to avoid the curse of dimensionality problem, the feature set with 50, 55, 30, and 80 features are selected using ELM classifier with linear, polynomial, RBF, and tansig kernel function respectively. The rest of the performance parameters are tabulated in Table 3.7.

Table 3.7: Performance analysis of kernel based ELM classifiers using Ovarian dataset.

Kernel	Accuracy	Precision	Recall	Specificity	F-measure
Linear Function	1.0000	1.0000	1.0000	1.0000	1.0000
Polynomial Function	0.9920	1.0000	0.9780	1.0000	0.9888
Tansig Function	0.9970	1.0000	0.9890	1.0000	0.9944
Radial Basis Function	0.9920	1.0000	0.9780	1.0000	0.9888

Similarly, Figure 3.6b shows the accuracy of RVM classifier with different kernel functions by varying the number of features using Ovarian dataset. From this figure, it is evident that maximum accuracy (minimum CV error) has been acquired when feature set with 30, 50, 40, and 30 features are selected using RVM classifier with linear, polynomial, RBF, and tansig kernel function respectively. After attaining the peak, the accuracy of RVM classifier either remains constant or reduces from maximum accuracy. Therefore, to avoid the curse of dimensionality problem, the feature set with 30, 50, 40, and 35 features are selected using RVM classifier with linear, polynomial, RBF, and tansig kernel function respectively. The rest of the performance parameters are tabulated in Table 3.8

Table 3.8: Performance analysis of kernel based RVM classifier using Ovarian dataset.

kernel	Accuracy	Precision	Recall	Specificity	F-measure
Linear Function	0.9960	1.0000	0.9890	1.0000	0.9945
Polynomial Function	0.9960	1.0000	0.9890	1.0000	0.9945
Tansig Function	0.9960	1.0000	0.9890	1.0000	0.9944
Radial Basis Function	0.9960	1.0000	0.9890	1.0000	0.9944

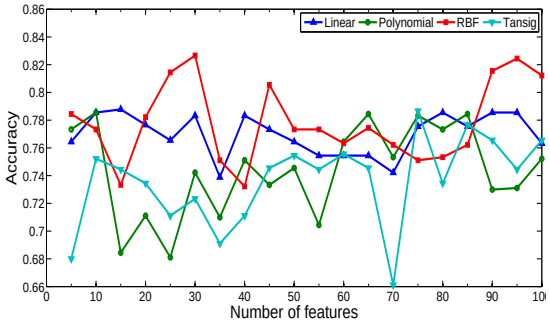
Similarly, Figure 3.6c shows the accuracy of KFIS classifier with different kernel functions by varying the number of features using Ovarian dataset. From this figure, it is evident that maximum accuracy (minimum CV error) has been acquired when feature set with 55, 65, 45, and 25 features are selected using RVM classifier with linear, polynomial, RBF, and tansig kernel function respectively. After attaining the peak, the accuracy of KFIS classifier either remains constant or reduces from maximum accuracy. Therefore, to avoid the curse of dimensionality problem, the feature set with 55, 65, 45, and 25 features are selected using RVM classifier with linear, polynomial, RBF, and tansig kernel function respectively. The rest of the performance parameters are tabulated in Table 3.8

Table 3.9: Performance analysis of kernel based KFIS classifier using Ovarian dataset.

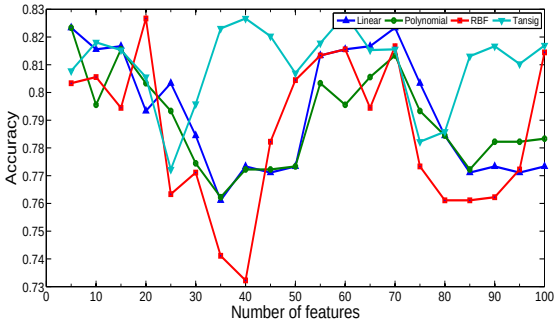
Kernel/#Features/r_a	Accuracy	Precision	Recall	Specificity	F-measure
Linear/55/0.2	0.9646	0.9556	0.9451	0.9755	0.9503
Polynomial/65/0.3	0.9684	0.9560	0.9560	0.9753	0.9560
Tansig/25/0.3	0.9723	0.9667	0.9560	0.9815	0.9613
RBF/45/0.4	0.9684	0.9663	0.9451	0.9815	0.9556

3.5.3 Case study: Breast cancer

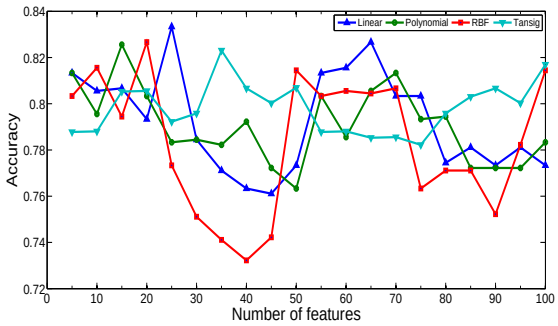
The breast cancer dataset consists of 97 samples and 24481 features (genes). The samples are categorized as 'relapse' and 'non-relapse' classes [110]. Out of 97 samples, the dataset contains 46 relapse and 51 no-relapse samples. Figure 3.7a shows the accuracy of ELM classifier with different kernel functions by varying the number of features using Breast cancer dataset. From Figure 3.7a, it is clear that maximum accuracy (minimum CV error) has been acquired when feature set with 15, 10, 30, and 75 features are selected using



(a) Classification accuracy of ELM classifier



(b) Classification accuracy of RVM classifier



(c) Classification accuracy of KFIS classifier

Figure 3.7: Accuracy of classifiers with various number of features using Breast cancer dataset

ELM classifier with linear, polynomial, RBF, and tansig kernel function respectively. After attaining the peak, the accuracy of ELM classifier either remains constant or reduces from maximum accuracy. Therefore, to avoid the curse of dimensionality problem, the feature set with 15, 10, 30, and 75 features are selected using ELM classifier with linear, polynomial, RBF, and tansig kernel function respectively.

Table 3.10: Performance analysis of kernel based ELM classifiers using Breast cancer dataset.

kernel	Accuracy	Precision	Recall	Specificity	F-measure
Linear Function	0.7835	0.7885	0.8039	0.7608	0.7961
Polynomial Function	0.7835	0.7419	0.9019	0.6521	0.8141
Tansig Function	0.7835	0.7777	0.8235	0.7391	0.8000
Radial Basis Function	0.8247	0.8148	0.8627	0.7826	0.8380

Similarly, Figure 3.7b shows the accuracy of RVM classifier with different kernel functions by varying the number of features using Breast cancer dataset. From this Figure, it is clear that maximum accuracy (minimum CV error) has been acquired when feature set with 5, 5, 20, and 40 features are selected using RVM classifier with linear, polynomial, RBF, and tansig kernel function respectively. After attaining the peak, the accuracy of RVM classifier either remains constant or reduces from maximum accuracy. Therefore, to avoid the curse of dimensionality problem, the feature set with 5, 5, 20, and 40 features are selected

using RVM classifier with linear, polynomial, RBF, and tansig kernel function respectively. The rest of the performance parameters are tabulated in Table 3.11.

Table 3.11: Performance analysis of kernel based RVM classifier using Breast cancer dataset.

kernel	Accuracy	Precision	Recall	Specificity	F-measure
Linear Function	0.8247	0.8695	0.7843	0.8696	0.8247
Polynomial Function	0.8454	0.9091	0.7843	0.9130	0.8421
Tansig Function	0.8267	0.8695	0.8039	0.8478	0.8283
Radial Basis Function	0.8247	0.8542	0.8039	0.8478	0.8283

Similarly, Figure 3.7c shows the accuracy of KFIS classifier with different kernel functions by varying the number of features using Breast cancer dataset. From this Figure, it is clear that maximum accuracy (minimum CV error) has been acquired when feature set with 25, 15, 10, and 35 features are selected using RVM classifier with linear, polynomial, RBF, and tansig kernel function respectively. After attaining the peak, the accuracy of KFIS classifier either remains constant or reduces from maximum accuracy. Therefore, to avoid the curse of dimensionality problem, the feature set with 25, 15, 10, and 35 features are selected using KFIS classifier with linear, polynomial, RBF, and tansig kernel function respectively. The rest of the performance parameters are tabulated in Table 3.12

Table 3.12: Performance analysis of kernel based KFIS classifier using Breast cancer dataset.

Kernel/#Features/r_a	Accuracy	Precision	Recall	Specificity	F-measure
Linear/25/0.4	0.8333	0.8302	0.8627	0.8043	0.8462
Polynomial/15/0.3	0.8247	0.8269	0.8431	0.8043	0.8350
Tansig/35/0.2	0.8247	0.8148	0.8627	0.7826	0.8381
RBF/10/0.3	0.8144	0.8113	0.8431	0.7826	0.8269

3.5.4 Comparative analysis

This section presents the comparative analysis performed for the three datasets using ELM, RVM and KFIS classifiers. The system configuration used in this analysis are as follows:

- Running time of the classification algorithm depends on number of features (genes) and number of training data points.
- Running time was recorded using MATLAB'13a on Intel Core(TM) i7 Processor with 3.40GHz speed and RAM of 4GB.

In this analysis, it is found that the performance (accuracy) of the four kernels varied depending on the type of data set (whether Leukemia, Breast or Ovarian cancer) used by the three classifiers viz., ELM, RVM, and KFIS. So the interpretation that can be drawn for the comparative analysis is as follows:

- From Table 3.4, it is evident that Polynomial kernel obtained better accuracy for Leukemia dataset when compared to both the used dataset and the kernel. Similarly from Table 3.7 and Table 3.10 it is can be inferred that Linear and RBF kernels obtained better accuracy for Ovarian and Breast cancer data sets respectively in case of ELM classifier.
- From Table 3.5, it is evident that tansig kernel obtained better accuracy for Leukemia dataset when compared to both the used dataset and the kernel. Similarly from Table 3.8 and Table 3.11 it is can be inferred that Polynomial kernels obtained better accuracy for both the Ovarian and Breast cancer data set in case of RVM classifier.
- The KFIS classifier gives the comparable results with the SVM.
- Table 3.13, Table 3.14, and Table 3.15 show the detailed comparison of ELM, RVM, KFIS, and SVM classifier in terms of average training, average testing accuracy and CPU time (in seconds). by considering varying numbers of feature sets. The median value of the best γ, C, and r_a from each fold is considered for each of the classifiers with different kernel functions.
- It can be realized that feature selection as well as the choice of kernels play a significant role in the classification of microarray data.

Table 3.13: Average training, average testing accuracy and CPU time (in Seconds) of ELM, RVM, KFIS, and SVM with different kernel functions for Leukemia Dataset.

ELM	Linear kernel		Polynomial kernel		RBF kernel		Tansig kernel	
	C=0.03125		γ = 3, C=0.03125		γ =2, C=0.03125		γ =0.0625, C=4	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	99.69(0.0030)	98.61(0.00063)	100(0.0096)	100(0.0018)	100(0.0055)	95.83(0.0014)	96.44(0.0059)	95.83(0.0013)
RVM	C=1		γ = 4 , C= 16		γ= 2, C= 32		γ = 3, C=32	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	99.38	98.61(42.10003)	98.85	97.22(90.52742)	99.41	97.22(88.17989)	98.85	98.61(39.90865)
KFIS	C=1, r=0.2		γ = 3 , C= 0.5, $r_a = 0.2$		γ= 0.5, C= 1, $r_a = 0.4$		γ = 0.5, C=0.1, $r_a = 0.2$	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	97.81	97.22 (7.6)	98.55	98.61 (44.3)	99.24	97.22 (5.5)	98.71	98.61 (41.7)
SVM	C=1.5		γ = 3, C=32		γ= 8, C=32		γ =0.125, C=24	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	99.69	100(128)	98.30	98.75(218)	99.38	100(114)	100	100(133)

Further, the trade-off between accuracy and time has been shown in Figure 3.8, where the performance of classifiers have been averaged over the all three datasets. From this figure it is concluded that,

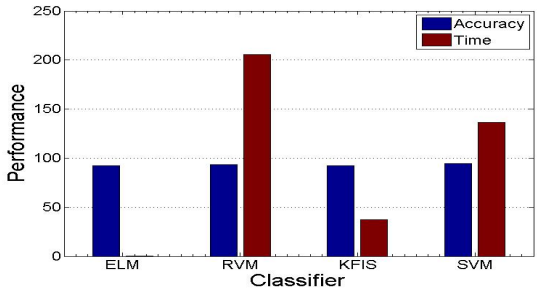
- There is no significant difference between accuracy of classifiers.
- ELM is very faster than all three classifiers, i.e., the processing time taken by classifier are in the order like $Time_{SVM} > Time_{RVM} > Time_{KFIS} > Time_{ELM}$.
- Hence, The performance of ELM is best among all these classifiers to process the microarray datasets.

Table 3.14: Average training, average testing accuracy and CPU time (in Seconds) of ELM, RVM, KFIS, and SVM with different kernel function for Breast cancer Dataset.

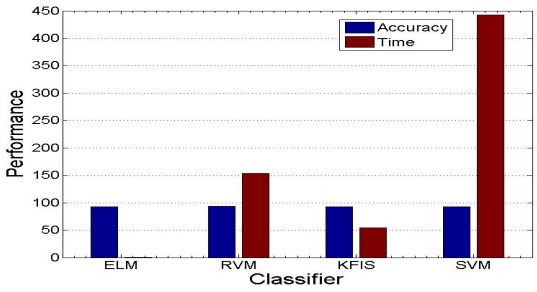
	Linear kernel		Polynomial kernel		RBF kernel		Tansig kernel	
ELM	C=0.03125		γ = 0.75, C=0.51563		γ = 0.3125, C=0.28125		γ = 0.046875 , C=24	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	80.18(0.0043)	78.35(0.00058)	82.45(0.0058)	78.35(0.00091)	97.72(0.0081)	82.47(0.0018)	76.99(0.0083)	78.35(0.0020)
RVM	C=2		γ = 0.0625, C= 32		γ = 4, C= 16		γ = 4, C= 16	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	81.42	82.47(79.52637)	83.29	84.54(64.57821)	81	82.47(105.4732)	83.29	82.67(95.54732)
KFIS	C=2, r_a = .4		γ = 3, C= .5, r_a = .3		γ = 4, C= 2, r_a = .3		γ = .5, C= 0.5 r_a = .0.2	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	85.42	83.33(09.11)	86.29	82.25(13.87)	81	81.55(20.45)	84.29	82.30(15.34)
SVM	C=32		γ = 0.125, C=32		γ = 1, C=4		γ = 0.5, C=32	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	82.02	83.44(135)	81.11	80.67(938)	82.71	81.56(185)	83.27	84.44(177)

Table 3.15: Average training, average testing accuracy and CPU time (in Seconds) of ELM, RVM, KFIS, and SVM with different kernel function for Ovarian cancer Dataset.

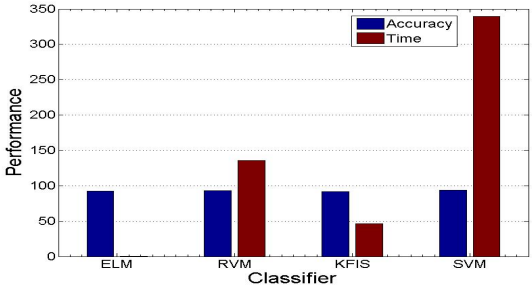
	Linear kernel		Polynomial kernel		RBF kernel		Tansig kernel	
ELM	C=0.03125		γ = 0.6, C=0.03125		γ = 0.03125, C=0.03125		γ = 0.03125, C=4.8	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	100(0.019)	100(0.0028)	99.03(0.054)	99.20(0.0058)	100(0.039)	99.21(0.0059)	99.61(0.043)	99.60(0.0067)
RVM	C=32		γ = 32, C=32		γ = 1, C=32		γ = 0.06255, C=32	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	99.13	99.60(495.7972)	99.61	99.60(305.1218)	99.44	99.60(213.1664)	99.43	99.60(203.2343)
KFIS	C=4, r_a = .2		γ = 3, C=0.5 ,r_a = .3		γ = 1, C=2, r_a = .4		γ = 0.06255, C=4, r_a = .3	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	98.45	96.64(95.84)	97.61	96.84(105.56)	99.44	96.83(113.45)	99.58	97.21(130.32)
SVM	C=32		γ = 32, C=32		γ = 1, C=32		γ = 0.06255, C=32	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
	99.52	100(146)	99.43	99.23(171)	99.86	100(720)	99.77	84.44(177)



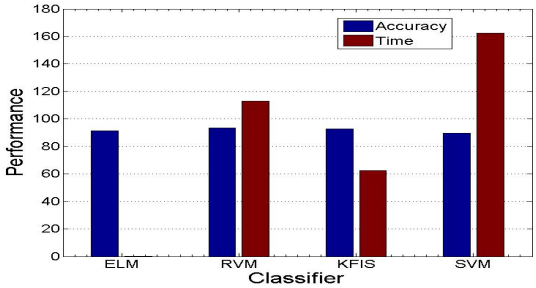
(a) Linear Kernel



(b) Polynomial Kernel



(c) RBF Kernel



(d) Tansig Kernel

Figure 3.8: Comparison of performance of classifier with different kernel functions

3.6 Summary

In this chapter, an attempt has been made to design classification models for classifying the samples of microarray data into their respective classes. Hence, a classification framework was designed using ELM, RVM, and KFIS classifier with various kernel functions. Feature selection was carried out using *t*-test. 10-fold CV technique was applied to enhance the performance of the classifiers. The performance of the classifiers for all three datasets were evaluated using performance parameters. From the computed results, it is observed that ELM with RBF kernel, RVM with polynomial kernel, and KFIS with RBF kernel as classifier yields better result when compared with ELM, RVM, and KFIS with remaining kernel functions and the existing classifier like SVM.

This chapter helps to induce a thought to solve the classification problem, when data size further increases. The rapid growth of diseases in the present day has lead to a huge increase in the datasize with respect to different category of disease, causing enormous loss to human life. To detect a disease causing factor of a particular class at an early stage within a large space of data is very much critical. The complexity of classifying a particular disease into a particular class can be reduced by the use of 'Big data' concept since the analysis is based on large pool of data. This can be achieved through the use of techniques such as High Performance Computing (HPC), Hadoop etc. which intend to minimize the amount of time required for classifying a disease into a particular category of class.

Chapter 4

Feature selection of Microarray Data using Scalable Statistical tests

In this chapter, various statistical tests are implemented on scalable cluster. The methods are able to handle and analyze the high-dimensional Big data in a distributed manner. The scalable statistical tests are applied on these datasets to select the relevant features, and the selected features are used for the further analysis.

4.1 Introduction

In recent years, the DNA microarray technique has made a great impact in determining *informative genes* that cause cancer [115, 116]. The major drawback that exists in microarray data analysis is the curse of dimensionality problem, which hinders usefulness of information from a dataset and leads to computational instability [117]. Therefore, the selection/extraction of relevant features (genes) remains an imperative task in analyzing cancer microarray datasets, which is a critical step towards effective classification.

In literature, many feature (gene) selection/extraction techniques have been proposed by various researchers and practitioners [138]. Meanwhile, recent developments in microarray chip technology have helped in assaying thousands/millions of genes simultaneously, generating a huge amount of data. However, it is difficult to process the data on a conventional system (where, data are stored on a standalone machine) with standard computational power.

In recent years, Big data applications have increasingly become the focus of attention because of the enormous increase in data generation and storage that has taken place. Extracting information from the large sized pool of data becomes a challenge because current data mining techniques are not adapted to the new space and time requirements. To overcome these challenges, many paradigms like MapReduce, Spark, etc. have been considered for developing scalable algorithms. Researchers have accomplished a significant array of relevant, intelligent techniques in data science development; where these approaches deal with imprecision, uncertainty, learning, and evolution in posing and solving computational problems [11].

Qian et al. [18] proposed a parallel and hierarchical attribute reduction method that

can be applied to Big data to analyze the intended data more efficiently. This model is able to mine decision rules under different levels of granularity. The proposed algorithms are implemented in the Hadoop framework using MapReduce, which can distribute the data and tasks in parallel and efficiently deal with Big data. Li et al. [17] developed a dominance-based rough sets approach, which is an extension of classical rough sets theory, for selecting relevant features efficiently. It processes information within the preference-ordered attribute domain and then uses it for multi-criteria decision analysis. Ayadi et al. [12] used the concept of biclusters, using DNA gene expression for microarray data. Initially, they considered a new tree structure, called the modified bicluster enumeration tree (MBET), in which biclusters are represented by the profile shapes of genes. In the next phase, they proposed an algorithm called BiMine+, which uses a pruning rule to avoid both trivial biclusters and the combinatorial explosion of the search tree. The performance of BiMine+ was assessed on both synthetic and real DNA microarray datasets. Triguero et al. [16] described the MROSEFW-RF algorithm based on a MapReduce parallelization strategy. This algorithm ensembles several highly scalable re-processing and mining methods. It performs the balancing of class distribution, detects cost-relevant features, and builds an appropriate random forest model. Islam et al. [13] proposed a MapReduce-based parallel gene selection method, that utilizes sampling techniques to reduce irrelevant genes by using the ratio of between-group to within-group sums of squares (BW). The BW ratio indicates the variances among gene expression values. After gene selection, it applies the MRkNN technique to execute multiple kNN in parallel using the MapReduce programming model. Finally, the effectiveness of the method was verified through extensive experiments using several real and synthetic datasets. Wang et al. [14] proposed a new method for calculating correlation and introduced an efficient algorithm based on MapReduce to optimize storage and correlation calculations. This algorithm is used as a basis for optimizing high-throughput molecular data (microarray data) correlation calculations. He et al. [15] described a parallel implementation of several classification algorithms (e.g., k-nearest neighbor, naive Bayesian models, and decision trees), which were executed concurrently on various clusters using the iris dataset.

The statistical tests like t-test, ANOVA, Wilcoxon ranked sum test, Kruskal-Wallis test, and Friedman test can be considered for features selection, but only three tests viz., ANOVA, Kruskal-Wallis test, and Friedman test are considered due to following advantages.

- For multiclass problem, t-test and Wilcoxon ranked sum test cannot be applied.
- For binary class problem, t-test is similar to ANOVA and Wilcoxon ranked sum test is similar to Friedman test, but Wilcoxon ranked sum test is not applicable for multiclass problem and Friedman test is not applicable for binary class problem.

Hence, only three tests viz., ANOVA, Kruskal-Wallis test, and Friedman test are considered for feature selection, which covers all the requirements of this research work.

This chapter emphasizes on the way of handling the Big data in an efficient manner. Various statistical methods like ANOVA, Kruskal-Wallis, and Friedman tests based on MapReduce and Spark are proposed and applied to select the features which are most informative and able to distinguish the diseases. These proposed methods are able to process the Big data data in distributed manner on various nodes of cluster. By applying these methods as feature selection methods, the curse of dimensionality issue has been also resolved.

The rest of the work is organized as follows: Section 4.4 presents the details of the experimental setup, which is used to for this project work. Section 4.2 described the dataset description used for the experiment. Section 4.5 presents the implementation details for the proposed feature selection approach. Section 4.6 discusses on the results obtained, interpretation drawn from it and also presents the comparative analysis for feature selection of various microarray datasets. Section 4.7 summarizes the chapter.

4.2 Datasets Used

The proposed algorithms are evaluated on the various benchmarked datasets taken from the NCBI GEO repository (NCBI GEO, `http://www.ncbi.nlm.nih.gov/gds/`) and Kent Ridge Bio-medical Dataset Repository [93]. The datasets are divided into training and testing set using the Algorithm 1. The detailed description of the datasets are presented as follows.

4.2.1 Leukemia cancer

The Leukemia dataset (file size= 6 MB) consists of expression profiles of 72 samples and 7129 features (genes) of each sample, categorized as acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML) classes. It has 47 ALL and 25 AML samples [109]. Table 4.1 represents the number of samples in each class for training set and testing set of dataset Leukemia.

Table 4.1: Class label, number of training and testing samples in each class of Leukemia dataset

Leukemia Classes	Class Label	#Training Samples	#Testing Samples
ALL	0	32	16
AML	1	16	8

4.2.2 Breast cancer

The breast cancer (data size = 20.2 MB) dataset consists of 24481 features (genes). It has 97 samples, categorized as ‘relapse’ and ‘non-relapse’ classes. Out of 97 samples, the dataset

contains 46 relapse and 51 no-relapse samples [139]. Table 4.2 represents the number of samples in each class for training set and testing set of dataset Breast cancer.

Table 4.2: Class label, number of training and testing samples in each class of Breast cancer dataset

Breast Classes	Class Label	#Training Samples	#Testing Samples
relapse	0	30	16
non-relapse	1	34	17

4.2.3 Ovarian cancer

The ovarian cancer dataset (file size= 30.2 MB) consists of 253 samples and 15154 features (genes) of each sample. It is categorized as 'cancer' and 'normal' classes. Out of 253 samples, the dataset contains 162 cancer and 91 normal samples [111]. Table 4.3 represents the number of samples in each class for training set and testing set of dataset Ovarian cancer.

Table 4.3: Class label, number of training and testing samples in each class of Ovarian cancer dataset

Ovarian Classes	Class Label	#Training Samples	#Testing Samples
cancer	0	108	54
normal	1	60	31

4.2.4 Dataset with accession number GSE24080

The dataset MULTMYEL (accession number GSE24080) has 559 samples and 54,675 features. It contains 2 classes (groups) and is of 493 MB in size [140]. Table 4.4 represents the number of samples in each class for training set and testing set of dataset GSE24080.

Table 4.4: Class label, number of training and testing samples in each class of GSE24080 dataset

GSE24080 Classes	Class Label	#Training Samples	#Testing Samples
cancer	0	124	78
normal	1	148	109

4.2.5 Dataset with accession number GSE13159

This comprises 2,096 samples in which each data sample is further represented by 54,675 feature dimensions. Again, the samples are categorized into 18 different classes. The

downloaded file is of size 1.93 GB [141]. Table 4.5 represents the number of samples in each class for training set and testing set of dataset GSE13159.

Table 4.5: Class label, number of training and testing samples in each class of GSE13159 dataset.

GSE13159 Classes	Class Label	#Training Samples	#Testing Samples
ALL with hyperdiploid karyotype	1	26	14
ALL with t(12;21)	2	39	19
ALL with t(1;19)	3	24	12
AML complex aberrant karyotype	4	32	16
AML with inv(16)/t(16;16)	5	19	9
AML with normal karyotype + other abnormalities	6	234	117
AML with t(11q23)/MLL	7	25	13
AML with t(15;17)	8	25	12
AML with t(8;21)	9	26	14
CLL	10	299	149
CML	11	51	25
MDS	12	137	69
Non-leukemia and healthy bone marrow	13	49	25
Pro-B-ALL with t(11q23)/MLL	14	47	23
T-ALL	15	116	58
c-ALL/Pre-B-ALL with t(9;22)	16	81	41
c-ALL/Pre-B-ALL without t(9;22)	17	158	79
mature B-ALL with t(8;14)	18	9	4

4.2.6 Dataset with accession number GSE13204

This is a super-series with a sample size of 1.96 GB that comprises the following two sub-series. Table 4.6 represents the number of samples in each class for training set and testing set of dataset GSE13204.

- GSE13159 Microarray Innovations in LEukemia (MILE) study: Stage 1 data
- GSE13164 Microarray Innovations in LEukemia (MILE) study: Stage 2 data

The former sub-series contains 54,675 features whereas the latter has only 1,428 features. These two are combined with appropriate genes to form the final super-series with 3,248 samples and 1,428 genes per sample. It is also grouped into 18 classes [141–143].

4.2.7 Dataset with accession number GSE15061

This is a subset composed of MILE study stage 1 data of size 650 MB. It has 870 samples and 54,675 features per sample, which are classified into three different classes [144]. Table 4.7 represents the number of samples in each class for training set and testing set of dataset GSE15061.

The datasets are summarized in Table 4.8. The next stage extracts suitable features from

Table 4.6: Class label, number of training and testing samples in each class of GSE13204 dataset.

LEukemia Classes	Class Label	#Training Samples	#Testing Samples
ALL with hyperdiploid karyotype	1	50	25
ALL with t(12;21)	2	81	41
ALL with t(1;19)	3	31	15
AML complex aberrant karyotype	4	48	24
AML with inv(16)/t(16;16)	5	32	16
AML with normal karyotype + other abnormalities	6	340	171
AML with t(11q23)/MLL	7	37	18
AML with t(15;17)	8	38	19
AML with t(8;21)	9	37	19
CLL	10	457	228
CML	11	79	40
MDS	12	218	109
Non-leukemia and healthy bone marrow	13	88	44
Pro-B-ALL with t(11q23)/MLL	14	62	31
T-ALL	15	169	84
c-ALL/Pre-B-ALL with t(9;22)	16	123	61
c-ALL/Pre-B-ALL without t(9;22)	17	263	132
mature B-ALL with t(8;14)	18	12	6

Table 4.7: Class label, number of training and testing samples in each class of GSE15061 dataset

GSE15061 Classes	Class Label	#Training Samples	#Testing Samples
disease state: AML	1	269	135
disease state: MDS	2	219	109
disease state: none-of-the-targets	3	92	46

Table 4.8: Microarray dataset used

Dataset	#Samples	#Features	#Classes	Data size	#Training samples	#Testing samples
Leukemia [109]	72	7129	2	6 MB	48	24
Ovarian Cancer [111]	253	15154	2	30.2 MB	168	85
Breast Cancer [139]	97	24481	2	20.2 MB	64	33
GSE24080 [140]	559	54675	2	493 MB	372	187
GSE13159 [141]	2096	54675	18	1.93 GB	1397	699
GSE13204 [142]	3428	1480	18	1.96 GB	2165	1083
GSE15061 [144]	870	54675	3	650 MB	580	290

both training and test sets using the aforementioned feature selection (FS) methods. The reduced set, obtained from the training set after feature selection, is then applied to model a classifier using stratified 3-fold cross validation. The developed model is then applied to the reduced test dataset to assign their respective labels.

4.3 Performance parameters

The performance of the classifiers/models are measured using the various parameters like:

- **Confusion matrix:** It provides the statistics for the number of correct and incorrect predictions made by a classification model and is compared with the actual classifications of the samples in the test data [135]. The confusion matrix is a table that shows the output and actual (target) class classification accomplished by the classifier. The confusion matrix for two class problem and corresponding measures of performance are represented in Table 4.9.

Table 4.9: Confusion matrix

		Target class		
		neg	**pos**	
Output class	Classified as **neg**	tn	fn	$npv = \frac{tn}{tn+fn}$
	Classified as **pos**	fp	tp	$Precision = \frac{tp}{tp+fp}$
		$Specificity = \frac{tn}{tn+fp}$	$Recall = \frac{tp}{tp+fn}$	$Accuracy = \frac{tp+tn}{tp+fp+fn+tn}$

For a multi class confusion matrix M with k classes, the performance parameters of i^{th} class is the summation of rows/columns of the matrix M, and is given as:

$$Recall_i = \frac{M_{ii}}{\sum_j^k M_{ji}} \tag{4.1}$$

$$Precision_i = \frac{M_{ii}}{\sum_j^k M_{ij}}$$

$$F - Measure_i = \frac{2 * Recall_i * Precision_i}{Recall_i + Precision_i}$$

$$Accuracy = \frac{\sum_i^k M_{ii}}{\sum_i^k \sum_j^k M_{ij}}$$

- **Processing efficiency:** It is defined as the number of features processed per second. It is calculated by dividing the total number of features by the total time taken for processing by the system (Hadoop cluster or conventional system).

4.4 Experimental setup

The proposed algorithms in the subsequent chapters are executed using MapReduce and Spark on the top of Hadoop cluster consisting of one master node and three slave nodes. Four commodity PCs connected with 10/100M switch are used in the experiment, and the configuration is as follows:

- Hardware configurations:
  - Master node: Name Node 1, CPU Intel core i5, 3.2 GHz x 4, RAM 14 GB, Hard disk 250 GB.

  - Slave node 1: Data Node 1, CPU Intel core i7, 3.4 GHz x 8, RAM 16 GB, Hard disk 500 GB
  - Slave node 2: Data Node 1, CPU Intel core i7, 3.4 GHz x 8, RAM 16 GB, Hard disk 500 GB
  - Slave node 3: Data Node 1, CPU Intel core i5, 3.2 GHz x 4, RAM 16 GB, Hard disk 250 GB

- Software used: Ubuntu 14.04, JDK, Hadoop 2.6.0, Python 2.7, Spark 1.5.0

4.5 Implementation

In this section, the implementation details of the proposed scalable statistical tests based on MapReduce and Spark are discussed. The input to the algorithm is a matrix of the form $N \times M$, where N is the total number of features and M is the number of samples in the dataset. As discussed earlier, the execution of the algorithms on MapReduce are divided into two parts, the map phase and the reduce phase. In map phase, each mapper running on a Datanode, reads a line (feature f_i) from the block and calculates the required test statistic (s_i) and p-value along with the feature ID (i) as a key-value pair ($\langle i, (s_i, p_i) \rangle$). It emits this pair into a intermediary file. The reducer, based on the p-value decides on whether to select or discard a feature. It then emits out the selected feature IDs ($\langle (fs_1, fs_2, fs_3, ...) \rangle$).

Similarly, the execution of algorithms on Spark are described as follows: The driver program, which is client side API and creates the Spark context. The Spark context talks to Spark driver and cluster manager and launch the Spark executers. The executers executes on the worker nodes. The input dataset is transformed into a RDD with the help of Spark context, which is resilient, and distributed dataset. The mapper and reducer is applied on the RDD and the processing is done, i.e., the mapper is applied on the RDD and it returns the transformed RDD of key-value pair ($\langle i, (s_i, p_i) \rangle$) by applying *map* transformation. In this way, various types of transformation is applied on the RDD to get the desired result, and finally some feasible action (e.g., collect, saveAsTextFile, first, count, etc.) is performed on the transformed RDD to evaluate and get the result.

4.5.1 Feature selection using scalable ANOVA (sANOVA)

Analysis of Variance (ANOVA) is applied to compare the 'multiple mean' values of the dataset, and visualize whether there exists any significant difference between multiple sample means. The statistic for ANOVA is called the F-statistic, which can be calculated using following steps:

1. The variation between the groups is calculated as:

$$Between\ sum\ of\ squares\ (BSS) = m_1\left(\bar{X}_1 - \bar{X}\right)^2 + m_2\left(\bar{X}_2 - \bar{X}\right)^2 + ... + m_k\left(\bar{X}_k - \bar{X}\right)^2 \quad (4.2)$$

$$Between\ mean\ squares\ (BMS) = BSS/df \quad (4.3)$$

2. The variation within the groups is calculated as:

$$Within\ sum\ of\ squares\ (WSS) = (m_1 - 1)\,\sigma_1^2 + (m_2 - 1)\,\sigma_2^2 + ... + (m_k - 1)\,\sigma_k^2 \quad (4.4)$$

$$Within\ mean\ squares(WMS) = WSS/df_w \quad (4.5)$$

where, $\bar{X}$ is the mean of the X, df is degree of freedom ,and equal to $M - 1$, $df_w = (M - k)$, σ= standard deviation M = Number of samples, k = Number of groups, and m_k = no. of samples in group k.

3. F-test statistic is calculated as:

$$F = BMS/WMS \quad (4.6)$$

Algorithms 6 and 7 describe the implementation of ANOVA based on MapReduce and Spark frameworks.

4.5.2 Feature selection using scalable Kruskal-Wallis test (sKruskal-Wallis)

Kruskal-Wallis test is a non-parametric approach, which tests the Null hypothesis. This implies that the mean ranks of the g different groups (classes) are the same. If the distributions are different, the Kruskal-Wallis test can reject the null hypothesis even though the medians are the same. For that, the data vectors are transformed into ranks (r) in increasing order ranging from $r = 1$ to $r = M$. In the presence of sequences of equal values (i.e., ties), mean ranks are assigned to the corresponding sequences. The test statistic can be computed according to Equation 4.7.

$$H = \frac{12}{M(M+1)}\left(\sum_{k=1}^{g}\frac{(R_k^2)}{m_k}\right) - 3(M+1) \quad (4.7)$$

Algorithm 6 MapReduce based ANOVA

Input: Let $Fd = \{(f_i)|f_i \in \mathbf{R}^M, i = 1, 2, ..., N\}$ is a set of input data, where N is number of features and M is number of samples.
Output: Top P features, i.e., $SelectedFeatures$.

procedure MAP_A(f_i) ▷ $i = 1, 2, ..., N$
  for each feature f_i **do** ▷ runs in parallel
    Calculate the value of BMS using Equation 4.3.
    Calculate the value of WMS using Equation 4.5.
    Calculate the F-value ($F_i = BMS/WMS$)
    Calculate the p-value (p_i) corresponding to each F-value using F-distribution curve
    Emit $\langle i, (F_i, p_i)\rangle$ ▷ *return*
  end for
end procedure
procedure REDUCE_A($\langle i, (F_i, p_i)\rangle$)
  for each feature f_i **do** ▷ runs in parallel
    if $p_i < 0.001$ **then**
      Select the feature, called fs_i
    else
      Discard the feature
    end if
  end for
  Emit $\langle (fs_1, fs_2, fs_3, ...)\rangle$ ▷ *return*
end procedure

Algorithm 7 Spark based ANOVA

Input: Let $Fd = \{(f_i)|f_i \in \mathbf{R}^M, i = 1, 2, ..., N\}$ is a set of input data, where N is number of features and M is number of samples.
Output: Top P features, i.e., $SelectedFeatures$.

procedure DRIVER_MAIN(Fd)
  Create RDD of Input data Fd, called $featureRDD$
  $SelectedFeatures \leftarrow featureRDD$.*map*(MAP_A).*filter*(REDUCE_A).*saveAsTextFile*(``$\langle HDFSPath\rangle$/result")
  Return $SelectedFeatures$
end procedure

where, R_k = the rank sum of k^{th} group, and $M = \sum_{k=1}^{g} m_k$ is the total number of sample size. Algorithms 8 and 9 describe the implementation of Kruskal-Wallis based on MapReduce and Spark frameworks.

4.5.3 Feature selection using scalable Friedman test (sFriedman)

Friedman test is a non-parametric alternative for this type of g dependent groups with equal sample sizes. The null hypothesis, $H0 : F(1) = F(2) = ... = F(g)$ is tested against the alternative hypothesis: at least one group does not belong to the same population. The data

Algorithm 8 MapReduce based Kruskal-Wallis test

Input: Let $Fd = \{(f_i)|f_i \in \mathbf{R}^M, i = 1, 2, ..., N\}$ is a set of input data, where N is number of features and M is number of samples.
Output: Top P features, i.e., $SelectedFeatures$.

procedure MAP_KW(f_i) ▷ $i = 1, 2, ..., N$
  for each feature f_i **do** ▷ runs in parallel
    Assign a rank r_j to each feature value ▷ $i = 1, 2, ..., N$ and $j = 1, 2, ..., M$
    Calculate $R_g = \sum_{f_{ij} \in g} r_j$ for each group (class) g
    Calculate the H-value using Equation 4.7
    Calculate the p-value (p_i) corresponding to each H-value using χ^2-distribution curve
    Emit $\langle i, (H_i, p_i) \rangle$ ▷ *return*
  end for
end procedure
procedure REDUCE_KW($\langle i, (F_i, p_i) \rangle$)
  for each feature f_i **do** ▷ runs in parallel
    if $p_i < 0.001$ **then**
      Select the feature, called fs_i
    else
      Discard the feature
    end if
  end for
  Emit $\langle (fs_1, fs_2, fs_3, ...) \rangle$ ▷ *return*
end procedure

Algorithm 9 Spark based Kruskal-Wallis test

Input: Let $Fd = \{(f_i)|f_i \in \mathbf{R}^M, i = 1, 2, ..., N\}$ is a set of input data, where N is number of features and M is number of samples.
Output: Top P features, i.e., $SelectedFeatures$.

procedure DRIVER_MAIN(Fd)
  Create RDD of Input data Fd, called $featureRDD$
  $SelectedFeatures \leftarrow featureRDD$.*map*(MAP_KW).*filter*(REDUCE_KW).*saveAsTextFile*(“$\langle HDFSPath \rangle$/result”)
  Return $SelectedFeatures$
end procedure

vector is ranked (r) in ascending order ranging from $r = 1$ to $r = m$ separately for each group (class), where m is the number of samples in each group. After that, the statistic of Friedman test is calculated according to Equation

$$F = \frac{12}{Mg(g+1)} \left(\sum_{k=1}^{g} R_k^2 \right) - 3M(g+1) \tag{4.8}$$

where, R_k = the rank sum of k^{th} group and $N = \sum_{k=1}^{g} m_k$ is the total number of sample size, and g is the total number of groups (classes). Algorithms 10 and 11 describe the

implementation of Friedman based on MapReduce and Spark frameworks.

Algorithm 10 MapReduce based Friedman test

Input: Let $Fd = \{(f_i)|f_i \in \mathbf{R}^M, i = 1, 2, ..., N\}$ is a set of input data, where N is number of features and M is number of samples.
Output: Top P features, i.e., $SelectedFeatures$.

procedure MAP_F(f_i) ▷ $i = 1, 2, ..., N$
  for each feature f_i **do** ▷ runs in parallel
    Divide the feature value into their various group g (Let the number of samples in each group be m) ▷ $i = 1, 2, ..., N$
    Assign a rank r_j to each feature value of each group g separately, where $j = 1, 2, ..., m$
    Calculate $R_g = \sum_{f_{ij} \in g} r_j$ for each group g
    Calculate the F-value using Equation 4.8
    Calculate the p-value (p_i)corresponding to each F-value using χ^2-distribution curve
    Emit $\langle (F_i, p_i) \rangle$
  end for
end procedure
procedure REDUCE_F($\langle i, (F_i, p_i) \rangle$)
  for each feature f_i **do** ▷ runs in parallel
    if $p_i < 0.001$ **then**
      Select the feature, called fs_i
    else
      Discard the feature
    end if
  end for
  Emit $\langle (fs_1, fs_2, fs_3, ...) \rangle$ ▷ *return*
end procedure

Algorithm 11 Spark based Friedman test

Input: Let $Fd = \{(f_i)|f_i \in \mathbf{R}^M, i = 1, 2, ..., N\}$ is a set of input data, where N is number of features and M is number of samples.
Output: Top P features, i.e., $SelectedFeatures$.

procedure DRIVER_MAIN(Fd)
  Create RDD of Input data Fd, called $featureRDD$
  $SelectedFeatures \leftarrow featureRDD$.*map*(MAP_F).*filter*(REDUCE_F).*saveAsTextFile*(“$\langle HDFSPath \rangle$/result”)
  Return $SelectedFeatures$
end procedure

4.6 Results and Interpretation

In this section, the obtained results are discussed for the proposed algorithms (Section 3.2) on various microarray datasets.

4.6.1 Analysis of feature selection methods

Since the dataset contains a large number of features with irrelevant information, it leads to the ‘‘curse of dimensionality problem”. To avoid it, feature selection method based on statistical tests viz., ANOVA test, Kruskal-Wallis test, and Friedman test are applied. These methods are applied separately on each feature of the microarray data, assuming that there is no interaction between the classes (or groups). The statistical tests consider two hypothesis, i.e., Null hypothesis and alternate hypothesis. The Null hypothesis assumes that the properties (like mean, median, and variance, etc.) of the classes are same, i.e., there is no significant difference between them; and the alternate hypothesis is that, there exists a significant difference between the groups (or classes). The features confirming Null hypothesis (H_0) implies that they do not affect the classification result and hence, can be discarded. On the contrary, the alternate hypothesis (H_1) implies that the features have significant difference between their properties. Hence, they are accepted. The statistical tests have been applied to each feature and the corresponding p-value is a measure of how effective it is at separating groups.

By considering the 99.9% of the confidence interval (CI), if the p-value is less than 0.001, the null hypothesis is rejected and alternate hypothesis is accepted. Sorting these features according to their p-values helps to identify the features with strong representation.

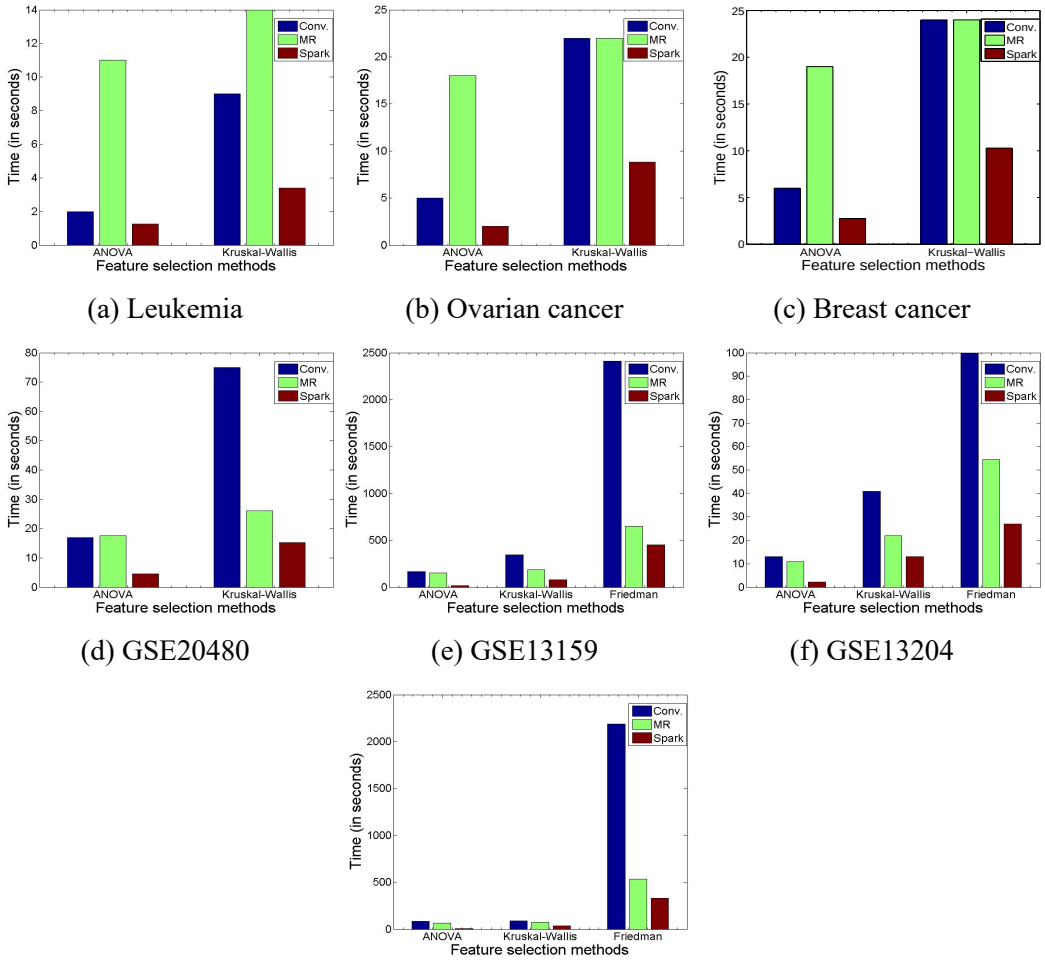
The proposed feature selection techniques have two variants, i.e., MapReduce and Spark. The implemented algorithms are executed on the top of Hadoop cluster and conventional system and performance is investigated. The execution details include the block size, number of mappers, minimum time taken by each mapper (T_{min}^{Map}) and reducer ($T_{min}^{Reducer}$), and total maximum time taken by the Hadoop cluster using MapReduce (MR) (T_{max}^{MR}), total time taken by Hadoop cluster using Spark (T_{max}^{Spark}), and conventional time (T_{max}^{Conv}). Conventional time (T_{max}^{Conv}) is defined as the time taken by a conventional system where data is not distributed over different machines, i.e., data is stored on a single machine, and the dataset is sequentially processed. The overall execution details of the system are tabulated in Table 4.10.

The comparison between the time taken by the Hadoop cluster (MR as well as Spark) and a conventional system is given in Figure 4.1. From this figure, it is inferred that when data size is small, the time taken by the MR is more than the time taken by a conventional system. But, as the size of data grows, the time taken by the MR is much less than that for the conventional system. But Spark is more faster than both MR and conventional system. Hence, we can say that, the execution of algorithms on Spark is faster than MapReduce and convention system, i.e., ($T_{max}^{Spark} \leq T_{max}^{MR}, T_{max}^{conv}$).

In this work, three different feature selection techniques such as ANOVA, Kruskal-Wallis, and Friedman have been considered to select right set of features over seven datasets such as Leukemia, Ovarian, Breast, GSE24080, GSE13159, GSE13204, and GSE15061; and their performance are compared using execution time (in sec).

Table 4.10: Execution details of various feature selection methods on Hadoop cluster (MR and Spark) and conventional system (Time is measured in seconds (s)).

Dataset	Feature selection method	Block size (MB)	No. of mapper/ reducer	T_{min}^{Map}	$T_{min}^{Reducer}$	T_{max}^{Conv}	T_{max}^{MR}	T_{max}^{Spark}	Processing efficiency Conv (s^{-1})	Processing efficiency MR (s^{-1})	Processing efficiency Spark (s^{-1})
Leukemia	**ANOVA**	2	2	2	4	2	11	1.26	117.5	21.34	187.25
	Kruskal-Wallis	2	2	5	4	9	14	3.41	30	19.29	79.18
Ovarian	**ANOVA**	2	2	4	3	5	18	2.02	64.8	18	160.79
	Kruskal-Wallis	2	2	12	3	22	22	8.83	14.73	14.73	36.71
Breast	**ANOVA**	2	2	4	1	6	19	2.75	809.83	255.74	1766.91
	Kruskal-Wallis	2	2	13	4	24	24	10.29	191.29	191.29	446.16
GSE24080	**ANOVA**	2	120	3	4	17	17.5	4.57	1152	1119.09	4285.34
	Kruskal-Wallis	2	120	4	4	75	26	15.22	273.36	788.54	1347.13
GSE13159	**ANOVA**	2	292	2	128.9	165	150	16.28	224.34	246.77	2273.71
	Kruskal-Wallis	2	292	3	158	347	185	81.48	106.33	199.44	452.84
	Friedman	2	292	10	398.7	2412	651.99	450	7.3	26.98	39.1
GSE13204	**ANOVA**	2	13	2	3.5	13	11	2.27	109.46	129.37	626.87
	Kruskal-Wallis	2	13	3	3.35	41	21.88	13	34.71	109.46	65.04
	Friedman	2	13	10	10.5	100	54.53	27	12.25	22.47	45.37
GSE15061	**ANOVA**	2	122	2	50.5	85	66	5.05	79.84	102.82	1343.76
	Kruskal-Wallis	2	122	4	60.324	89	76	38.13	109.4	128.11	255.34
	Friedman	2	122	21	276	2190	532.65	330	2.48	10.2	16.46



(a) Leukemia (b) Ovarian cancer (c) Breast cancer

(d) GSE20480 (e) GSE13159 (f) GSE13204

(g) GES15061

Figure 4.1: Comparison of execution time on Hadoop cluster (MR and Spark) and Conventional system (Conv) of various feature selection methods using different microarray datasets.

We have also considered three different platforms to execute this experiment. So for each FS techniques (ANOVA, Kruskal-Wallis), a total number of one set is used, each with 21 (7 dataset * 3 platforms). But, in case of Friedman, each set has 9 (3 dataset * 3 platforms) data points. The results of comparison analysis for performance parameters are summarized in Table 4.11.

Table 4.11 contains two sub tables. The first table shows the mean difference of execution time between the FS methods, and second table shows the mean difference of execution time on various platforms like Conv., MapReduce, and Spark. From Table 4.11a, we observe that ANOVA takes minimum time compared to other approaches. From Table 4.11b, we also observed that Spark takes minimum time to process the datasets as compare to other platforms.

Table 4.11: Performance comparison of various features selection methods

(a) Comaparison of execution time with various FS methods

	ANOVA	Kruskal-Wallis	Friedman
ANOVA	0	-25.0733	-720.287
Kruskal-Wallis	25.07333	0	-695.214
Friedman	720.2871	695.2138	0

(b) Comaparison of execution time on various platforms

	Conv	MR	Spark
Conv	0	217.7324	270.0259
MR	-217.732	0	52.29353
Spark	-270.026	-52.2935	0

Hence, from the above experiment it is observed that the total execution time (average) of these proposed methods on Spark is reduced by approximately 81.94% and 46.78% than the conventional system and MapReduce respectively; and the execution time on MapReduce is reduced by 66.06% than conventional system.

Table 4.12 shows the number of selected features, which have strong discriminating capacity to distinguish the samples into different classes, obtained by applying the various scalable statistical test (or feature selection methods). In GSE13159 dataset, with eighteen (18) classes and having 54675 features, the number of features selected are 37016, 36897, and 17593 after applying the feature selection methods like, ANOVA, Kruskal-Wallis, and Friedman test respectively. Similarly, the selected features for different datasets using various feature selection methods are shown in the Table 4.12.

4.7 Summary

In this chapter, various types of statistical tests based on MapReduce and Spark frameworks are implemented to select the relevant features from the microarray high-dimensional data. The performance of these techniques are investigated on the top of Hadoop cluster and compared with the conventional system.

The selected features are used to build an efficient model to classify the microarray data, which are discussed in the subsequent chapters.

Table 4.12: Number of selected relevant features

Dataset	ANOVA	Kruskal-Wallis	Friedman
Leukemia	235	270	—
Breast	324	324	—
Ovarian	4859	4591	—
GSE24080	83	75	—
GSE13159	37016	36897	17593
GSE13204	1423	1423	1225
GSE15061	6786	9736	5431

Chapter 5

Classification of Microarray Data using Scalable Proximal Support Vector Machine Classifier

---

In this chapter, a scalable proximal support vector machine (sPSVM) classifier is proposed. The proposed classifier works in a distributed manner and fit into any scalable cluster. The scalability of the proposed classifier increases as the number of nodes in the cluster increases. To test the scalability of the classifier two distributed and scalable frameworks viz., MapReduce and Spark on the top of Hadoop cluster are used. The performance of this proposed classifier is investigated using different dimensions of microarray data. The performance of sPSVM has been measured using the performance parameters like accuracy, precision, recall, and processing efficiency.

5.1 Introduction

After successful implementation of feature selection methods in Chapter 4, this chapter emphasizes on a classifier, which is able to classify the microarray dataset in a scalable manner.

As the size of the data samples increases, the training time increases and also the computational complexity increases in case of Support Vector Machine (SVM) [145]. In order to overcome the drawbacks of SVM, proximal support vector machine (PSVM) was developed. It is based on Least square support vector machine (LS-SVM) [146, 147]. The training time required by PSVM is less as compared to large training time in case of standard SVM. PSVM assigns the classes to the data points by measuring its proximity from the two parallel hyperplanes and the data points are clustered around the two parallel hyperplanes; whereas SVM divides the space into two half spaces such that datapoints of different classes are separated. The hyperplanes are designed such that the margin of separation between the two classes is maximized. The idea behind proposed work is to maximize the margin between the hyperplanes or the decision surface such that data points lie on the correct side of

the hyperplanes in order to increase the generalization ability of the classifier or to minimize the generalization error.

In this chapter, proximal support vector machine (sPSVM) classifier based on MapReduce as well as Spark has been proposed to classify the microarray high-dimensional dataset. The proposed algorithm not only helps to process large datasets, but also can be extended to execute on a cluster. The scalability of the algorithm is tested by varying the data size of high-dimensional data on a Hadoop cluster. The classifier is implemented on two frameworks like MapReduce and Spark, and the performance of the algorithm is tested on Hadoop cluster with three slave (data) nodes and a conventional system.

The rest of the work is organized as follows: Section 5.2 presents the proposed work for classifying the microarray data proximal support vector machine based on MapReduce and Spark frameworks. Section 5.3 presents the implementation details for the proposed approach. Section 5.4 discusses on the results obtained, interpretation drawn from it and also presents the comparative analysis for classification of various microarray datasets. Section 5.5 summarizes the chapter with scope for future work.

5.2 Proposed work

This section presents an approach for classification of microarray data, which consists of three phases:

- The input data is preprocessed using methods such as missing data imputation, normalization and feature selection using statistical tests based on MapReduce and Spark frameworks.
- After selecting the relevant features, MapReduce as well as Spark based Proximal support vector machine (sPSVM) has been applied to classify microarray dataset into their respective classes, i.e., (cancerous/non-cancerous).

Figure 5.1 shows the graphical representation of proposed approach.
The brief note on steps for the proposed approach is given as follows:

a. **Data collection**

   The dataset for classification analysis, which acts as requisite input to the models is obtained from Kent Ridge Bio-medical Data Set Repository [93] and National center of Biotechnology Information (NCBI GEO, `http://www.ncbi.nlm.nih.gov/gds/`).

b. **Missing data imputation and normalization of dataset**

   Missing data of a feature (gene) in microarray dataset are imputed by using the *mean* value of the respective feature. Input feature values are normalized over the range $[0, 1]$ using Min-Max normalization technique [95, 126].

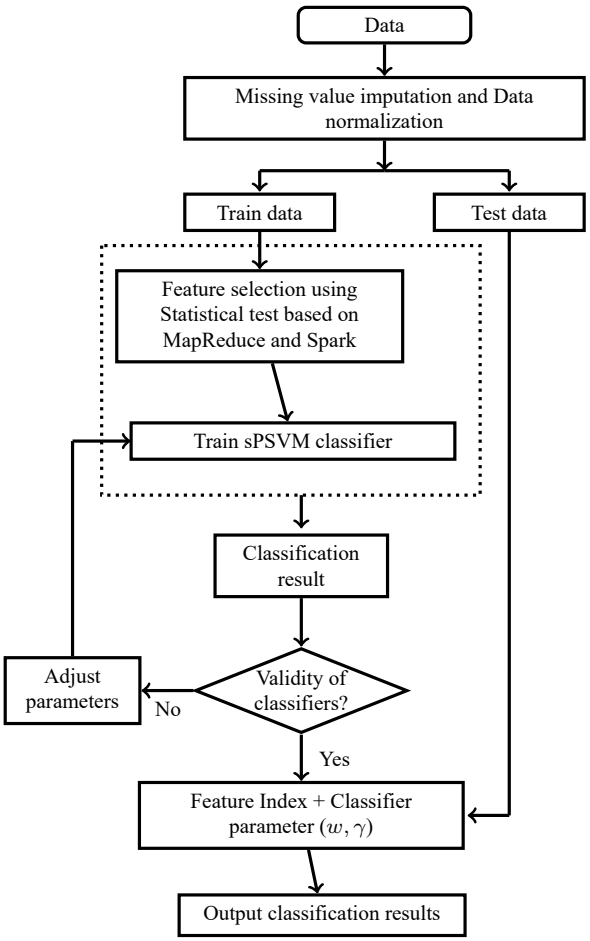


Figure 5.1: Proposed approach for microarray classification.

c. **Feature selection**

Statistical tests viz., ANOVA-test, Kruskal-Wallis test, and Friedman test based on MapReduce and Spark have been applied to select the features having high relevance value and thus the curse of dimensionality issue has been addressed.

d. **Division of dataset**

The dataset is divided into two categories: training set and testing set as discussed in Section 5.4.

e. **Building classifier**

Proximal support vector machine classifier based on MapReduce and Spark has been built to classify the microarray dataset.

f. **Testing the classifier**

Classifier is tested using the test dataset and the performance of the classifier is evaluated.

5.3 Implementation

In this section, the implementation of the proposed algorithms using MapReduce and Spark are discussed.

5.3.1 Scalable Implementation of Proximal Support Vector Machine (sPSVM) classifier

5.3.1.1 MapReduce based Proximal Support Vector Machine (mrPSVM) classifier

Let A be the matrix of M samples and N features in the space of $\mathbf{R}^N$ represented by $M \times N$ matrix, D is the class label, denoted as $+1$ or -1 for two class problem.

In multiclass problem (SVM or PSVM) the class label $D \in \mathbf{R}^{M \times k}$ is defined in the following way, where M is the number of samples and k is the number of classes [148].

$$D_i = \begin{cases} \left\{1, \frac{-1}{k-1}, ..., \frac{-1}{k-1}\right\}; & if\ sample\ i\ belongs\ to\ class\ 1 \\ \left\{\frac{-1}{k-1}, 1, ..., \frac{-1}{k-1}\right\}; & if\ sample\ i\ belongs\ to\ class\ 2 \\ \vdots & \\ \left\{\frac{-1}{k-1}, \frac{-1}{k-1}, ..., 1\right\}; & if\ sample\ i\ belongs\ to\ class\ k \end{cases}$$

To classify the M samples with Support vector machine (SVM) with linear kernel is given by the following quadratic problem [145].

$$\begin{aligned} \min_{w,\gamma,\xi} J(w,\gamma,\xi) &= Ce'\xi - \frac{1}{2}w'w \\ s.t.\quad D(Aw - e\gamma) + \xi &\geq e \\ e &\geq 0 \end{aligned} \tag{5.1}$$

where, C is the regularization factor, ξ is the error term, e is a vector filled with ones, w is the normal to the bounding planes:

$$\begin{aligned} x'w &= \gamma + 1 \\ x'w &= \gamma - 1 \end{aligned} \tag{5.2}$$

that constrained the most of the sets in to $A^+(+1)$ or $A^-(-1)$ respectively, and γ is the relative location of planes to the origin. It divides the space into two half spaces such that data points of different classes are separated. The plane acts as a linear classifier as follows:

$$sign(x'w - \gamma) \begin{cases} = 1, & then\ x \in +1 \\ = -1, & then\ x \in -1 \end{cases} \tag{5.3}$$

In SVM, as the size of the data samples increases, both the training time and the computational complexity of the algorithm increases. In order to overcome the drawbacks of

SVM, PSVM has been developed using the constraints of least-square (LS) SVM [146, 147]. The training time required by PSVM is less as compared to large training time in case of standard SVM. PSVM assigns the classes to the data points by measuring its proximity from the two parallel hyperplanes [149]. The hyperplanes are designed in such a manner that the margin of separation between the two classes is maximized [150]. The parameters of hyperplanes are obtained by optimization of the following quadratic function.

$$\begin{aligned} \underset{w,\gamma,\xi}{min}\ J(w,\gamma,\xi) = \frac{C}{2}\left\|\xi\right\|^2 - \frac{1}{2}\left(w'w+\gamma^2\right) \\ s.t.\quad D\left(Aw-e\gamma\right)+\xi = e \end{aligned} \tag{5.4}$$

In PSVM, the inequality constraint of LS-SVM is replaced with the equality constraint, which makes the computation cheaper. By substituting the value of ξ in Equation 5.4 from the constraint, we get

$$\underset{w,\gamma}{min}\ J(w,\gamma) = \frac{C}{2}\left\|D\left(Aw-e\gamma\right)-e\right\|^2 - \frac{1}{2}\left(w'w+\gamma^2\right) \tag{5.5}$$

To minimize the cost function J, the Equation 5.5 is partially differentiated with respect to $\begin{bmatrix} w \\ \gamma \end{bmatrix}$, and we get after differentiation

$$\begin{aligned} -CA'D(D(Aw-e\gamma)-e)+w = 0 \\ -Ce'D(D(Aw-e\gamma)-e)+\gamma = 0 \end{aligned} \tag{5.6}$$

This can be written as:

$$\begin{bmatrix} \left(AA'+\frac{I}{C}\right) & -A'e \\ e'A & \left(\frac{I}{C}+1\right) \end{bmatrix} \begin{bmatrix} w \\ \gamma \end{bmatrix} = \begin{bmatrix} A'De \\ -e'De \end{bmatrix} \tag{5.7}$$

$$\left[\frac{1}{C}+\begin{bmatrix} A' \\ -e' \end{bmatrix}\begin{bmatrix} A & -e \end{bmatrix}\right]\begin{bmatrix} w \\ \gamma \end{bmatrix} = \begin{bmatrix} A' \\ -e' \end{bmatrix} De \tag{5.8}$$

By substituting $E = \begin{bmatrix} A & -e \end{bmatrix}$,

$$\left[\frac{1}{C}+E'E\right]\begin{bmatrix} w \\ \gamma \end{bmatrix} = E'De \tag{5.9}$$

By solving this system of equations, the value of w and γ can be evaluated using the training dataset. After evaluating the value of $w \in \mathbf{R}^{N\times k}$ and $\gamma \in \mathbf{R}^{1\times k}$, a new test sample $x_t \in \mathbf{R}^{1\times N}$ can be classified as:

$$z = x_t * w - \gamma \tag{5.10}$$

where, $z \in \mathbf{R}^k$ is a matrix of dimension $1 \times k$ that provides the weight values of the class label of the test sample $x_t \in \mathbf{R}^{1 \times N}$. The class label is predicted as:

$$classLabel = \underset{1,2,...,k}{argmax} \{z\} \tag{5.11}$$

The term $E'E$ and $E'De$ $(E'eD)$ from Equation 5.9 can be expressed in the following forms:

$$E'E = E'_1E_1 + E'_2E_2 + ... + E'_ME_M \tag{5.12}$$

$$E'De = E'_1D_1e + E'_2D_2e + ... + E'_MD_Me \tag{5.13}$$

using the above expression, the term $E'E$ and $E'De$ can be parallelized, and calculated using MapReduce (or Spark) on Hadoop framework. Each individual term of the RHS (right hand side) of Equation 5.12 and 5.13 is calculated using the MAP_PSVMTrain() function and the result of each individuals are sent to the REDUCE_PSVMTrain() function. The Reducer function aggregates the results of each individual term and yields the final result. The respective algorithms 12 and 13 describe the training and testing of MapReduce based PSVM.

Algorithm 12 Training: MapReduce based Proximal Support Vector Machine (mrPSVM) classifier

Input: Let $X = \{(x_1, y_1)(x_2, y_2), ..., (x_i, y_i)\}$ is a set of M training samples with N attributes, where $i = 1, 2, ..., M$, i.e., $x \in \mathbf{R}^{M \times N}$ and $y \in \mathbf{R}^{M \times 1}$

Output: Calculation of w and γ

procedure MAP_PSVMTrain(X_i) ▷ $i = 1, 2, ..., M$
  for each sample X_i **do** ▷ runs in parallel
    Parse the label y and the value of each attributes.
    Form matrix $D_i \in \mathbf{R}^{1 \times k}$ from label y_i
    Form matrix $A_i \in \mathbf{R}^{1 \times N}$ from x_i consisting of the values of each attribute
    Form matrix $e \in \mathbf{1}^{1 \times 1}$
    Form matrix $E_i = \begin{bmatrix} A & -e \end{bmatrix} \in \mathbf{R}^{1 \times (N+1)}$
    Calculate $S = E'_i * E_i$ and $R = E'_ieD_i$
    Emit $\langle i, (S_i, R_i) \rangle$ ▷ *return*
  end for
end procedure
procedure REDUCE_PSVMTrain($\langle i, (S_i, R_i) \rangle$)
  Initialize C $\leftarrow$ 0.1
  Read a key-value pair from mapper
  Sum all the S-values and R-values to generate $E'E$ and $E'eD$
  Calculate w and γ by solving the system of linear equation 5.9
  Emit $\langle w, \gamma \rangle$ ▷ *return*
end procedure

Algorithm 13 Testing: MapReduce based Proximal Support Vector Machine (mrPSVM) classifier

Input: Test set $x_t \in \mathbf{R}^{p \times N}$, $w \in \mathbf{R}^{N \times k}$, and $\gamma \in \mathbf{R}^{1 \times k}$
Output: Calculation of $classLabel$ and $Accuracy$.

procedure MAP_PSVMTEST($testsample$) ▷ $i = 1, 2, ..., p$
  for each test sample **do** ▷ runs in parallel
    Calculate z using the Equation 5.10
    Calculate $classLabel$ using Equation 5.11
    Emit $\langle label, classLabel \rangle$ ▷ *return*
  end for
end procedure
procedure REDUCE_PSVMTEST($\langle label, classLabel \rangle$)
  Read a $key - value$ pair from mapper
  Initialize $counter = 0$
  for each $key - value$ pair **do**
    if $key == value$ **then**
      $counter = counter + 1$
    end if
  end for
  $Accuracy = counter/p$ ▷ p= number of instances in test data
  Emit $\langle classLabel, Accuracy \rangle$ ▷ *return*
end procedure

5.3.1.2 Spark based Proximal Support Vector Machine (sf-PSVM) classifier

The above discussed PSVM algorithm is implemented using Spark framework by making some modification. The training data and testing data are read from HDFS and transformed into RDD with the help of Spark context, called $trainRDD$ and $testRDD$ respectively. The PSVM is trained with $trainRDD$ and tested using $testRDD$ as described in Algorithm 14.

Algorithm 14 sf-PSVM: Spark based PSVM

Input: Let $X = \{(x_1, y_1)(x_2, y_2), ..., (x_i, y_i)\}$ is a set of M training samples with N attributes, where $i = 1, 2, ..., M$, i.e., $x \in \mathbf{R}^{M \times N}$ and $y \in \mathbf{R}^{M \times 1}$
Test set $x_t \in \mathbf{R}^{p \times N}$, $w \in \mathbf{R}^{N \times k}$, and $\gamma \in \mathbf{R}^{1 \times k}$
Output: Calculation of $classLabel$ and $Accuracy$.

procedure DRIVER_MAIN(X, x_t)
  Read train data X and test data x_t from HDFS
  Create RDD of train data and test data i.e., called $trainRDD$ and $testRDD$ respectively.
  $weight \leftarrow trainRDD$.*map*(MAP_PSVMTRAIN).*map*(REDUCE_PSVMTRAIN).*collect*()
  Broadcast $weight$ to all worker nodes
  $result \leftarrow testRDD$.*map*(MAP_PSVMTEST).*map*(REDUCE_PSVMTEST).*saveAsTextFile*("$\langle HDFSPath \rangle$/result")
end procedure

5.4 Results and interpretation

In this section, the obtained results for the proposed algorithms (Section 5.2) are analyzed while various microarray datasets, which are being given as input.

After feature selection, the proposed classification algorithm, i.e., MapReduce as well as Spark based proximal support vector machine (sPSVM) classifier has been applied to classify the datasets into the appropriate classes. However, unless one has some prior knowledge of the dataset, it is difficult to decide on the optimal number of features required for classification. To overcome this problem, forward feature selection method is considered, in which top ranked features corresponding to ascending p-values are used. Different subsets of the top ranked features are used to classify the microarray dataset using MapReduce as well as Spark based proximal support vector machine (sPSVM) classifier and their corresponding classification accuracies are computed.

The model is trained using training dataset and performance of model is measured using testing dataset. Various case studies using different datasets are carried out to explore the performance of the classifier on various cluster nodes, which have been discussed in the following subsections.

5.4.1 Result of Leukemia dataset

There are 72 samples in Leukemia dataset, out of which 48 samples are selected as training and 24 as testing samples. Out of 24 testing samples, 16 are in ALL (0) and 8 are in AML (1) class. Figure 5.2 shows the accuracy of the sPSVM classifier with various feature sets obtained from different feature selection methods. From Figure 5.2, it is clear that, when ANOVA is used as a FS method, where top 60 features are sufficient to achieve high accuracy of 100%; and when Kruskal-Wallis test is used, top 50 features are sufficient to achieve high accuracy. After attaining the peak accuracy, the accuracy of classifier either remains constant or decreases from the peak. Therefore, to avoid the curse of dimensionality problem, top selected features are used to analyze the microarray datasets using sPSVM classifier and the various performance parameters of the classifier are evaluated.

Figure 5.3 represents the confusion matrix obtained using leukemia dataset by sPSVM with different feature selection methods; which denotes all the performance parameters like accuracy, recall, precision, specificity, etc. From Figure 5.3a and Figure 5.3b, it is clear that accuracy, recall, precision, and specificity are 100%, 100%, 100%, and 100% respectively.

5.4.2 Result of Breast cancer dataset

There are 97 samples in breast cancer dataset, out of which 64 samples are selected as training and 33 as testing samples. Out of these 33 testing samples, 16 are relapse (0), and 17 are non-relapse (1). Figure 5.4 shows the accuracy of the sPSVM classifier with

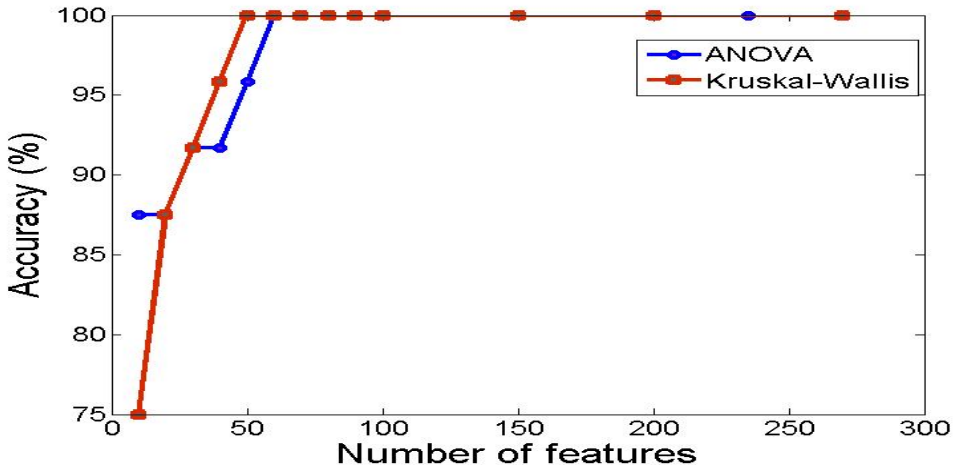
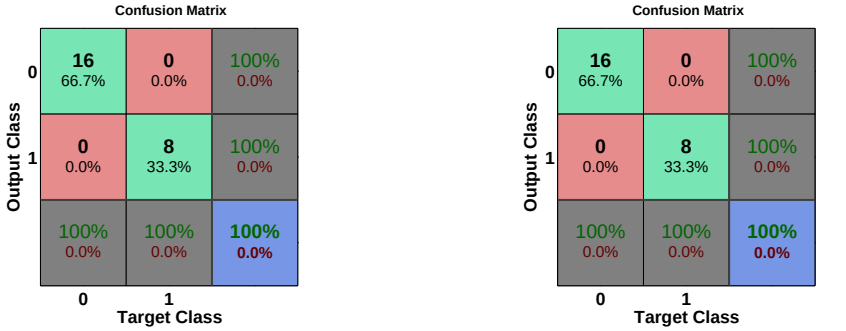


Figure 5.2: Testing accuracy sPSVM classifier with different set of features using Leukemia dataset



(a) ANOVA ($f = 60$) (b) Kruskal-Wallis ($f = 50$)

Figure 5.3: Confusion matrix of sPSVM classifier using Leukemia cancer dataset with various feature selection methods.

various feature sets obtained from different feature selection methods. From Figure 5.4, it is observed that, when ANOVA as feature selection is applied, top 30 features ($f = 30$) are sufficient to achieve the peak accuracy value (accuracy=72.72%) for the dataset (Breast cancer). The corresponding confusion matrix has been shown in Figure 5.5a. Similarly, Figure 5.5b represent the confusion matrix for dataset reduced by Kruskal-Wallis test. The obtained values of performance parameters like accuracy, recall, specificity, and precision using ANOVA, and Kruskal-Wallis, are 72.72%, 76.5%, 68.8% and 72.2% respectively at the respective optimal number of features f, as shown in Figure 5.5.

5.4.3 Result of Ovarian dataset

There are 253 samples in ovarian dataset, out of which 168 samples are selected as training and 85 as testing samples. Out of these 85 samples, 54 are cancerous (0), and 31 are normal (1) samples. Figure 5.6 shows the accuracy of the sPSVM classifier with various feature sets obtained from different feature selection methods. From Figure 5.6, the peak accuracy

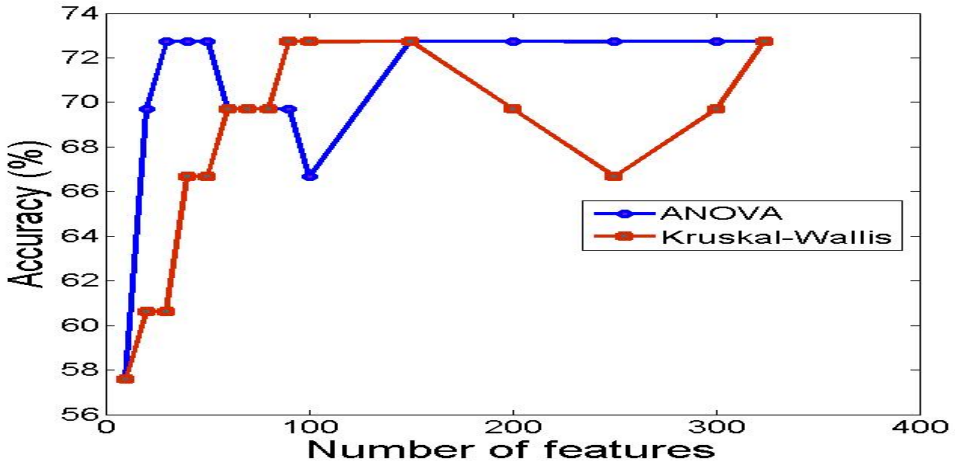
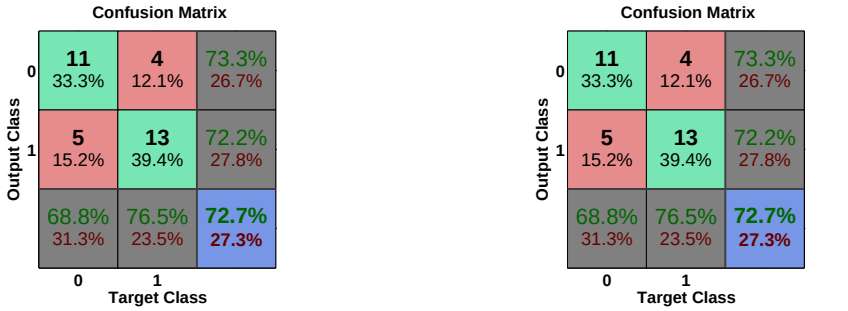


Figure 5.4: Testing accuracy sPSVM classifier with different set of features using Breast dataset



(a) ANOVA ($f = 30$) (b) Kruskal-Wallis ($f = 90$)

Figure 5.5: Confusion matrix of sPSVM classifier using Breast cancer dataset with various feature selection methods.

value for dataset reduced through ANOVA (accuracy=100%) has been obtained at $f = 600$ and the corresponding confusion matrix has been presented in Figure 5.7a. Similarly, Figure 5.7b represents the confusion matrix for dataset reduced by Kruskal-Wallis test and obtained accuracy is 100% at the optimal number of top selected features $f = 500$. The value of rest of the parameters like recall, specificity, and precision are 100%, 100%, and 100% respectively and as shown in Figure 5.7.

5.4.4 Result of GSE24080 dataset

There are 559 samples in GSE24080 dataset, out of which 372 samples are selected as training and 187 as testing samples. Out of these 187 samples 57 are labeled as '0' and 130 are labeled as '1'. Figure 5.8 shows the accuracy of the sPSVM classifier with various feature sets obtained from different feature selection methods. From Figure 5.8, it is evident that, the peak accuracy value for dataset reduced through ANOVA (accuracy=70.59%) has been obtained when the top 60 number of features (f) are used. The corresponding confusion matrix has been shown in Figure 5.9a.

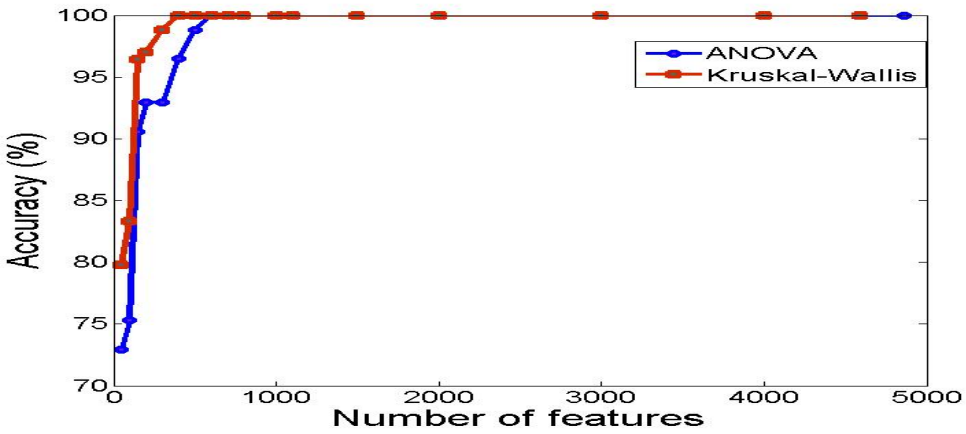
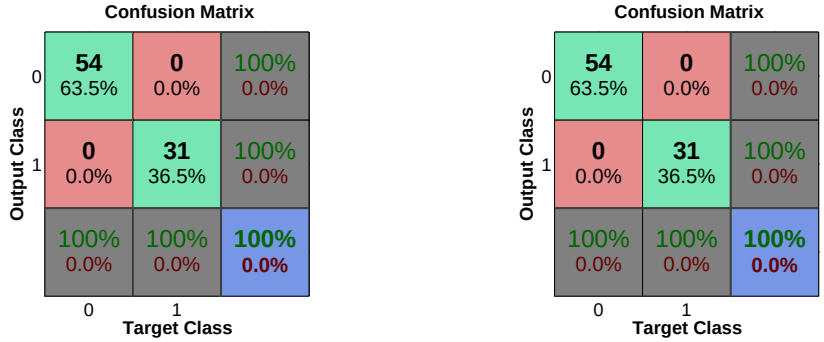


Figure 5.6: Testing accuracy sPSVM classifier with different set of features using Ovarian dataset



(a) ANOVA ($f = 600$) (b) Kruskal-Wallis ($f = 500$)

Figure 5.7: Confusion matrix of sPSVM classifier using Ovarian dataset with various feature selection methods.

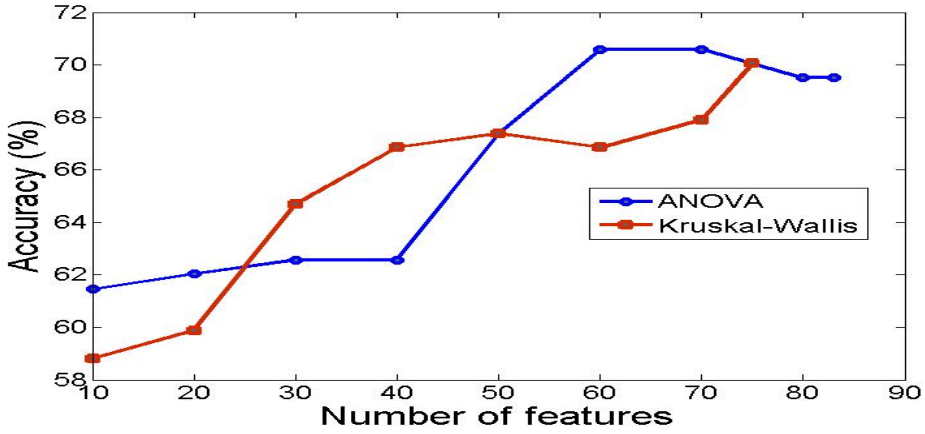
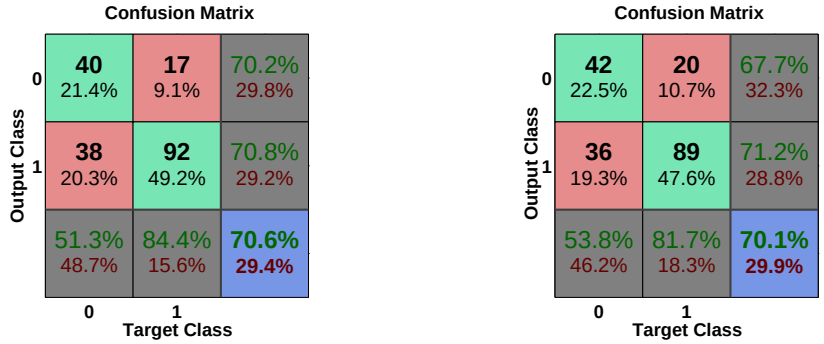


Figure 5.8: Testing accuracy of sPSVM classifier with different set of features using GSE24080 dataset

Figure 5.9b, represents the confusion matrix for dataset reduced by Kruskal-Wallis test and the obtained accuracy is 70.05% at top 75 features. The rest of the performance parameters like recall, specificity, and precision, etc. have been shown in Figure 5.9. The



(a) ANOVA ($f = 60$) (b) Kruskal-Wallis ($f = 75$)

Figure 5.9: Confusion matrix of sPSVM classifier using GSE24080 dataset with various feature selection methods.

respective value of parameters like recall, specificity, and precision are 84.4%, 51.3%, and 70.8%, when ANOVA is applied as a feature selection method with sPSVM classifier. Similarly, when Kruskal-Wallis feature selection method with sPSVM is applied, the value of performance parameters like recall, specificity, and precision are 81.7%, 53.8%, and 71.2%.

5.4.5 Result of GSE15061 dataset

This dataset is the subset of the MILE Study (Microarray Innovations In high-dimensional) program, stored in the NCBI repository with accession number GSE15061. There are 870 samples in GSE15061 dataset, out of which, 580 samples are selected as training and 290 as testing samples. These samples are divided into 3 classes and labeled as shown in Table 4.7.

Figure 5.10 shows the accuracy of the sPSVM classifier with various feature sets obtained from different feature selection methods. From Figure 5.10, it is inferred that the peak accuracy value for dataset reduced through ANOVA (accuracy=84.14%) has been obtained when top 5,000 features (f) are used and the corresponding confusion matrix has been tabulated in Figure 5.11a. Similarly, Figure 5.11b, and Figure 5.11c represent the confusion matrix for dataset reduced by Kruskal-Wallis, and Friedman test respectively. The obtained accuracies of sPSVM classifier using ANOVA, Kruskal-Wallis, and Friedman test are 84.14%, 83.45%, and 83.45%, respectively at the top selected features f. The respective values of precision and recall of each class have been shown in the last column and row of the confusion matrix.

5.4.6 Result of GSE13159 dataset

There are 2096 samples in GSE13159 dataset, out of which, 1397 samples are selected as training and 699 as testing samples. These samples are divided into 18 classes and labeled as shown in Table 4.5.

Figure 5.12 shows the accuracy of the sPSVM classifier with various feature sets obtained from different feature selection methods. From Figure 5.12, it is observed the peak accuracy

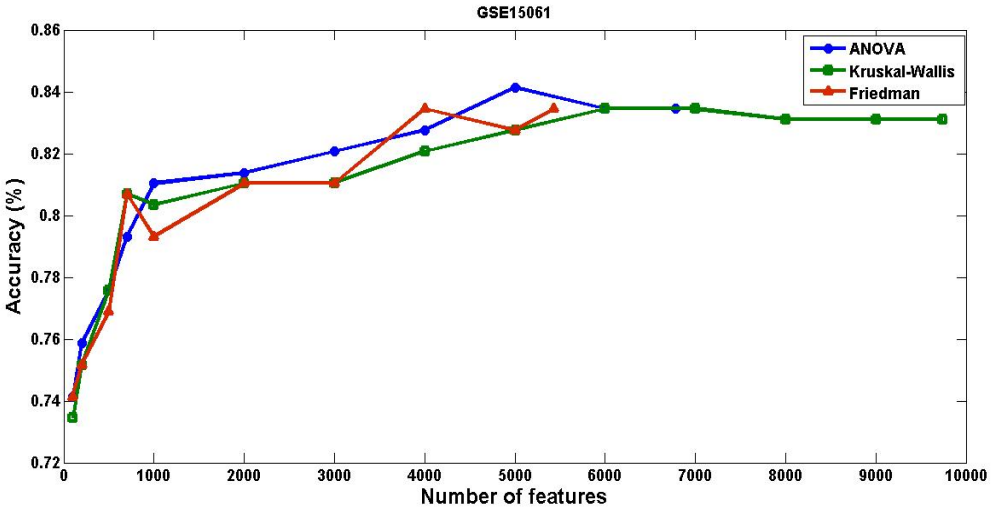
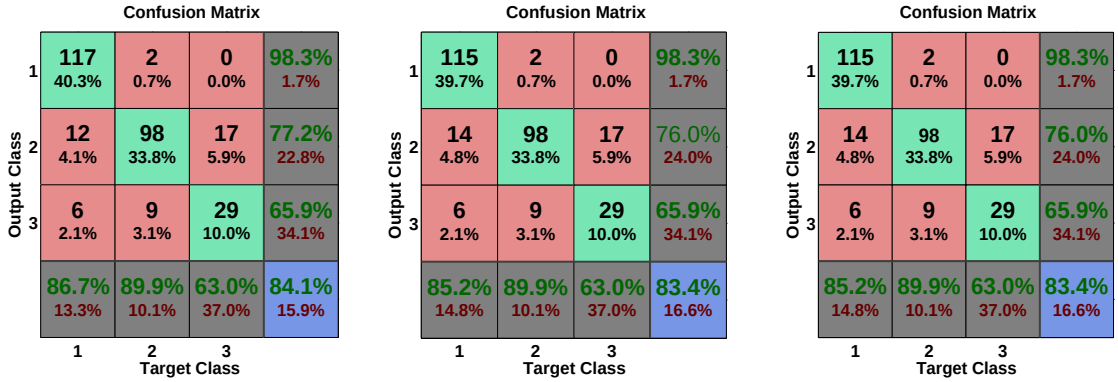


Figure 5.10: Testing accuracy sPSVM classifier with different set of features using GSE15061 dataset



(a) ANOVA ($f = 5000$) (b) Kruskal-Wallis ($f = 6000$) (c) Friedman ($f = 4000$)

Figure 5.11: Confusion matrix of sPSVM classifier using GSE15061 dataset with various feature selection methods.

value for dataset reduced through ANOVA (accuracy=81.11%) has been obtained when top 14,000 features (f) are used and the corresponding confusion matrix has been tabulated in Figure 5.13. Similarly, Figure 5.14, and Figure 5.15 represent the confusion matrix for dataset reduced by Kruskal-Wallis, and Friedman test respectively. The obtained accuracies of sPSVM classifier using ANOVA, Kruskal-Wallis, and Friedman test are 81.11%, 81%, and 80.40%, respectively at the top selected features f. The respective values of precision and recall of each class have been shown in the last column and row of the confusion matrix.

Table 5.1 and Table 5.2 represent the execution details, i.e., number of mappers and reducers, time taken by each mapper and reducer, and total time taken of the sPSVM classifier on the Hadoop cluster with three slave nodes in the training and testing phase respectively in seconds (s).

Table 5.3 shows the total time elapsed by the Hadoop cluster and conventional machine and their processing efficiency. The processing efficiency describes the capability of the

Table 5.1: Execution details of sPSVM classifier on Hadoop cluster (MapReduce and Spark) in Training phase (Time is measured in seconds (s)).

Dataset	Feature selection methods	Block Size	No. of MR	Time (each mapper)	Time (each Reducer)	Total MR time (training)	Spark time (training)
Leukemia	**ANOVA**	16	2/1	4.80	5.5	48.37	30
	Kruskal-Wallis	16	2/1	5.80	2.2	51.43	33
Breast Cancer	**ANOVA**	16	2/1	4.30	2.1	43.1	25
	Kruskal-Wallis	16	2/1	5.30	3.7	48.76	31
Ovarian	**ANOVA**	4	2/1	143.20	112.8	193	123
	Kruskal-Wallis	4	2/1	41.50	20.4	82.2	64
GSE24080	**ANOVA**	16	2/1	9.10	6.4	51.69	36
	Kruskal-Wallis	16	2/1	6.60	6.8	55.55	37
GSE15061	**ANOVA**	2	16/1	4.00	22	32	19
	Kruskal-Wallis	2	22/1	4.00	28	55	34.5
	Friedman	2	12/1	3.00	31	65	40
GSE13159	**ANOVA**	16	29/1	4.00	6	124	105
	Kruskal-Wallis	16	29/1	3.00	113	134	111
	Friedman	16	17/1	4.00	6	131	107

Table 5.2: Execution details of sPSVM classifier on Hadoop cluster (MapReduce and Spark) in Testing phase (Time is measured in seconds (s)).

Dataset	Feature selection methods	Block Size	No. of MR	Time (each mapper)	Time (each Reducer)	Total MR time	Spark time
Leukemia	**ANOVA**	16	2/1	5.00	2	16	9
	Kruskal-Wallis	16	2/1	4.00	1.9	15.9	11
Breast Cancer	**ANOVA**	16	2/1	4.20	1.9	15.9	11
	Kruskal-Wallis	16	2/1	4.60	2.7	21.21	14.5
Ovarian	**ANOVA**	4	2/1	6.00	2.4	17.8	12.3
	Kruskal-Wallis	4	2/1	8.70	2.1	15.8	10.6
GSE24080	**ANOVA**	16	2/1	4.20	2	16.9	11.2
	Kruskal-Wallis	16	2/1	5.90	2.4	22.12	15
GSE15061	**ANOVA**	2	8/1	4.00	30	44	30
	Kruskal-Wallis	2	11/1	5.00	45	62	45
	Friedman	2	7/1	4.00	42	58	41
GSE13159	**ANOVA**	16	15/1	70.00	53	167	105
	Kruskal-Wallis	16	15/1	72.00	74	162	101
	Friedman	16	9/1	54.00	53	84	65

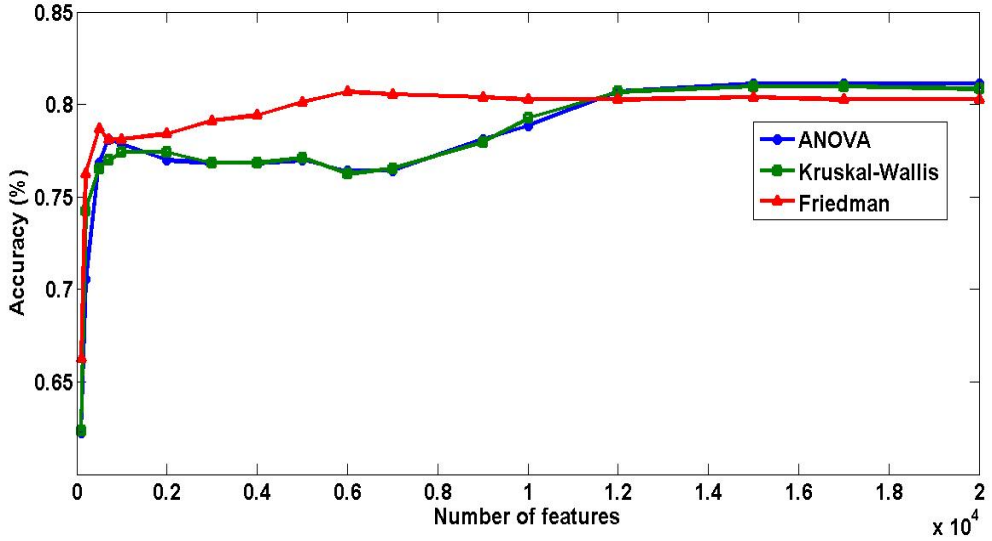


Figure 5.12: Testing accuracy sPSVM classifier with different set of features using GSE13159 dataset

classifier to process the data on the machine.

From the obtained result, it is inferred that, when the data size is small, Hadoop cluster takes more time than the conventional system to complete the job, but as the size of data increases, Hadoop takes very less time than a conventional system. The comparison of time taken by sPSVM classifier on Hadoop cluster and a conventional system has been shown in Figure 5.16. From Figure 5.16, it is clear that the time taken by sPSVM classifier to analyze the datasets of Leukemia, Ovarian, Breast cancer, and GSE24080 (data size is small) on Hadoop cluster is more than conventional system. But, in case of GSE15061 and GSE13159 (data size is large), time taken by sPSVM classifier on Hadoop cluster is very less than a conventional system (Conv.).

Table 5.3: Timing details (in seconds) and processing efficiency of the sPSVM classifier (Training + Testing)

Dataset	Feature selection methods	Total time (Conv)	Total time (MR)	Total Time (Spark)	Processing Efficiency (Conv)	Processing Efficiency (MR)	Processing Efficiency (Spark)
Leukemia	**ANOVA**	0.493	64.37	39	121.70	0.932	1.54
	Kruskal-Wallis	0.46	67.33	44	108.70	0.74	1.14
Breast Cancer	**ANOVA**	13.326	68.59	36	2.25	0.44	0.84
	Kruskal-Wallis	14.772	77.67	45.5	6.09	1.16	1.98
Ovarian	**ANOVA**	109.78	210.8	135.3	5.57	2.85	4.44
	Kruskal-Wallis	97.396	98	74.6	5.13	5.1	6.71
GSE24080	**ANOVA**	1.78	59	47.2	33.71	1.02	1.27
	Kruskal-Wallis	2.54	69.97	52	29.53	1.07	1.44
GSE15061	**ANOVA**	955	76	49	5.24	65.79	102.04
	Kruskal-Wallis	1186	97	79.5	5.06	61.86	75.47
	Friedman	745	123	81	5.37	32.52	49.38
GSE13159	**ANOVA**	1265.102	291	210	11.06	48.12	66.67
	Kruskal-Wallis	1516.17	296	212	9.89	50.68	70.76
	Friedman	669.75	215	172	8.96	27.91	34.88

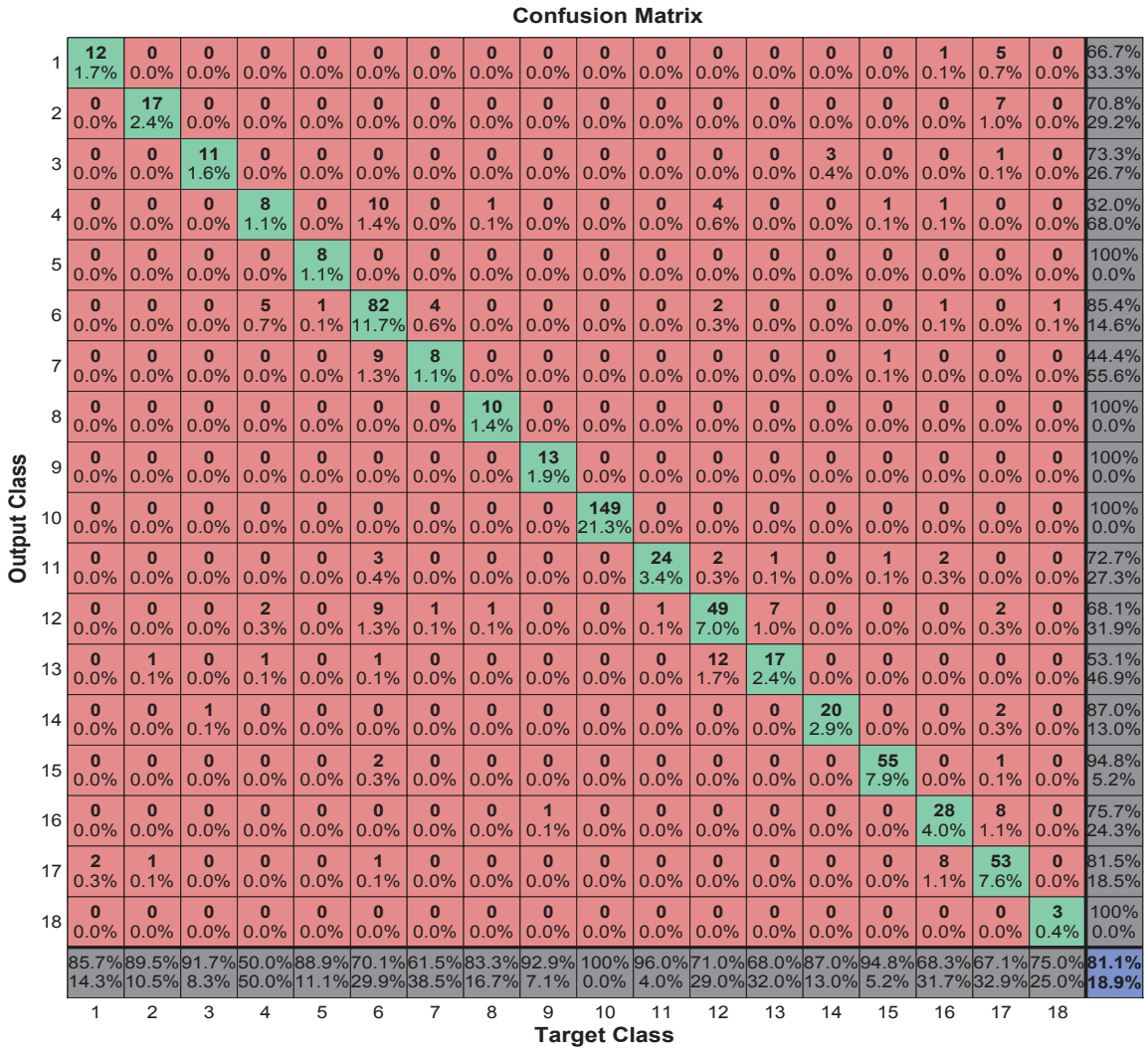


Figure 5.13: Confusion matrix for sPSVM classifier with ANOVA using GSE13159 dataset ($f = 14,000$)

5.4.7 Comparative analysis

In this section, emphasis has been laid on designing feature selection models based on statistical tests and a classifier which can obtain comparative result for classification of microarray datasets, to categorize the cancer causing genes into their respective classes. These models are implemented using MapReduce and Spark on Hadoop framework that analyzes the dataset in distributed manner on various Datanodes.

Therefore, a comparative analysis is carried out in order to choose a feature selection model with a classifier that provides a better classification accuracy. Figure 5.17 shows the comparative analysis of classification accuracy of the proposed sPSVM classifier with different feature selection models using various microarray dataset. From the obtained results, it can be inferred that among the various feature selection models used in permutation with sPSVM classifier, ANOVA provides better accuracy in the case of all datasets like Leukemia, Breast cancer, Ovarian cancer, GSE24080, GSE15061, and GSE13159 dataset.

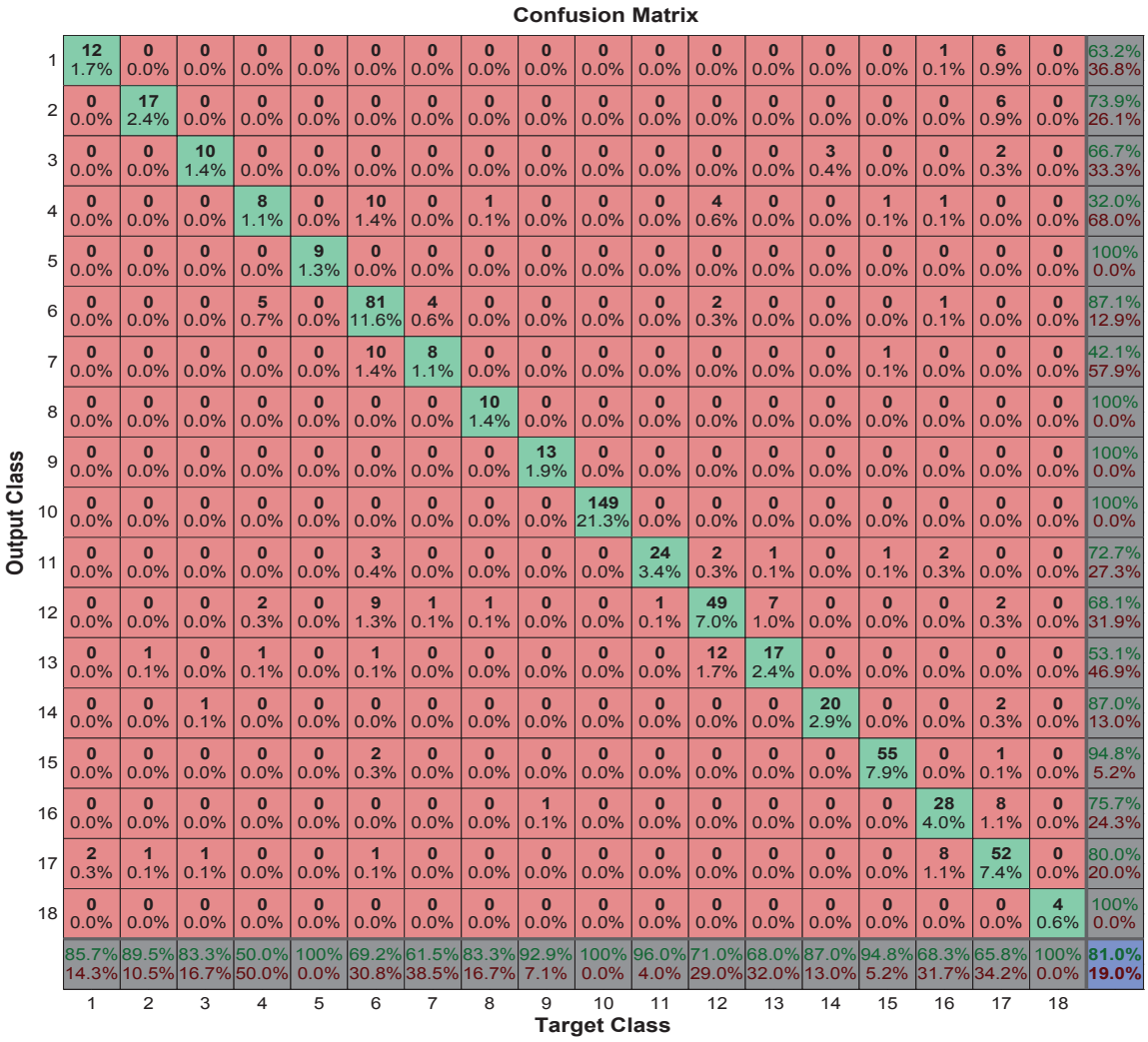


Figure 5.14: Confusion matrix for sPSVM classifier with Kruskal-Wallis using GSE13159 dataset ($f = 15,000$)

5.5 Summary

In this chapter, an attempt has been made to design the classification model for classifying the samples of various datasets into their respective class labels. A PSVM classifier and feature selection using statistical tests (ANOVA, Kruskal-Wallis, and Friedman test) based on MapReduce and Spark have been developed. The proposed approach works in a distributed manner on scalable clusters. The performance of the classifier for various datasets is evaluated by varying the number of features (f). There are three major contributions of this chapter:

i. Harnessing the power of distributed computing for better storage and faster processing of datasets.

ii. Comparative analysis of various feature selection techniques based on statistical tests in permutation with sPSVM classifier.

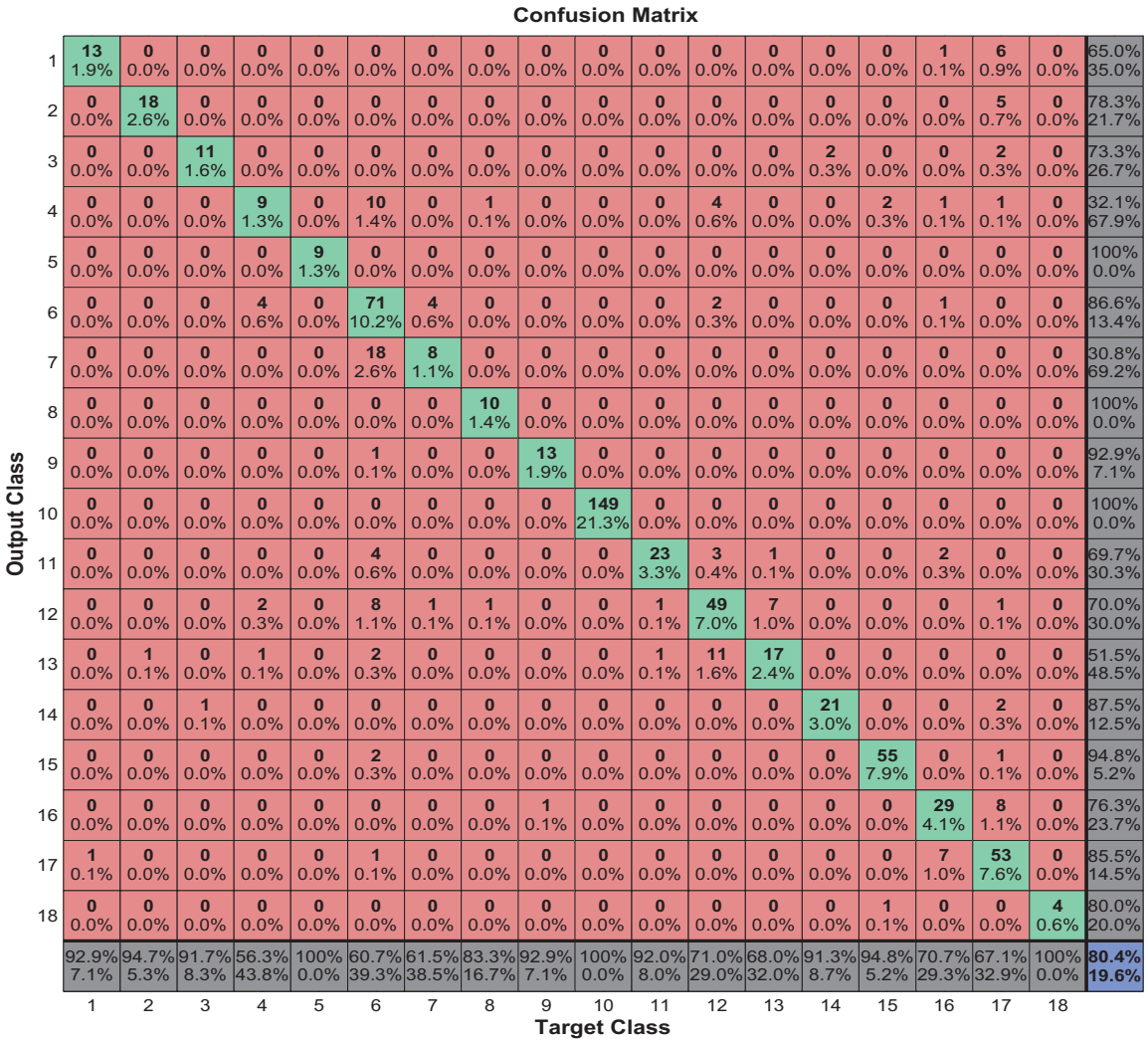
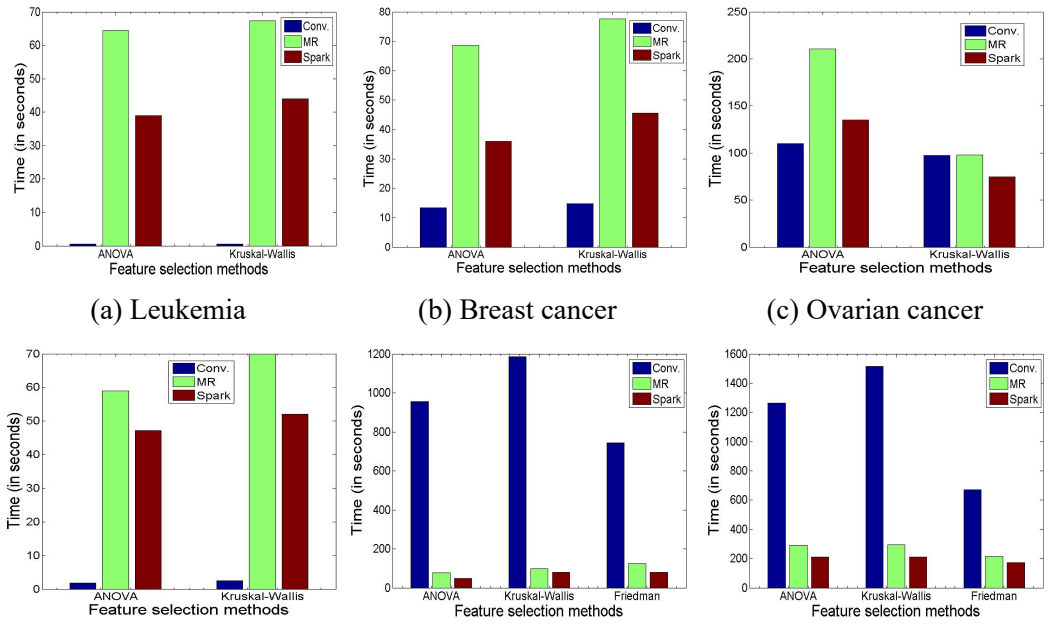


Figure 5.15: Confusion matrix for sPSVM classifier with Friedman using GSE13159 dataset ($f = 6,000$)

iii. Comparative analysis between the size of datasets and the time taken for processing using a conventional system vs. Hadoop cluster (MapReduce and Spark).

From the obtained results it is inferred that the proposed algorithm provides better insights than the traditional PSVM. The sPSVM classifier is able to process the data with any dimensions (GBs, or TBs) on various nodes of clusters. The algorithm is implemented on two scalable frameworks, i.e., MapReduce and Spark on the top of Hadoop cluster and compared with the conventional system. The results shows that the scalable algorithm provides better performance on scalable frameworks than the conventional machine in terms of time, if datasize is with reasonable size. Hence, to gain the insights of scalable algorithms, only the datasets of reasonable size (Big data) are considered for analysis from the next chapter. It is also observed that, among these two scalable frameworks, i.e., MapReduce and Spark, the Spark is faster than the MapReduce.

There are various constraints of MapReduce over Spark like



(a) Leukemia (b) Breast cancer (c) Ovarian cancer

(d) GSE24080 (e) GSE15061 (f) GSE13159

Figure 5.16: Comparison of execution time in sPSVM on Hadoop cluster with MapReduce (MR), Spark, and Conventional system (Conv.) using various feature selection methods.

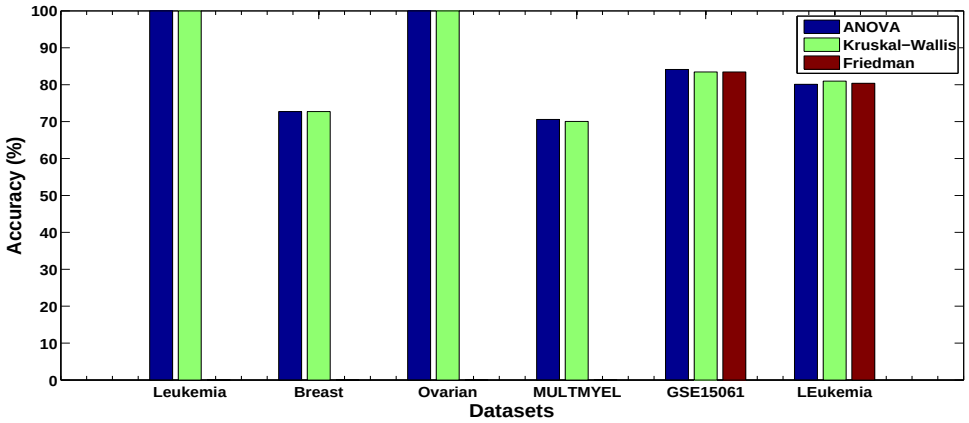


Figure 5.17: Classification accuracy of sPSVM classifier with various feature selection methods using different microarray datasets.

- Data is stored in memory in Spark, but on disk in Hadoop. Secondly, resilient distributed datasets (RDD), which is the data storage model of Spark, uses an intelligent way of ensuring fault tolerance that reduces network I/O.
- RDDs achieve fault tolerance through lineage: if an RDD partition is lost, there is enough information in the RDD about the way this partition was derived from others and we are able to reconstruct that partition. This eliminates the need for replication for fault tolerance. Hadoop, on the contrary, uses replication to achieve fault tolerance.
- Spark uses the concept of RDD which allows us to store data on memory and persist

it as per the requirements. This increases the performance of batch processing job (up to ten to hundred times as much as that of traditional Map Reduce).

- Spark grants caching data in memory, which is useful in case of iterative algorithms such as those used in machine learning.
- Traditional MapReduce and DAG engines are not suitable for these applications as they are based on acyclic data flow: Applications run as a chain of distinct jobs, each of which reads data from stable storage (e.g. a distributed file system) and then writes back to the same. This loading and writing back incurs a significant cost.
- stream processing is possible in Spark, with large input data and so we can deal with only a chunk of data on the fly. This is also helpful for online machine learning, and is suitable for use cases with a requirement for real time analysis(highly ubiquitous industry requirement).
- In particular, MapReduce is inefficient for multi-pass applications that require low-latency data sharing across multiple parallel operations. These applications are quite common in analytics, and include:
  - Iterative algorithms, like machine learning algorithms and graph algorithms like PageRank.
  - Interactive data mining, where it is necessary to load data into RAM across a cluster and query it multiple times.
  - Streaming applications that store and use aggregate state over time.

Due to applicability and advantages of Spark over MapReduce, in the next chapter, various classifiers based on Spark is considered for microarray high-dimensional data classification.

Chapter 6

Classification of Microarray Data using Various Scalable Classifiers on Spark

In this chapter, various scalable classifiers are proposed to classify the high-dimensional data. The proposed classifiers work in a distributed manner and fit into any scalable cluster. The scalability of the proposed classifiers increase as the number of nodes in the cluster increases. To test the scalability of the classifiers a distributed and scalable framework Spark on the top of Hadoop cluster is used. The performance of these proposed classifiers are investigated using microarray high-dimensional data like GSE13159, GSE13204, and GSE15061, which are of different data sizes.

6.1 Introduction

Recently, Big data applications are increasingly becoming the focus of attention because of huge increase in data generation and storage that has taken place in the last few years. Extracting information from these data becomes a challenging task, as the current data mining techniques are not well suited for the new space and time requirements [8]. To overcome these challenges, the new paradigms have been considered to develop scalable algorithms.

After selecting the relevant features using feature selection methods like ANOVA, Kruskal-Wallis, and Friedman statistical tests (as discussed in Chapter 4), various scalable classifiers are considered to classify the microarray datasets. The scalability of the algorithms are tested by varying the data size of high-dimensional data on a Hadoop cluster. The performance of the proposed algorithms in terms of execution time and accuracy are tested on Spark with three slave (worker) nodes and one master (driver) system, which are then compared with that of conventional system. The main motivation of this chapter is to build a model which analyzes the microarray data in an efficient way. The efficiency of the model is measured in terms of performance parameters like, recall, precision, F-Measure, accuracy and time. In other words, the model should behave with high accuracy in a minimal time. To achieve this goal, Spark framework is applied which works in a distributed and scalable manner. It also provides an insight to the classifier to act as a scalable algorithm. In this chapter, various classifiers like, Logistic regression (LR), Support vector machine

(SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), Artificial Neural Network (ANN), Radial basis function network (RBFN) with hybrid learning, and Radial basis function with gradient descent learning based on Spark framework are proposed. The proposed classifiers are investigated with various microarray high-dimensional datasets and their performance is measured.

The rest of the chapter is organized in the following manner:

Section 6.2 presents the proposed work for classifying the microarray high-dimensional data using classifiers based on Spark. The implementation details for the proposed approach are provided in Section 6.3. Results and interpretation details are discussed in Section 6.4, which also presents the comparative analysis in terms of execution time for feature selection techniques and classifiers based on Spark with that of conventional system. The summary of this chapter is mentioned in Section 6.5.

6.2 Proposed work

The presence of a huge number of insignificant features creates bottleneck during the analysis aspect of diseases like cancer. This issue could be resolved by analyzing the dataset with proper perspective. This section presents an approach for classification of microarray data, which consists of three phases:

i. Missing data imputation and normalization methods are used for preprocessing the input data.

ii. Statistical tests, i.e., ANOVA, Kruskal-Wallis and Friedman based on Spark framework on top of Hadoop cluster are used for selecting relevant features as discussed in Chapter 4.

iii. After selecting the relevant features, Spark based Logistic regression (sf-LoR), Spark based Support vector machine (sf-SVM), Spark based Naive Bayes (sf-NB), Spark based K-Nearest Neighbor (sf-KNN), Spark based Artificial Neural Network (sf-ANN), Spark based Radial basis function (sf-RBFN) (both hybrid and gradient descent learning techniques) are applied to classify microarray dataset into their respective classes (binary/multi-class).

Figure 6.1 shows the graphical representation of the proposed approach.

6.3 Implementation

In this section, the implementation of the proposed algorithms using RDD abstraction on Spark framework is discussed.

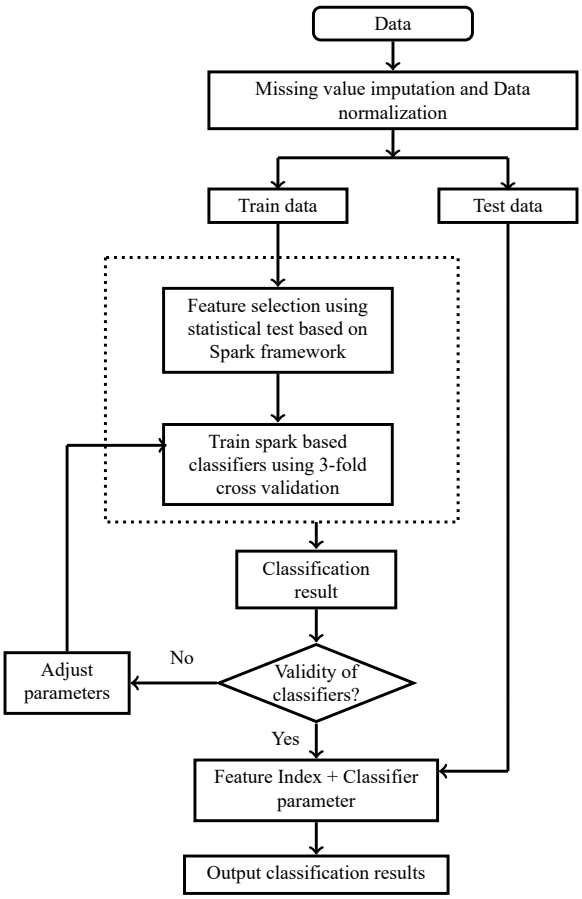


Figure 6.1: Proposed approach for microarray classification.

6.3.1 Classification

Classification is a common form of supervised learning, where algorithms attempt to predict a variable from features of objects using labeled training data. Spark is best suited for running parallel classification algorithms on a large dataset. The basic classifiers supports only binary classification, but can be extended to handle multi-class classification by using either One-versus-all (OVA) or All-versus-all (AVA).

6.3.1.1 Multi-class classificaton using OVA

OVA is used for classifying multi-class problem with the help of binary classifiers. In this procedure, K binary classifiers are used, where K is equal to the number of classes. The i^{th} classifier is trained with positive samples belonging to class i and negative samples not belonging to class i. During testing of an unknown sample, the classifier that gives the highest probability value, is considered, and its corresponding class label is assigned to that sample.

6.3.2 F-Fold cross validation based on Spark

Cross-validation technique is used to generalize the model by reducing the variance and biasness. Here, F-fold cross validation is implemented using the resilient distributed dataset (RDD) of Spark, which works in a distributed and scalable manner. Algorithm 15 shows the implementation of F-fold cross Validation in Spark.

Algorithm 15 F-Fold cross validation based on Spark

Input: $M \times N$ Matrix, where N is number of features and M is number of samples.
Output: Accuracy of model

procedure CROSSVALIDATION($Data$)
  Read $M \times N$ Matrix file from HDFS
  Create RDD of Input Data $M \times N$ Matrix, called $dataRDD$ ▷ $dataRDD$ $contains$ $(classlabels, featureVector)$
  for $i = 1$ to F **do**
    Divide the $dataRDD$ into RDD of training set $trainRDD_i$ and testing set $testRDD_i$
    Initialize Tuning parameter (β) $\leftarrow$ $\lambda = [2^{-10}, 2^5]$ or $HiddenNodes = [10, 30, 50, 90, 110, 150, 200, 300, 400]$ ▷ λ for sf-RBFN and $HiddenNodes$ for sf-ANN
    for $j = 1$ to F **do**
      Divide the training RDD $trainRDD_i$ into RDD of learning set $learnRDD_j$ and validation set $validRDD_j$
      for $k = 1$ to $length(\beta)$ **do**
        Train the model using learning set $learnRDD_j$
        Validate the model using validation set $validRDD_j$
        Calculate the accuracy of the model
      end for
      Calculate the mean accuracy of the model corresponding to each β
    end for
    Select β, corresponding to model having high accuracy (called β')
    Train the model with training RDD $trainRDD_i$ using β'
    Test the model with testing RDD $testRDD_i$ using β' and calculate its accuracy
  end for
end procedure

6.3.3 Logistic Regression based on Spark (sf-LoR)

Logistic regression is a parametric form for the distribution $P(Y|X)$ where Y is a discrete value and $X = \{x_1, ..., x_n\}$ is a vector containing discrete or continuous values [102, 151]. The parametric model of logistic regression can be written as:

$$P(Y = 1|X) = \frac{1}{1 + exp(w_0 + \sum_{i=1}^{n} w_i X_i)} \tag{6.1}$$

and

$$P(Y = 0|X) = \frac{exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + exp(w_0 + \sum_{i=1}^{n} w_i X_i)} \tag{6.2}$$

The parameter w of the logistic regression is chosen by maximizing the likelihood of conditional data. It is the probability of the observed Y values in the training data. The constraint can be written as:

$$w \leftarrow \underset{w}{argmax} \sum_{l} lnP(Y^l|X^l, w) \tag{6.3}$$

Algorithm 16 describes the implementation details of Logistic regression based on Spark.

Algorithm 16 sf-LoR: Spark based Logistic Regression Classifier

Let $X = \{(x_1, y_1)(x_2, y_2), ..., (x_i, y_i)\}$ is a set of M training samples with N attributes, where $i = 1, 2, ..., M$, i.e., $x \in \mathbf{R}^{M \times N}$ and $y \in \mathbf{R}^{M \times 1}$
Test set $x_t \in \mathbf{R}^{p \times N}$ ***Output***: Calculation of $classLabel$ and $Accuracy$.

1: **procedure** DRIVER_MAIN(X, x_t)
2: Read train data X and test data x_t from HDFS
3: Create RDD of train data and test data i.e., called $trainRDD$ and $testRDD$ respectively.
4: Initialize weight vector w and broadcast to all worker nodes.
5: **for** $iter \leftarrow 1$ to $iterations$ **do**
6: $[gradient] \leftarrow trainRDD$.*map*$\{ \; p \Rightarrow p.x * (1/\,(1 + exp(-p.y + (w \; dot \; p.x))) - 1) * p.y \; \}$.*reduce*$((a, b) \Rightarrow a + b)$
7: Update w as, $w- = gradient$
8: Broadcast updated w to worker nodes
9: **end for**
10: $result \leftarrow testRDD$.*map*$(x \Rightarrow x \; dot \; w)$.filter$(x \Rightarrow$
$if(x \geq 0.5)\{ \; var \; label = 1\}$
$else$
$var \; label = 0$
$return \; label$). *saveAsTextFile*("$\langle HDFSPath \rangle$/result")
11: **end procedure**

6.3.4 Support Vector Machine based on Spark (sf-SVM)

SVM is a superior data classifier as it is capable in handling non-linear classification problems by mapping input vectors into a high-dimensional feature space, with a kernel function [107]. The multi-class classification problem is solved by decomposing it to several binary problems using OVA for which the standard SVM is used.

Let A be the matrix of M samples and N features in the space of $\mathbf{R}^N$ represented by $M \times N$ matrix, D is the class label $+1$ or -1 for two class problem. To classify the M samples with Support vector machine (SVM) with linear kernel is given by the following

quadratic problem [145, 151].

$$\begin{aligned} \min_{w,\gamma,\xi} J(w,\gamma,\xi) = Ce'\xi - \frac{1}{2}w'w \\ s.t. \quad D(Aw - e\gamma) + \xi \geq e \\ e \geq 0 \end{aligned} \tag{6.4}$$

where, C is the regularization factor, ξ is the error term, e is a unit vector, w is the normal to the bounding planes:

$$\begin{aligned} x'w = \gamma + 1 \\ x'w = \gamma - 1 \end{aligned} \tag{6.5}$$

that constrained most of the sets in to A^+ or A^- respectively, and γ is the relative location of planes to the origin. It divides the space into two half spaces such that the data points of different classes are separated. The plane acts as a linear classifier as follows:

$$sign(x'w - \gamma) \begin{cases} = 1, & then\ x \in +1 \\ = -1, & then\ x \in -1 \end{cases} \tag{6.6}$$

In SVM, as the size of the data samples increases, both the training time and the computational complexity of the algorithm increases. The implementation details of Spark based SVM is described in [152].

6.3.5 Naive Bayes based on Spark (sf-NB)

Naive Bayes method is a supervised learning algorithm which applies Bayesian theorem with the assumption of independence between every pair of features [102]. The Bayes theorem states that

$$P(A/B) = P(B/A) * P(A)/P(B) \tag{6.7}$$

Thus, for a given sample X with attribute value $X = (a_1, a_2, ..., a_N)$ and class V_j. The probability value that X belongs to class V_j can be calculated as:

$$P(V_j/X = (a_1, a_2, ..., a_N)) = P(X/V_j)P(V_j)/P(X) \tag{6.8}$$

$$= P(V_j) \prod_{i=1}^{N} (P(a_i/V_j))/P(X) \tag{6.9}$$

Where $P(V_j)$ is the prior probability for each class and $P(a_i/V_j)$ is the likelihood probability for each attribute, where $i = 1, ..., N$ and $j = 1, ..., C$.

In the testing phase the likelihood for each attribute for a given class is calculated and a new instance x^t having attributes $(a_1^t, a_2^t, ..., a_N^t)$ is classified as:

$$\hat{P}(a_i^t/V_j) = (phi_{ij}^{a_i^t}) * (1 - phi_{ij})^{(1-a_i^t)}) \tag{6.10}$$

where, phi is the weight of each attribute for a given class; which is calculated as follows:

$$phi = X' * Y/M \tag{6.11}$$

where X is the matrix of training samples ($M \times N$), Y is a matrix of dimension $M \times C$, where M is the number of training samples, N is the number of attributes, and C is the number of classes. Each row of Y consists of zeros except for the position that indicates the class label.

After calculating the value of $P(x_i^t/V_j)$, the new instance x^t is labeled as the following equation

$$V_{nb} = \underset{V_j \in V}{argmax}\ P(V_j) \prod_{i=1}^{N} \hat{P}(a_i^t/V_j) \tag{6.12}$$

It is implemented using two procedures, one for training and other for testing. In the training phase, the prior probabilty $P(V_j)$ and the likelihood factor are calculated, which are described in the algorithm as shown in Algorithm 17. After calculating the prior and likelihood probabilities using training set, testing data is classified using the Algorithm 18. Algorithm 19 describes the driver program which is executed on RDDs of training data and testing data, and final classification result is obtained.

Algorithm 17 Training: Spark based Naive Bayes classifier

Input: Let $X = \{(x_i, y_i) | x_i \in \mathbf{R}^N, y_i \in \mathbf{R}^1, i = 1, 2, ..., M\}$ is a set of training samples with attributes $(a_1, a_2, ..., a_N)$.

Output: Calculation of prior $(P(V_j))$ and likelihood probabilities $P(X/V_j)$

procedure MAP_NBTRAIN(X_i) ▷ $i = 1, 2, ..., M$
  for each sample X_i **do** ▷ runs in parallel
    Parse the label and the value of each attributes.
    Form matrix $y \in \mathbb{R}^{1 \times C}$ from label and matrix $x \in \mathbb{R}^{1 \times N}$ consisting of the values of each attribute.
    Calculate $S = x' * y$
    Emit $\langle label, S \rangle$ ▷ *return*
  end for
end procedure
procedure REDUCE_NBTRAIN($\langle label, S \rangle$)
  Read a key-value pair from mapper
  Sum all the S-values to generate $phi \in \mathbb{R}^{N \times C}$.
  Calculate the prior probability $P(V_j)$ by adding the label value.
  Emit $\langle P(V_j), phi \rangle$ ▷ *return*
end procedure

Algorithm 18 Testing: Spark based Naive Bayes classifier

Input: Let $X^t = \{(x_i^t, y_i^t)|x_i^t \in \mathbf{R}^N, y_i^t \in \mathbf{R}^1, i = 1, 2, ..., p\}$ is a set of testing samples with attributes $(a_1^t, a_2^t, ..., a_N^t)$, prior $(P(V_j))$, and phi
Output: Calculation of class label and accuracy.

procedure MAP_NBTEST(X_i^t) ▷ $i = 1, 2, ..., p$
  for each test sample X_i^t **do** ▷ runs in parallel
    Calculate the probability of each attribute using the Equation 6.10.
    Calculate the posterior probability of testing sample with each class label using "$V = P(V_j) \prod_{i=1}^{N} \hat{P}(a_i^t/V_j)$".
    Emit $\langle id, label, V \rangle$ ▷ *return*
  end for
end procedure
procedure REDUCE_NBTEST($\langle id, label, V \rangle$)
  Read a key-value pair from mapper
  Calculate maximum a posteriori (MAP) using Equation 6.12, called V_{nb} and estimate the class label j. ▷ $j = 1, 2, ..., C$
  Emit $\langle id, j \rangle$ ▷ *return*
end procedure

Algorithm 19 sf-NB: Spark based Naive Bayes

Input: Let $X = \{(x_i, y_i)|x_i \in \mathbf{R}^N, y_i \in \mathbf{R}^1, i = 1, 2, ..., M\}$ is a set of training samples with attributes $(a_1, a_2, ..., a_N)$. Test set $X^t \in \mathbf{R}^{p \times N}$
Output: Calculation of $classLabel$ and $Accuracy$.

procedure DRIVER_MAIN(X, x_t)
  Read train data X and test data x_t from HDFS
  Create RDDs for train data and test data i.e., called $trainRDD$ and $testRDD$ respectively.
  $[P(V_j), \ phi] \leftarrow trainRDD$.*map*(MAP_NBTRAIN).*map*(REDUCE_NBTRAIN).*collect*()
  Broadcast $[P(V_j), \ phi]$ to all worker nodes
  $result \leftarrow testRDD$.*map*(MAP_NBTEST).*map*(REDUCE_NBTEST).*saveAsTextFile*("$\langle HDFSPath \rangle$/result")
end procedure

6.3.6 K-Nearest Neighbor based on Spark (sf-KNN)

The training dataset ($X = \{(x_i, t_i)|x_i \in \mathbf{R}^N, t_i \in \mathbf{R}^c, i = 1, 2, ..., M\}$) and testing dataset ($S = \{(s_i)|s_i \in \mathbf{R}^N, i = 1, 2, ..., M\}$) are read from the HDFS using spark context and transformed into RDD, i.e., $trainRDD$ and $testRDD$ for training data and testing data respectively. The $trainRDD$ is broadcasted to all the worker nodes. The implementation details of Spark based K-Nearest neighbor is described in Algorithm 20. Finally, the result obtained is saved in HDFS.

Algorithm 20 sf-KNN: Spark based KNN

Input: Let $X = \{(x_i, t_i)|x_i \in \mathbf{R}^N, t_i \in \mathbf{R}^c, i = 1, 2, ..., M\}$ be the training set and $S = \{(s_i)|s_i \in \mathbf{R}^N, i = 1, 2, ..., M\}$ is a set of testing samples to be classified.
Output: Classification result of testing instance.

procedure MAP_KNN(s_i) ▷ $i = 1, 2, ..., M$
  for each testing samples s_i **do** ▷ runs in parallel
    for each training samples x_j **do**
      Calculate the Euclidean distance ($dist$) between a test sample and a training sample.
      Accumulate distance values for each training sample
    end for
    Emit $\langle s_i, (dist_1, dist_2, ..., dist_j)\rangle$ ▷ *return*
  end for
end procedure
procedure REDUCE_KNN($\langle s_i, (dist_1, dist_2, ..., dist_j)\rangle$)
  for each testing sample $s_i \in S$ **do**
    Sort the distance values in ascending order obtained from key-value pairs.
    Select the K nearest neighbors (training samples) to the testing sample.
    Assign the testing sample (s_i) to the most frequent class (e.g., 'c') in the set of training samples.
    if $(a\ tie\ occurs)$ **then**
      The sum of distances of the neighbors in each class is computed.
      if $(no\ tie\ occurs)$ **then**
        Move s_i into the 'minimum sum' class
      else
        Move s_i into the last 'minimum sum' class.
      end if
    else
      Move s_i into the majority class.
    end if
  end for
  Return $\langle s_i, c_i\rangle$ ▷ *return*
end procedure
procedure DRIVER_MAIN(X, S)
  Read train data X and test data S from HDFS
  Create RDD of train data and test data i.e., called $trainRDD$ and $testRDD$ respectively.
  Broadcast $trainRDD$ to all worker nodes
  $result \leftarrow$ testRDD.*map*(MAP_KNN).*map*(REDUCE_KNN).*saveAsTextFile*("$\langle HDFSPath\rangle$/result")
end procedure

6.3.7 Artificial Neural Network classifier based on Spark (sf-ANN)

Artificial Neural Network (ANN) is a network of simulated neurons. It is inspired by the examination of central nervous system. Warren in 1943 created a computational model for neural networks based on mathematical formulations and algorithms [104]. ANN is a

non-linear data modeling tool, which is usually used to model complex relationships between inputs and outputs, and find patterns in data. This section gives a brief description of the basic structure and working of ANN technique applied for predicting the organizational performance. In general, a neuron in an ANN is a node having some activation function ($f(.)$) which maps the input vector (X) to the output vector (Y). The neurons (a signaling element) connected with synapses are called as weight vector (W). ANN architecture utilizes its computational features that can be well applied for prediction of the outcome involved in the analysis.

The Back Propagation Neural Network (BPNN) is one of the most widely applied neural network. It mainly involves the feed forward network and the back propagation learning, and uses the iterative gradient algorithm to minimize the mean square error (MSE) between the actual output of a multilayer feed-forward perceptron and the desired output. The BPNN can obtain the activation value by feed forward step, and adjusts the 'weights' and 'biases' according to the difference between the desired and actual network outputs by using the back propagation step. The execution of these two steps, i.e., feed forward and back propagation terminate when the convergence criteria of network is satisfied [153, 154]. The mathematical intuition behind selecting the number of layers in the network is that each layer in a feed-forward multi-layer perceptron adds its own level of non-linearity that cannot be contained in a single layer. The inputs in the single layer network are only linearly combined, and hence cannot produce the non-linearity that can be seen through multiple layers. In this study, a three layer network is considered, where in the first layer is simply the input layer, which encodes a set of features learned from the inputs, while second layer encodes a different (higher-level, more abstracted) set of features learned from the outputs of first layer, and the third layer i.e., the output layer generates a feasible output. Particularly, this network is used to classify the microarray datasets which have similar type of features. Each layer in the network corresponds to each dimension of the data. Hence, a single hidden layer is sufficient to achieve the desired output. But, as the number of layers increases, the complexity of the network also increases and the chances of over-fitting of network may also arise.

In this study, the sigmoid function is used as the activation function ($f(.)$) in both the hidden and the output layer. Figure 6.2 shows the typical architecture of BPNN model.

Let $X = \{(\{1, x_i\}, t_i)|x_i \in \mathbf{R}^N, t_i \in \mathbf{R}^1, i = 1, 2, ..., M\}$ be the training set, where M is the number of training samples and N is the number of features. A scalar value 1, which acts as a bias is added to each sample vector x_i. The class label $t \in \mathbf{R}^{M \times m}$ is defined in the

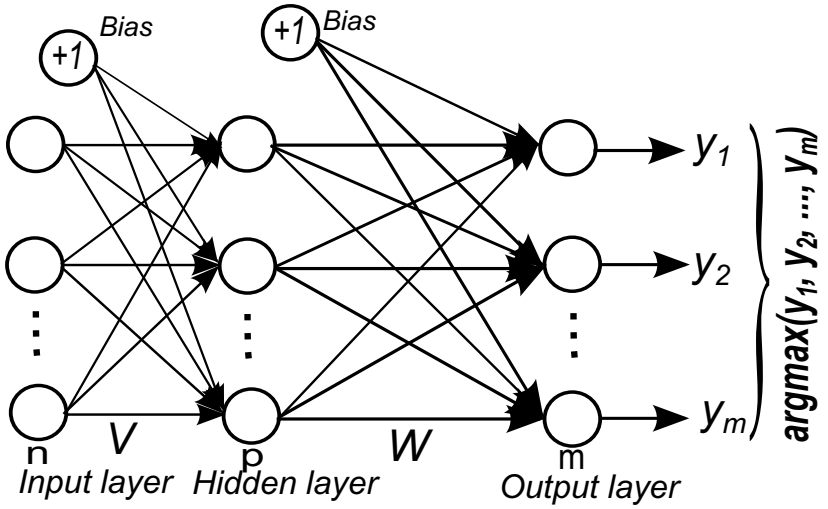


Figure 6.2: Architecture of back-propagation neural network (BPNN).

following way, where m is maximum number of classes.

$$t_i = \begin{cases} \{1, 0, ..., 0\}\,; & if\ sample\ i\ belongs\ to\ class\ 1 \\ \{0, 1, ..., 0\}\,; & if\ sample\ i\ belongs\ to\ class\ 2 \\ \quad\vdots & \\ \{0, 0, ..., 1\}\,; & if\ sample\ i\ belongs\ to\ class\ m \end{cases}$$

The weight vector w_{jk} between j^{th} node of hidden layer to k^{th} node of output layer is updated using Equation 6.15. The weight vector v_{ij} between i^{th} node of input layer to j^{th} node of hidden layer is updated using Equation 6.16. δw and δv in Equation 6.13 and 6.14 respectively, are calculated for each sample, which are then accumulated in cluster nodes with the help of accumulator variables. The accumulator variables, i.e., $grad_w$ and $grad_v$, sums the δw and δv values respectively of all samples in a distributed environment. The final gradients ($Mean(grad_w)$ and $Mean(grad_v)$) are equal to the mean of all sample gradients ($\frac{1}{M}\sum_{i=1}^{M}\delta w$ and $\frac{1}{M}\sum_{i=1}^{M}\delta v$) obtained in each epoch.

$$\delta w_{jk} = \alpha\delta_k f(v_{0j} + \sum_{i=1}^{n} x_i v_{ij}) \qquad (6.13)$$
$$\delta w_{0k} = \alpha\delta_k$$

$$\delta v_{ij} = \alpha\delta_j x_i \qquad (6.14)$$
$$\delta v_{0j} = \alpha\delta_j$$

$$w_{jk}(t+1) = w_{jk}(t) + Mean(grad_w) + \eta[w_{jk}(t) - w_{jk}(t-1)] \qquad (6.15)$$

$$v_{ij}(t+1) = v_{ij}(t) + Mean(grad_v) + \eta[v_{ij}(t) - v_{ij}(t-1)] \qquad (6.16)$$

where $f(.)$ is the activation function, α is the learning rate that affects the convergence of BPN network and $\eta \in [0, 1]$ is the momentum factor. The back propagation error δ_k and δ_j are calculated using Equations 6.17 and 6.18 respectively.

$$\delta_k = (t_k - y_k) f'(w_{0k} + \sum_{j=1}^{p} f(v_{0j} + \sum_{i=1}^{n} x_i v_{ij}) * w_{jk}) \tag{6.17}$$

$$\delta_j = \left(\sum_{k=1}^{m} \delta_k w_{jk} \right) f'(v_{0j} + \sum_{i=1}^{n} x_i v_{ij}) \tag{6.18}$$

where $f'(.)$ is the derivative of $f(.)$, n is number of input nodes, p is number of hidden nodes and m is the number of output nodes in $(n - p - m)$ three layered network. The predicted output y_k is calculated using Equation 6.19 and their corresponding class labels are obtained by using Equation 6.20.

$$y_k = f(w_{0k} + \sum_{j=1}^{p} f(v_{0j} + \sum_{i=1}^{n} x_i v_{ij}) * w_{jk}) \tag{6.19}$$

$$classLabel = \underset{k=1,2,...,m}{\operatorname{argmax}} \{ Y = (y_1, y_2, y_3, ..., y_m) \}) \tag{6.20}$$

The proposed Spark based ANN (sf-ANN) algorithm consists of a driver program and various RDD transformation procedures like, ANNTrain( ) and ANNTest( ). The driver program is responsible for creating a SparkContext, RDDs, and performing transformations and actions. In RDD transformation method, the distributed dataset (RDD) is transformed by reading a line (single sample) and returning the result as per the Transformation method. Parallel transformation operations on RDDs are performed by worker nodes and the results are returned to the driver program. The driver performs RDD action operation by collecting the results from all worker nodes. Figure 6.3 shows the data flow diagram of sf-ANN.

Algorithm 21 Transformation method for training ANN Model

Input: Training Set X
Output: $grad_w = \sum \delta w$ and $grad_v = \sum \delta v$

```
1: procedure ANNTrain(x)
2:     for each sample x in RDD do                    ▷ Runs in parallel
3:         Calculate δw from Equation 6.13
4:         Calculate δv from Equation 6.14
5:         grad_w += δw
6:         grad_v += δv
7:     end for
8:     return grad_w, grad_v
9: end procedure
```

Algorithm 21 shows the implementation of ANNTrain() method. This method is used for calculating δw and δv for each input sample and finally summing it up using an accumulator variable. In the driver program, $grad_w$ and $grad_v$ are initialized as accumulator

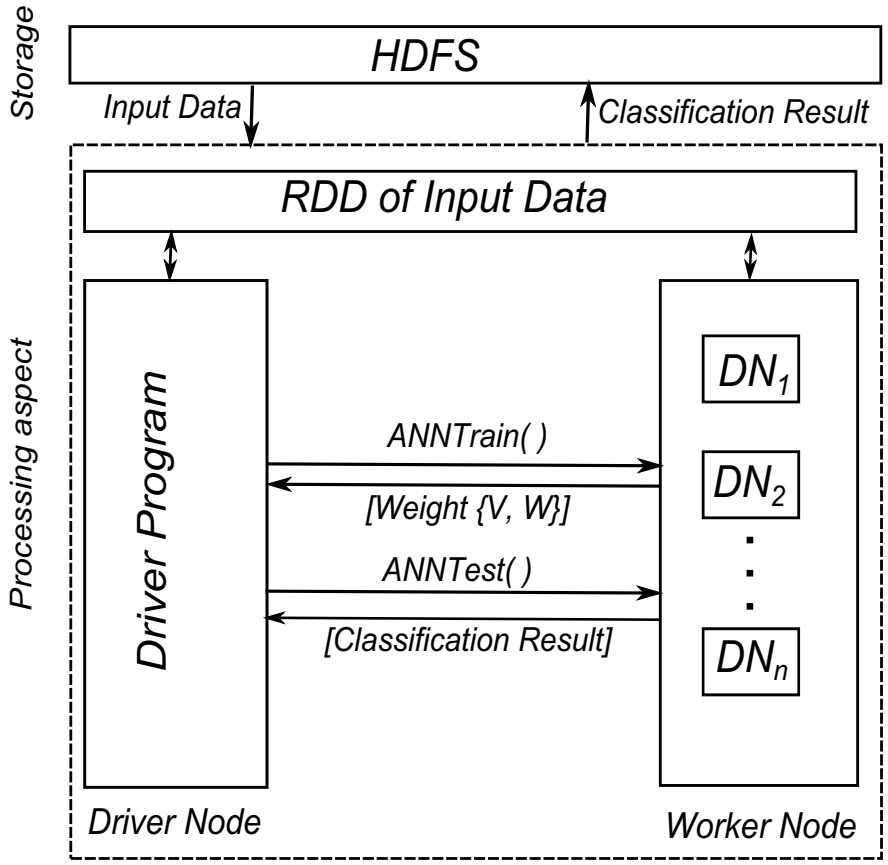


Figure 6.3: Workflow of sf-ANN algorithm

variables, which are common to all worker nodes. The value of these variables can only be read within the driver program. The worker nodes can only update the accumulator variables. The average value of $grad_w$ and $grad_v$ obtained in each iteration are used to update weights w_{jk} and v_{ij} respectively, which are then broadcasted to worker nodes. This process continues until the given iteration criteria is met. The final weights obtained gives the decision boundary parameter for multi-class classification problem.

Algorithm 22 shows the implementation of ANNTest(). This method transforms the RDD of testing dataset by returning the predicted class label of each sample along with original class label. The result from all worker nodes are collected by the driver program, which then calculates the accuracy of sf-ANN. Algorithm 23 shows the implementation of sf-ANN in driver program using RDD Transformation and Action operation.

6.3.8 Radial Basis Function Network based on Spark (sf-RBFN)

Radial basis function (RBFN) is considered as a variant of artificial neural network [155]. It was first formulated by Broomhead et al. [156] and popularized subsequently by Moody et al. [157]. The idea of RBFN is derived from the theory of function approximation. An RBF network consists of three layers, namely the input layer, the hidden layer, and the output layer.

Algorithm 22 Transformation method for testing ANN Model

Input: Testing Set X_{test}
Output: $\langle Predicted\_classLabel, Original\_classLabel \rangle$

```
1: procedure ANNTest(x)
2:     for each sample x in RDD do                          ▷ Runs in parallel
3:         for each class do
4:             Calculate y_k using Equation 6.19
5:         end for
6:         Calculate classLabel using Equation 6.20
7:         return ⟨Predicted_classLabel, Original_classLabel⟩
8:     end for
9: end procedure
```

Algorithm 23 ANN based on Spark

Input: Training Set X, Testing Set X_{test}
Output: Classification result of X_{test}

```
1: procedure Driver_Main(X, X_test)
2:     Read X file from HDFS
3:     Create RDD of train data X, called trainRDD
4:     trainRDD ← trainRDD.cache()
5:     Initialize and broadcast sf-ANN weights w_jk and v_ij to worker nodes
6:     Initialize grad_w and grad_v as accumulator variables
7:     for iter = 1 to iterations do
8:         [ grad_w, grad_v] ← trainRDD.map(ANNTrain( )).collect( )
9:         Update w_jk using Equation 6.15
10:        Update v_ij using Equation 6.16
11:        Restrict w_jk, v_ij ∈ [−1, 1]
12:        Broadcast updated w_jk and v_ij to worker nodes
13:    end for
14:    Create RDD of test data X_test, called testRDD
15:    Output ← testRDD.map(ANNTest( )).collect( )
16:    Calculate accuracy of sf-ANN model
17: end procedure
```

- Input layer: transmits the input vector to each unit in the hidden layer.
- Hidden layer: maps the input space into nonlinear space using a radial basis function. The transformation from input space to hidden space is nonlinear.
- Transformation from hidden unit space to output space is linear.

The input layer consists of source nodes. In hidden layer each neuron computes its output using a radial basis function, which is in general a Gaussian function. The output layer builds a linear weighted sum of the output of hidden neurons and supplies the response of the network.

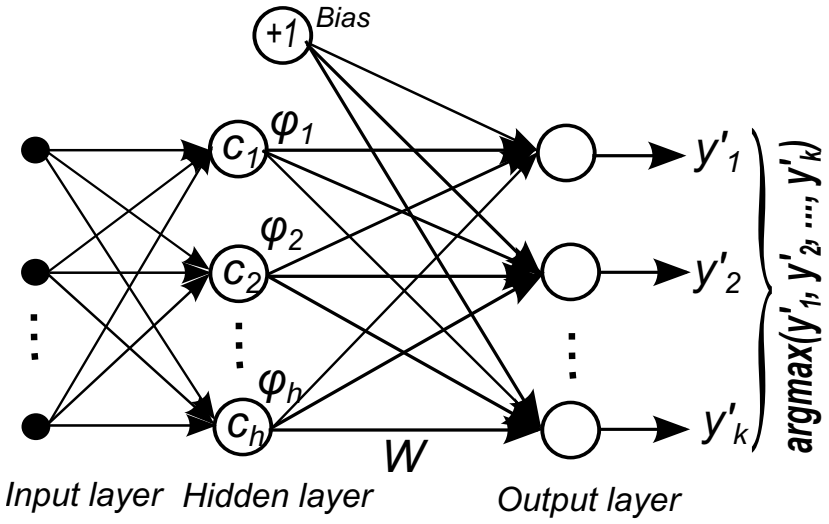


Figure 6.4: Radial basis function network

The parameters of RBFN viz., centers (used in Gaussian function) and weights (between the neurons of hidden layer and output layer) are trained using different techniques like gradient descent, and hybrid learning. Figure 6.4 shows the architecture of RBF network.

6.3.8.1 Hybrid learning

Hybrid learning technique is applied for updating the centers and weights of sf-RBFN model. The centers are obtained by applying KMeans clustering algorithm based on Spark framework, and the weights (W) are updated using gradient descent learning method.

Let $X = \{(x_i, y_i)|x_i \in \mathbf{R}^N, y_i \in \mathbf{R}^1, i = 1, 2, ..., M\}$ be the training set, where M is the number of training samples, and N is the number of features. The cluster centers $(C_1, C_2, ..., C_h)$ of training set are obtained by applying K-Means clustering algorithm based on Spark framework. The input vectors are then mapped using the Gaussian function as mentioned in Equation 6.21 at hidden layer of sf-RBFN. The number of nodes in hidden layer is equal to the number of centers (h), which are obtained using Spark based K-Means clustering algorithm. The weight vector $W \in \mathbf{R}^{k\times(1+h)}$ of sf-RBFN is optimized using the cost (J) in Equation 6.22 and gradient in Equation 6.23, which are obtained from gradient descent based on Mean Square Error (MSE).

$$\phi_j(x) = \exp(-\frac{\|x - c_j\|^2}{(\sigma_j)^2}) \tag{6.21}$$

where $\|.\|$ denotes the sum of squared distance between the input vector and centers, $x \in R^N$ is the input, $\phi_j(x)$ is the Gaussian radial basis function, $c_j \in R^N$ and σ_j are the center and width of j^{th} hidden neuron respectively.

$$J_k = \left[\frac{(\Phi \cdot W_i - Y_i)^2}{2 * M} + \left\{\frac{\lambda}{2 * M}\sum_{j=2}^{h+1} w_j^2\right\}\right] \tag{6.22}$$

$$g_k = \left[\frac{\Phi(\Phi \cdot W_i - Y_i)}{M} + \left\{ \frac{\lambda}{M} * W_i[2 : h+1] \right\} \right] \tag{6.23}$$

$$grad_{list} = \{g_1, g_2, ..., g_k\} \tag{6.24}$$

where λ is the regularization parameter, $\Phi = \{1, \phi_1, \phi_2, ..., \phi_h\}$, w_j is the output layer weights of the RBFN, h denotes the number of clusters, i.e., number of hidden nodes in hidden layer and k is the number of classes. J_k and g_k in Equations 6.22 and 6.23 are the cost function and gradient of each class k respectively. The $grad_{list}$ in Equation 6.24 is a list of gradient values obtained for each class. This gradient list ($grad_{list}$) is for a single sample, which is accumulated for each samples in cluster nodes. $gradVector$ is the accumulator variable which sums the gradient value of all samples ($\sum_{i=1}^{M} grad_{list}$) in a distributed environment. The final gradient ($Mean(gradVector)$) is equal to the mean of all sample gradients ($\frac{1}{M}\sum_{i=1}^{M} grad_{list}$) obtained in each epoch. The weights are updated using Equation 6.25. Class label $Y \in \mathbf{R}^{M \times k}$ is defined in the following way; where k is maximum number of classes..

$$Y_i = \begin{cases} \{1, 0, ..., 0\}; & if\ sample\ i\ belongs\ to\ class\ 1 \\ \{0, 1, ..., 0\}; & if\ sample\ i\ belongs\ to\ class\ 2 \\ \quad \vdots & \\ \{0, 0, ..., 1\}; & if\ sample\ i\ belongs\ to\ class\ k \end{cases}$$

The predicted output of samples are obtained by using Equation 6.26 and their corresponding class labels are obtained by using Equation 6.27.

$$W_{new} = W_{old} - Mean(gradVector) \tag{6.25}$$

$$F(x) = W * \Phi^T \tag{6.26}$$

$$classLabel = \underset{i=1,2,...,k}{\operatorname{argmax}}\ F(x) \tag{6.27}$$

The proposed sf-RBFN algorithm using Hybrid learning technique consists of driver program and various RDD Transformation methods like, KMeans( ), hlRBFNTrain( ) and RBFNTest( ). The driver program is responsible for creating a SparkContext, RDDs, and performing transformations and actions. In RDD transformation method, the distributed dataset (RDD) is transformed by reading a line (single sample) and returning the result as per the Transformation method. Parallel transformation operation on RDDs are performed by worker nodes and the results are returned to the driver program. The driver performs RDD action operation by collecting the results from all worker nodes. Figure 6.5 shows the workflow of sf-RBFN algorithm using Hybrid learning technique.

Algorithm 24 shows the implementation of KMeans( ) and filterData( ) Transformation method. KMeans( ) method is used to transform the RDD of training dataset by assigning each sample to its closest cluster. The output of KMeans( ) method is a new

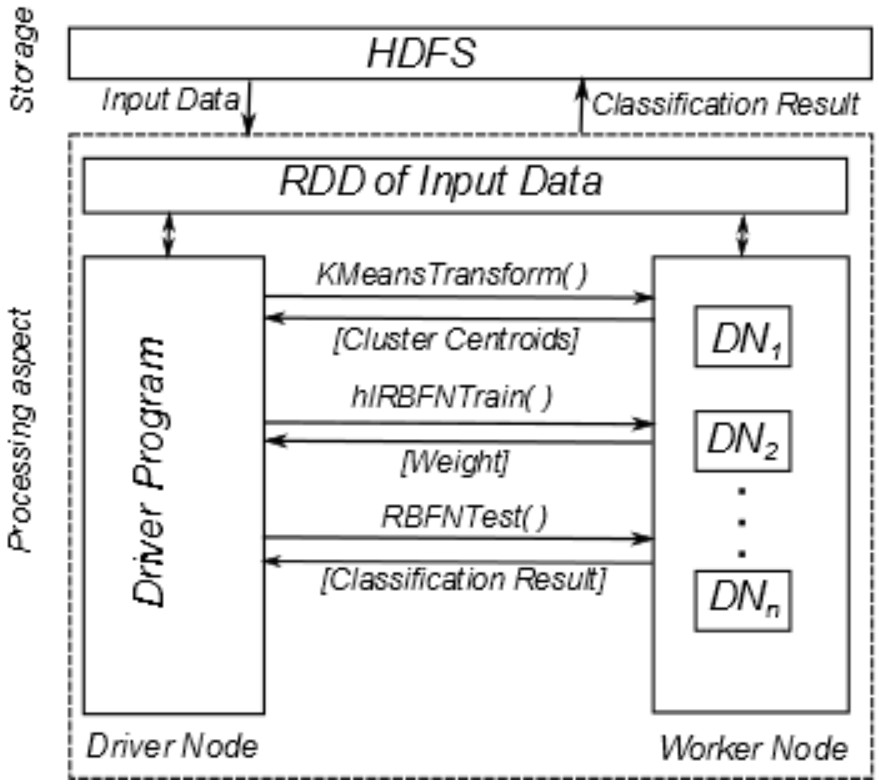


Figure 6.5: Workflow of sf-RBFN algorithm using Hybrid learning technique

RDD, which is passed to FILTERDATA( ) method. The FILTERDATA( ) method transforms the RDD by returning the data points of each sample associated with a cluster. The new cluster centroids are computed with each RDD transformation and action operations are performed in driver program. This process terminates when the given condition (convergence criteria) is satisfied. The final cluster centroids obtained are then broadcasted to worker nodes. Algorithm 25 shows the implementation of HLRBFNTRAIN( ) method. This method is used for optimizing the weights (W) of sf-RBFN model by calculating the gradient ($gradVector$) in one single iteration. In the driver program, $gradVector$ is initialized as an accumulator variable, which is common to all worker nodes. The value of this variable can only be read within the driver program. The worker nodes can only update an accumulator variable. The $gradVector$ value obtained in each iteration is used to update W, which is then broadcasted to worker nodes. This process continues until the given iteration criteria is met. The final weights (W) obtained gives the decision boundary parameter for multi-class classification problem.

Algorithm 26 shows the implementation of RBFNTEST(). This method transforms the RDD of testing dataset by returning the predicted class label of each sample along with original class label. The result from all worker nodes are collected by the driver program, which then calculates the accuracy of sf-RBFN using hybrid learning technique. Algorithm 27 shows the implementation of sf-RBFN with hybrid learning technique in driver program, using RDD transformations and actions.

Algorithm 24 KMeans Transformation method

Input: Training Set X
Output: $clusterPoints$

procedure KMEANS(X)
    Create RDD of train data X, called $trainRDD$
    for each sample x in $trainRDD$ **do** ▷ Runs in parallel
        for each c in cluster centroids **do**
            Find c having minimum Euclidean distance with sample x
        end for
        Assign sample x to closest cluster c
        Return $\langle sample\_id, assigned\_cluster \rangle$
    end for
end procedure
procedure FILTERDATA($sample\_id, assigned\_cluster$)
    for each sample x in $trainRDD$ **do** ▷ Runs in parallel
        Find $datapoints$ corresponding to $sample\_id$
        Return $\langle datapoints, assigned\_cluster \rangle$
    end for
end procedure

Algorithm 25 Transformation method for training RBFN Model using Hybrid learning technique

Input: Training Set X
Output: $gradVector$

procedure HLRBFNTRAIN(X)
    Create RDD of train data X, called $trainRDD$
    for each sample x in $trainRDD$ **do** ▷ Runs in parallel
        for each c in cluster centroids **do**
            Calculate ϕ using Equation 6.21
        end for
        for each class k **do**
            Calculate J_k using Equation 6.22
            Calculate g_k using Equation 6.23
        end for
        Calculate $grad_{list}$ using Equation 6.24
        $gradVector += grad_{list}$
    end for
    Return $gradVector$
end procedure

6.3.8.2 Gradient descent

In this technique both the centers (C) and weights (W) of sf-RBFN are optimized using gradient descent learning method. Let $X = \{(x_i, y_i) | x_i \in \mathbf{R}^N, y_i \in \mathbf{R}^1, i = 1, 2, ..., M\}$ be the training set, where M is the number of training samples, and N is the number of features. The optimization of weight vector $W \in \mathbf{R}^{k \times (1+h)}$ is similiar to sf-RBFN using Hybrid

Algorithm 26 Transformation method for testing RBFN Model

Input: Testing Set X_{test}
Output: $\langle Predicted\_classLabel, Original\_classLabel \rangle$

procedure RBFNTEST(X_{test})
  for each sample x in test data X_{test} **do** ▷ Runs in parallel
    for each c in cluster centroids **do**
      Calculate ϕ using Equation 6.21
    end for
    Calculate $F(x)$ using Equation 6.26
    Calculate $classLabel$ using Equation 6.27
    Return $\langle Predicted\_classLabel, Original\_classLabel \rangle$
  end for
end procedure

Algorithm 27 RBFN using hybrid learning technique based on Spark

Input: Training Set X, Testing Set X_{test}
Output: Classification result of X_{test}

procedure DRIVER_MAIN(X, X_{test})
  Create RDD of train data X, called $trainRDD$
  $trainRDD \leftarrow trainRDD$.cache( )
  Initialize and broadcast cluster centroids $Centroid_{old}$ to worker nodes
  $clusterPoints \leftarrow trainRDD$.map(KMEANS( )).FILTERDATA( ).collect( )
  $Centroid_{new} \leftarrow$ meanvalue($clusterPoints$)
  while $((Centroid_{old} - Centroid_{new})^2 \geq 1)$ **do**
    $clusterPoints \leftarrow trainRDD$.***map***(KMEANS( )).FILTERDATA( ).collect( )
    $Centroid_{new} \leftarrow$ meanvalue($clusterPoints$)
  end while
  Broadcast $Centroid_{new}$ to worker nodes
  Initialize and broadcast sf-RBFN weights (W) to worker nodes
  Initialize gradient ($gradVector$) as accumulator variable
  for $i = 1$ to $iterations$ **do**
    $gradVector \leftarrow trainRDD$.***map***(HLRBFNTRAIN( )).count( )
    Update W using $Mean(gradVector)$
    Restrict $W \in [-1, 1]$
    Broadcast updated W to worker nodes
  end for
  Create RDD of test data X_{test}, called $testRDD$
  $Output \leftarrow testRDD$.***map***(RBFNTEST( )).***collect***( )
  Calculate accuracy of sf-RBFN model
end procedure

learning technique i.e., Equation 6.25. The centers of training set are optimized using g_h in Equation 6.28. The g_h in Equation 6.28 is the center gradient of each class. The $grad_{center}$ in Equation 6.29 is a list of center gradient values obtained for each class. This gradient list ($grad_{center}$) is for a single sample, which is accumulated for all samples in cluster nodes. $gradVector_c$ is the accumulator variable which sums the center gradient value of all samples

($\sum_{i=1}^{M} grad_{center}$) in a distributed environment. The final gradient ($Mean(gradVector_c)$) is equal to the mean of all sample gradients ($\frac{1}{M}\sum_{i=1}^{M} gradVector_c$) obtained in each epoch. The centers are updated using Equation 6.30.

$$g_h = \left[\frac{\lambda}{M}\Phi_i \cdot (\Phi_i \cdot W_i - Y_i) \cdot W_i \cdot \left\{\sum_{j=1}^{h}(x_j - c_{ij})\right\}\right] \tag{6.28}$$

$$grad_{center} = \{g_1, g_2, ..., g_h\} \tag{6.29}$$

where λ is the regularization parameter, Φ_i and W_i are the basis function and weight vector for i^{th} class respectively, h denotes the number of centers, i.e., number of hidden nodes in hidden layer and k is the number of classes. The predicted output of samples and their corresponding class labels are obtained in a similar manner as in hybrid learning technique i.e., Equations 6.26 and 6.27 respectively.

$$C_{new} = C_{old} - Mean(gradVector_c) \tag{6.30}$$

The proposed sf-RBFN algorithm using gradient descent learning technique consists of driver program and various RDD Transformation methods, i.e., GDRBFNTRAIN( ) and RBFNTEST( ). Figure 6.6 shows the workflow of sf-RBFN algorithm using gradient descent learning technique.

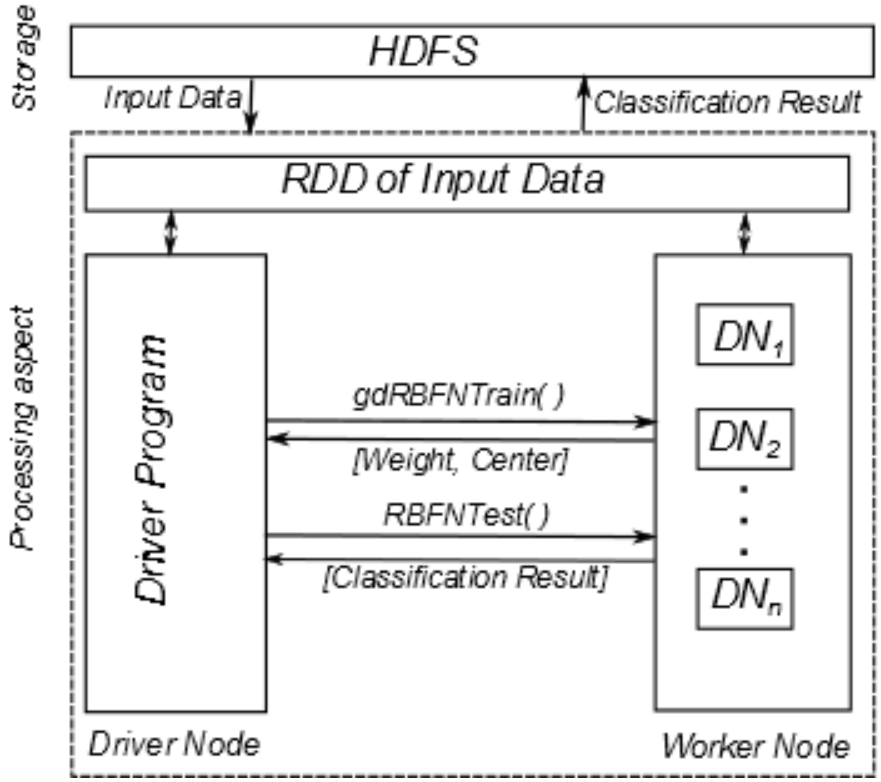


Figure 6.6: Workflow of sf-RBFN algorithm using gradient descent learning technique

Algorithm 28 shows the implementation of GDRBFNTRAIN( ) method. This method is

used for optimizing the weights (W) and centers (C) of sf-RBFN model by calculating $gradVector$ and $gradVector_c$ respectively in one single iteration. In driver program $gradVector$ and $gradVector_c$ are initialized as accumulator variables that are common to all worker nodes. The value of these variables can only be read within the driver program. The worker nodes can only update the accumulator variables. The $gradVector$ and $gradVector_c$ values obtained in each iteration are used to update W and C respectively, which are then broad-casted to worker nodes. This process continues until the given iteration criteria is met. The final weights (W) and centers (C) obtained gives the decision boundary parameters for multi-class classification problem. The transformation method for calculating the accuracy of sf-RBFN using gradient descent learning technique is same as Algorithm 26. Algorithm

Algorithm 28 Transformation method for training RBFN Model using gradient descent learning technique

Input: Training Set X
Output: $gradVector$, $gradVector_c$

```
1:  procedure GDRBFNTRAIN(X)
2:      Create RDD of train data X, called trainRDD
3:      for each sample x in trainRDD do                    ▷ Runs in parallel
4:          for each c in cluster centroids do
5:              Calculate φ using Equation 6.21
6:          end for
7:          for each class k do
8:              Calculate J_k using Equation 6.22
9:              Calculate g_k using Equation 6.23
10:             for each center h do
11:                 Calculate g_h using Equation 6.28
12:             end for
13:         end for
14:         Calculate grad_list using Equation 6.24
15:         Calculate grad_center using Equation 6.29
16:         gradVector += grad_list
17:         gradVector_c += grad_center
18:     end for
19:     Return gradVector, gradVector_c
20: end procedure
```

29 shows the implementation of sf-RBFN using gradient descent learning technique in driver program, along with RDD transformations and actions.

6.4 Results and Interpretation

In this section, the results obtained from the proposed algorithms (Section 6.3) on various microarray high-dimensional datasets like GSE13159, GSE13204, and GSE15061 (described in Section 4.2) are discussed. The performance of the classifier is measured using

Algorithm 29 RBFN using gradient descent learning technique based on Spark

Input: Training Set X, Testing Set X_{test}
Output: Classification result of X_{test}

procedure DRIVER_MAIN(X, X_{test})
  Create RDD of train data X, called $trainRDD$
  $trainRDD \leftarrow trainRDD$.cache( )
  Initialize and broadcast sf-RBFN weights (W) and centers (C) to worker nodes
  Initialize $gradVector$ and $gradVector_c$ as accumulator variable
  for $i = 1$ to $iterations$ **do**
    [$gradVector$, $gradVector_c$] $\leftarrow trainRDD$.*map*(GDRBFNTRAIN( )).collect( )
    Update W using $Mean(gradVector)$
    Update C using $Mean(gradVector_c)$
    Restrict $W \in [-1, 1]$
    Broadcast updated W and C to worker nodes
  end for
  Create RDD of test data X_{test}, called $testRDD$
  $Output \leftarrow testRDD$.map(RBFNTEST( )).collect( )
  Calculate accuracy of sf-RBFN model
end procedure

the various parameters like accuracy, precision, recall, and processing efficiency [158].

6.4.1 Analysis of Spark based classifiers

After selecting relevant features from training dataset set, the proposed Spark based classifiers are applied to classify the microarray datasets with reduced number of features. When the samples are sequentially selected for training purpose, the model designed may be over-trained or under-trained, as the selected samples may belong to similar classes or groups, i.e., only cancerous or non-cancerous samples. To avoid this, '3-fold cross validation (CV)' technique has been applied to assess the performance of classifiers, as it provides a more realistic assessment by generalizing significantly to unseen data. In CV technique, the parameters of classifiers are initially tuned within a certain range and an optimal value is selected for each fold. The median value of the best parameter from each fold is considered for training the final model. The performance of the final model is tested using the testing dataset. Cross-Validation using the train dataset provides the training accuracy of the model, and performance evaluation using test dataset gives the testing accuracy of the model.

6.4.1.1 Result of sf-LoR classifier

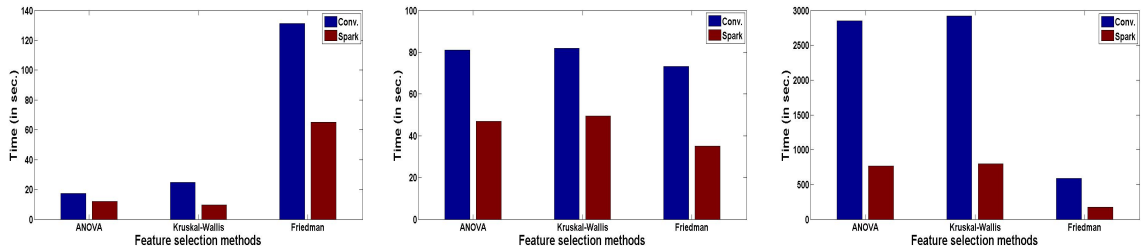
Logistic Regression based on Spark (sf-LoR) classifier is used with OVA to classify multiple class problem. 3-fold Cross-Validation is applied on training dataset in order to validate the classifier. The best regularization parameter for each fold is obtained by varying it within a certain range. The median value of the best regularization parameter from each fold is considered for the final model. The performance of the final model is tested using the testing

dataset. Cross-Validation using the train dataset gives the training accuracy of the model, and performance evaluation using the test dataset gives the testing accuracy of the model.

The execution time and processing efficiency of sf-LoR (in testing phase) based on Spark framework and on conventional machine are tabulated in Table 6.1. Processing efficiency in Table 6.1 is defined as the number of samples processed per second. The amount of time consumed by Logistic Regression classifier on Spark and conventional system is shown in Figure 6.7. From Figure 6.7, it is evident that the time taken by the classifiers based on Spark framework, to analyze the datasets is much less as compared to the same in conventional system. The performance parameters of Spark based Logistic Regression classifier, i.e., Recall, Precision, F-Measure and Accuracy for dataset GSE13159, GSE13204 and GSE15061 are mentioned in Table 6.2, 6.3, and 6.4 respectively.

Table 6.1: Execution time and processing efficiency result of Logistic Regression (sf-LoR) classfier

FS method	Dataset	Conv. Timing (sec)	Spark Timing (sec)	Conv. Processing efficiency (s^{-1})	Spark Processing efficiency (s^{-1})
ANOVA	**GSE13159**	2851.29	766.74	0.24	0.91
	GSE13204	80.99	46.84	13.37	23.12
	GSE15061	17.47	11.97	16.59	24.22
Kruskal-Wallis	**GSE13159**	2920.61	799.89	0.23	0.87
	GSE13204	81.88	49.40	13.22	21.92
	GSE15061	24.75	9.54	11.71	30.39
Friedman	**GSE13159**	589.22	173.50	1.18	4.02
	GSE13204	73.17	35.06	14.80	30.88
	GSE15061	131.21	65.18	2.21	4.44



(a) GSE15061 (b) GSE13204 (c) GSE13159

Figure 6.7: Comparison in terms of execution time, between Spark and conventional system for Logistic Regression (sf-LoR) classifier (in testing phase)

6.4.1.2 Result of sf-SVM classifier

SVM based on Spark (sf-SVM) classifier with linear kernel is used with OVA to classify multiple class problem. 3-fold Cross-Validation is applied on training dataset in order to validate the classifier. The best regularization parameter for each fold is obtained, by varying it within a certain range of $[2^{-5}, 2^{5}]$. The median value of the best regularization parameter

Table 6.2: Performance parameter of Logistic Regression for GSE13159 dataset

Feature Selection	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.7143	0.8947	0.6667	0.1250	0.2222	0.8120	0.2308	0.7500	0.7857	0.9732	0.7200	0.9130	0.6400	0.9130	0.9483	0.6341	0.7975	0.5000	0.6800
	Precision	0.7692	0.8095	1.0000	0.5000	0.5000	0.7787	0.6000	0.9000	0.7857	1.0000	0.8571	0.6117	0.5517	0.8750	0.9483	0.8667	0.7500	0.5000	0.7558
	F-Measure	0.7407	0.8500	0.8000	0.2000	0.3077	0.7950	0.3333	0.8182	0.7857	0.9864	0.7826	0.7326	0.5926	0.8936	0.9483	0.7324	0.7730	0.5000	0.6984
	Accuracy																			0.7897
Kruskal-Wallis	Recall	0.7143	0.8947	0.6667	0.1250	0.2222	0.8120	0.2308	0.7500	0.7857	0.9732	0.7200	0.8986	0.6800	0.8696	0.9310	0.6098	0.7595	1.0000	0.7024
	Precision	0.7692	0.8095	1.0000	0.5000	0.5000	0.7787	0.6000	0.9000	0.7857	0.9864	0.8571	0.6078	0.5484	0.8333	0.9474	0.8333	0.7500	0.6667	0.7596
	F-Measure	0.7407	0.8500	0.8000	0.2000	0.3077	0.7950	0.3333	0.8182	0.7857	0.9797	0.7826	0.7251	0.6071	0.8511	0.9391	0.7042	0.7547	0.8000	0.7097
	Accuracy																			0.7640
Friedman	Recall	0.7857	0.7895	0.6667	0.1875	0.3333	0.8034	0.3077	0.6667	0.7143	0.9664	0.6800	0.8696	0.7200	0.8261	0.9138	0.5854	0.7722	0.7500	0.6855
	Precision	0.7857	0.7895	0.8889	0.6000	0.4286	0.7966	0.6667	0.7273	0.7143	0.9730	0.8095	0.6122	0.5455	0.8261	0.9298	0.8000	0.7531	0.6000	0.7359
	F-Measure	0.7857	0.7895	0.7619	0.2857	0.3750	0.8000	0.4211	0.6957	0.7143	0.9697	0.7391	0.7186	0.6207	0.8261	0.9217	0.6761	0.7625	0.6667	0.6961
	Accuracy																			0.7540

Table 6.3: Performance parameter of Logistic Regression (sf-LoR) for GSE13204 dataset

Feature Selection	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.7600	0.9756	0.9333	0.6250	1.0000	0.9064	0.8333	0.9474	1.0000	1.0000	0.9250	0.7798	0.7500	1.0000	0.9524	0.8033	0.8788	0.8333	0.8835
	Precision	0.7917	0.8696	0.9333	0.8333	1.0000	0.8564	1.0000	1.0000	1.0000	0.9870	1.0000	0.7870	0.7674	0.9688	0.9524	0.9245	0.8406	1.0000	0.9173
	F-Measure	0.7755	0.9195	0.9333	0.7143	1.0000	0.8807	0.9091	0.9730	1.0000	0.9935	0.9610	0.7834	0.7586	0.9841	0.9524	0.8596	0.8593	0.9091	0.8981
	Accuracy																			0.7903
Kruskal-Wallis	Recall	0.7600	0.9756	0.9333	0.5417	1.0000	0.9064	0.8333	0.9474	1.0000	1.0000	0.9500	0.7798	0.7727	1.0000	0.9643	0.8033	0.8788	0.8333	0.8822
	Precision	0.7917	0.8696	0.9333	0.8125	1.0000	0.8564	1.0000	1.0000	1.0000	0.9870	1.0000	0.8019	0.7727	0.9688	0.9419	0.9245	0.8406	1.0000	0.9167
	F-Measure	0.7755	0.9195	0.9333	0.6500	1.0000	0.8807	0.9091	0.9730	1.0000	0.9935	0.9744	0.7907	0.7727	0.9841	0.9529	0.8596	0.8593	0.9091	0.8965
	Accuracy																			0.7812
Friedman	Recall	0.8000	0.9756	0.9333	0.4583	1.0000	0.9064	0.8333	0.9474	1.0000	1.0000	0.8750	0.8073	0.7500	0.9677	0.9524	0.8525	0.8485	0.8333	0.8745
	Precision	0.7143	0.8511	0.9333	0.7857	1.0000	0.8564	1.0000	1.0000	1.0000	0.9870	1.0000	0.8000	0.7174	0.9677	0.9639	0.9286	0.8421	1.0000	0.9082
	F-Measure	0.7547	0.9091	0.9333	0.5789	1.0000	0.8807	0.9091	0.9730	1.0000	0.9935	0.9333	0.8037	0.7333	0.9677	0.9581	0.8889	0.8453	0.9091	0.8873
	Accuracy																			0.7966

Table 6.4: Performance parameter of Logistic Regression (sf-LoR) for GSE15061 dataset

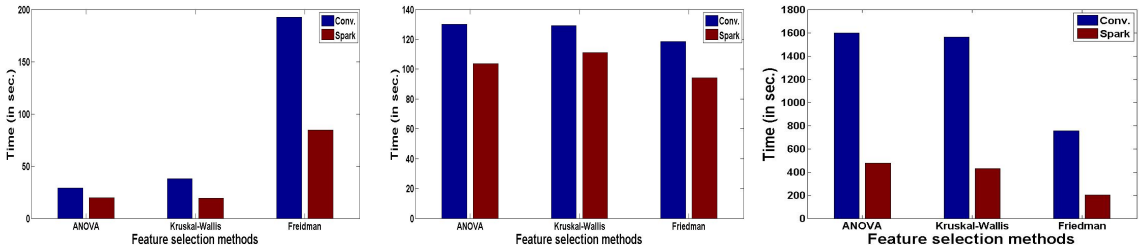
Feature Selection	Performance parameter	1	2	3	Average
ANOVA	Recall	0.9778	0.9541	0.3696	0.7672
	Precision	0.9565	0.7879	0.8500	0.8648
	F-Measure	0.9670	0.8631	0.5152	0.7818
	Accuracy				0.6824
Kruskal-Wallis	Recall	0.9778	0.4495	0.3478	0.5917
	Precision	0.6197	0.8305	0.8889	0.7797
	F-Measure	0.7586	0.5833	0.5000	0.6140
	Accuracy				0.6793
Friedman	Recall	0.9778	0.4404	0.3261	0.5814
	Precision	0.6197	0.8276	0.7895	0.7456
	F-Measure	0.7586	0.5749	0.4615	0.5983
	Accuracy				0.6724

from each fold is considered for the final model. The performance of the final model is tested using the testing dataset.

The execution time and processing efficiency of SVM classifier (in testing phase) based on Spark framework and on conventional machine are tabulated in Table 6.5. Processing efficiency in Table 6.5 is defined as the number of samples processed per second. The amount of time consumed by SVM classifier on Spark and conventional system is shown in Figure 6.8. From Figure 6.8 it is evident that the time taken by the classifiers based on Spark framework, to analyze the datasets is much less as compared to the same in conventional system. The performance parameters of Spark based SVM classifier, i.e., Recall, Precision, F-Measure and Accuracy for dataset GSE13159, GSE13204 and GSE15061 are mentioned in Tables 6.6, 6.7, and 6.8 respectively.

Table 6.5: Execution time and processing efficiency result of sf-SVM classfier (in testing phase)

FS method	Dataset	Conv. Time (sec.)	Spark Time (sec.)	Conv. Processing efficiency (s^{-1})	Spark processing efficiency (s^{-1})
ANOVA	GSE13159	1598.72	478.18	0.43	1.46
	GSE13204	129.96	103.78	8.33	10.43
	GSE15061	29.11	19.65	9.96	14.75
Kruskal-Wallis	GSE13159	1561.38	430.87	0.44	1.62
	GSE13204	129.23	111.18	8.38	9.74
	GSE15061	37.87	19.34	7.65	14.99
Friedman	GSE13159	756.53	203.77	0.92	3.43
	GSE13204	118.36	94.25	9.15	11.49
	GSE15061	192.75	84.46	1.50	3.43



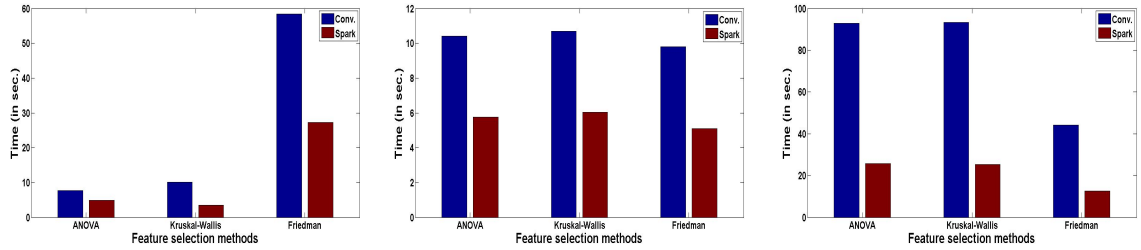
(a) GSE15061 (b) GSE13204 (c) GSE13159

Figure 6.8: Comparison in terms of execution time, between Spark and conventional system for sf-SVM classifier (in testing phase)

6.4.1.3 Result of sf-NB classifier

Naive Bayes classifer based on Spark (sf-NB) is used to classify multiple class problem. 3-fold Cross-Validation is applied on training dataset in order to validate the classifier. The performance of the final model is tested using the testing dataset.

The execution time and processing efficiency of Naive Bayes classifier based on Spark framework and on conventional machine are tabulated in Table 6.9. Processing efficiency in Table 6.9 is defined as the number of samples processed per second. The amount of time consumed by Naive Bayes classifier on Spark and conventional system is shown in Figure 6.9. From Figure 6.9, it is observed that the time taken by the classifiers based on Spark framework, to analyze the datasets is much less as compared to the same in conventional system. The performance parameters of Spark based Naive Bayes classifier, i.e., Recall, Precision, F-Measure and Accuracy for dataset GSE13159, GSE13204 and GSE15061 are mentioned in Table 6.10, 6.11, and 6.12 respectively.



(a) GSE15061 (b) GSE13204 (c) GSE13159

Figure 6.9: Comparison in terms of execution time, between Spark and conventional system for Naive Bayes (sf-NB) classifier (in testing phase)

6.4.1.4 Results of sf-KNN classifier

The proposed classifier is trained using 3-fold cross validation by varying the parameter $K \in [1, 21]$ with a span of 2 on the training dataset. The optimal values of K are obtained in each fold, corresponding to the training accuracy of that fold. Finally, the average accuracy

Table 6.6: Performance parameter of sf-SVM for GSE13159 dataset

Feature Selection	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.7857	0.8947	0.7500	0.1250	0.4445	0.8804	0.3077	0.8334	0.7857	1.0000	0.7200	0.9131	0.6400	0.9131	0.9483	0.6342	0.8102	0.7500	0.7298
	Precision	0.8462	0.8095	1.0000	0.5000	1.0000	0.8110	1.0000	1.0000	0.9167	1.0000	0.9000	0.6429	0.5517	0.9131	0.9483	0.8667	0.7711	0.6000	0.8376
	F-Measure	0.8148	0.8500	0.8571	0.2000	0.6154	0.8443	0.4706	0.9091	0.8462	1.0000	0.8000	0.7545	0.5926	0.9130	0.9483	0.7324	0.7901	0.6667	0.7558
	Accuracy																			0.8383
Kruskal-Wallis	Recall	0.7857	0.7894	0.5833	0.1250	0.4444	0.8803	0.3077	0.8333	0.7857	1.0000	0.7200	0.9130	0.6400	0.9130	0.9484	0.6341	0.8101	0.7500	0.7146
	Precision	0.8462	0.78941	1.0000	0.5000	0.8000	0.8047	1.0000	1.0000	0.9167	1.0000	0.9000	0.6429	0.5517	0.8750	0.9483	0.8667	0.7619	0.6000	0.8224
	F-Measure	0.8148	0.7894	0.7368	0.2000	0.5714	0.8408	0.4706	0.9091	0.8461	1.0000	0.8000	0.7545	0.5925	0.8936	0.9482	0.7323	0.7852	0.6667	0.7418
	Accuracy																			0.8326
Friedman	Recall	0.7857	0.7894	0.5833	0.1250	0.4444	0.8803	0.3076	0.8333	0.7857	1.0000	0.7200	0.9143	0.6400	0.9130	0.9483	0.6341	0.8101	0.7500	0.7146
	Precision	0.8461	0.7894	1.0000	0.5000	0.8000	0.8046	1.0000	1.0000	0.9167	1.0000	0.9000	0.6429	0.5517	0.8750	0.9483	0.8667	0.7619	0.6000	0.8224
	F-Measure	0.8148	0.7895	0.7368	0.2000	0.5714	0.8408	0.4706	0.9091	0.8461	1.0000	0.8000	0.7545	0.5925	0.8936	0.9482	0.7324	0.7853	0.6667	0.7418
	Accuracy																			0.8326

Table 6.7: Performance parameter of sf-SVM for GSE13204 dataset

Feature Selection	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.3200	0.8780	0.2667	0.1667	0.3125	0.9298	0.1667	0.2632	0.2632	0.9605	0.8500	0.8624	0.1136	0.9032	0.9762	0.5902	0.9091	0.3333	0.5592
	Precision	1.0000	0.8182	0.8000	0.8000	0.5000	0.6974	0.6000	0.4545	0.5556	0.9821	0.8293	0.6528	0.8333	1.0000	0.9318	0.9000	0.6452	1.0000	0.7778
	F-Measure	0.4848	0.8471	0.4000	0.2759	0.3846	0.7970	0.2609	0.3333	0.3571	0.9712	0.8395	0.7431	0.2000	0.9492	0.9535	0.7129	0.7547	0.5000	0.5980
	Accuracy																			0.7839
Kruskal-Wallis	Recall	0.2000	0.8537	0.2667	0.1667	0.4375	0.8596	0.2222	0.3158	0.3684	0.9518	0.6750	0.8165	0.1818	0.8065	0.9405	0.6393	0.8788	0.6667	0.5693
	Precision	0.3846	0.7609	0.4000	0.5714	0.4118	0.7277	0.4000	0.4000	0.6364	0.9731	0.7714	0.6357	0.6154	0.8065	0.8876	0.8478	0.6824	0.8000	0.6507
	F-Measure	0.2632	0.8046	0.3200	0.2581	0.4242	0.7882	0.2857	0.3529	0.4667	0.9623	0.7200	0.7149	0.2807	0.8065	0.9133	0.7290	0.7682	0.7273	0.5881
	Accuracy																			0.7599
Friedman	Recall	0.3200	0.8537	0.4667	0.1667	0.4375	0.8596	0.2222	0.3158	0.3684	0.9518	0.6750	0.8165	0.1818	0.8065	0.9167	0.6066	0.8561	0.3333	0.5642
	Precision	0.4706	0.8140	0.5385	0.5714	0.4118	0.7277	0.4000	0.4000	0.6364	0.9731	0.7714	0.6357	0.6154	0.7813	0.8750	0.7708	0.6807	0.6667	0.6522
	F-Measure	0.3810	0.8333	0.5000	0.2581	0.4242	0.7882	0.2857	0.3529	0.4667	0.9623	0.7200	0.7149	0.2807	0.7937	0.8953	0.6789	0.7584	0.4444	0.5855
	Accuracy																			0.7572

Table 6.8: Performance parameter of sf-SVM for GSE15061 dataset

Feature Selection	Performance parameter	1	2	3	Average
ANOVA	Recall	0.9852	0.9633	0.2391	0.7292
	Precision	0.9708	0.7500	0.8462	0.8557
	F-Measure	0.9779	0.8434	0.3729	0.7314
	Accuracy				0.8586
Kruskal-Wallis	Recall	0.9778	0.4587	0.2609	0.5658
	Precision	0.6197	0.7937	0.8571	0.7568
	F-Measure	0.7586	0.5814	0.4000	0.5800
	Accuracy				0.6690
Friedman	Recall	0.9926	0.4220	0.3478	0.5875
	Precision	0.6175	0.8679	0.8000	0.7618
	F-Measure	0.7614	0.5679	0.4848	0.6047
	Accuracy				0.6759

Table 6.9: Execution time and processing efficiency result of Naive Bayes classifier (sf-NB) (in testing phase)

FS Technique	Dataset	Conv. Time (sec)	Spark Time (sec)	Conv. Processing efficiency (s^{-1})	Spark Processing efficiency (s^{-1})
ANOVA	GSE13159	92.99	25.66	7.51	27.24
	GSE13204	10.41	5.76	104.03	188.02
	GSE15061	7.71	4.84	37.61	59.91
Kruskal-Wallis	GSE13159	93.32	25.33	7.49	27.59
	GSE13204	10.67	6.05	101.49	179.00
	GSE15061	10.15	3.50	28.57	82.85
Friedman	GSE13159	44.21	12.57	15.81	55.60
	GSE13204	9.80	5.09	110.51	212.77
	GSE15061	58.35	27.21	4.97	10.65

value is considered for each fold. The *average accuracy* value is the training accuracy of the model with the optimal K (median values of K in each fold). Using the optimal K, the model is tested on the test data and the *testing accuracy* is obtained.

6.4.1.4.1 Results for the GSE15061 Dataset This dataset is a subset of the Microarray Innovations In LEukemia (MILE) study program stored in the NCBI repository with accession number GSE15061. There are 870 samples in the GSE15061 dataset, of which 580 samples are selected as training samples and 290 as testing samples. These samples are divided into 3 classes and are labeled as shown in Table 4.7.

The model sf-KNN is trained using the training data of various feature selection (FS) methods by varying the value of $K \in [1, 21]$ with a span of 2 and training accuracies are obtained. The training accuracies obtained using ANOVA, Kruskal-Wallis, and Friedman

Table 6.10: Performance parameter of Naive Bayes (sf-NB) for GSE13159 dataset

Feature Selection	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.7857	0.8947	0.6667	0.1250	0.2222	0.8547	0.2308	0.7500	0.7857	1.0000	0.7200	0.9130	0.6400	0.9130	0.9483	0.6341	0.8101	0.7500	0.7025
	Precision	0.8462	0.8095	1.0000	0.6667	1.0000	0.7874	1.0000	1.0000	0.9167	1.0000	0.9000	0.6176	0.5517	0.8750	0.9483	0.8667	0.7619	0.6000	0.8415
	F-Measure	0.8148	0.8500	0.8000	0.2105	0.3636	0.8197	0.3750	0.8571	0.8462	1.0000	0.8000	0.7368	0.5926	0.8936	0.9483	0.7324	0.7853	0.6667	0.7274
	Accuracy																			0.8269
Kruskal-Wallis	Recall	0.7857	0.8947	0.6667	0.1250	0.2222	0.8120	0.2308	0.7500	0.7857	1.0000	0.7200	0.9130	0.6400	0.9130	0.9483	0.6341	0.8101	0.7500	0.7001
	Precision	0.8462	0.8095	1.0000	0.6667	0.6667	0.7787	0.6000	0.9000	0.9167	1.0000	0.8571	0.6176	0.5517	0.8750	0.9483	0.8667	0.7619	0.6000	0.7924
	F-Measure	0.8148	0.8500	0.8000	0.2105	0.3333	0.7950	0.3333	0.8182	0.8462	1.0000	0.7826	0.7368	0.5926	0.8936	0.9483	0.7324	0.7853	0.6667	0.7189
	Accuracy																			0.8197
Friedman	Recall	0.7857	0.8947	0.6667	0.1250	0.2222	0.8547	0.2308	0.7500	0.7857	1.0000	0.7200	0.9130	0.6400	0.9130	0.9483	0.6341	0.8101	0.7500	0.7025
	Precision	0.8462	0.8095	1.0000	0.6667	1.0000	0.7874	1.0000	1.0000	0.9167	1.0000	0.9000	0.6176	0.5517	0.8750	0.9483	0.8667	0.7619	0.6000	0.8415
	F-Measure	0.8148	0.8500	0.8000	0.2105	0.3636	0.8197	0.3750	0.8571	0.8462	1.0000	0.8000	0.7368	0.5926	0.8936	0.9483	0.7324	0.7853	0.6667	0.7274
	Accuracy																			0.8269

Table 6.11: Performance parameter of Naive Bayes (sf-NB) for GSE13204 dataset

Feature Selection	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.9200	0.9756	0.9333	0.6667	1.0000	0.7485	0.7778	0.9474	1.0000	0.9825	0.9250	0.7890	0.7727	0.9677	0.9286	0.7705	0.6667	1.0000	0.8762
	Precision	0.5610	0.7547	0.8235	0.5714	0.8889	0.9275	0.5385	1.0000	1.0000	0.9956	0.7872	0.7890	0.5763	0.9677	0.9512	0.8545	0.8544	0.4286	0.7928
	F-Measure	0.6970	0.8511	0.8750	0.6154	0.9412	0.8285	0.6364	0.9730	1.0000	0.9890	0.8506	0.7890	0.6602	0.9677	0.9398	0.8103	0.7489	0.6000	0.8207
	Accuracy																			0.8476
Kruskal-Wallis	Recall	0.9200	0.9756	0.9333	0.6667	1.0000	0.7485	0.7222	0.9474	1.0000	0.9825	0.9250	0.7890	0.7727	0.9677	0.9286	0.7705	0.6667	1.0000	0.8731
	Precision	0.5610	0.7547	0.8235	0.5714	0.8889	0.9209	0.5200	1.0000	1.0000	0.9956	0.7872	0.7890	0.5763	0.9677	0.9512	0.8545	0.8544	0.4286	0.7914
	F-Measure	0.6970	0.8511	0.8750	0.6154	0.9412	0.8258	0.6047	0.9730	1.0000	0.9890	0.8506	0.7890	0.6602	0.9677	0.9398	0.8103	0.7489	0.6000	0.8188
	Accuracy																			0.8467
Friedman	Recall	0.9200	0.9756	0.9333	0.6667	1.0000	0.7485	0.7222	0.9474	1.0000	0.9825	0.9250	0.7982	0.7727	0.9677	0.9524	0.7869	0.6742	0.8333	0.8670
	Precision	0.5897	0.7692	0.7368	0.6154	0.8889	0.9209	0.5200	1.0000	1.0000	0.9956	0.8222	0.7699	0.5965	0.9677	0.9524	0.8571	0.8641	0.3571	0.7902
	F-Measure	0.7188	0.8602	0.8235	0.6400	0.9412	0.8258	0.6047	0.9730	1.0000	0.9890	0.8706	0.7838	0.6733	0.9677	0.9524	0.8205	0.7574	0.5000	0.8168
	Accuracy																			0.8504

Table 6.12: Performance parameter of Naive Bayes (sf-NB) for GSE15061 dataset

Feature Selection	Performance parameter	1	2	3	Average
ANOVA	**Recall**	0.9111	0.3670	0.2609	0.5130
	Precision	0.6029	0.7273	0.3871	0.5724
	F-Measure	0.7257	0.4878	0.3117	0.5084
	Accuracy				0.6034
Kruskal-Wallis	**Recall**	0.9037	0.3761	0.2826	0.5208
	Precision	0.6040	0.7455	0.3939	0.5811
	F-Measure	0.7240	0.5000	0.3291	0.5177
	Accuracy				0.6069
Friedman	**Recall**	0.9111	0.5780	0.2391	0.5761
	Precision	0.6949	0.7500	0.3793	0.6081
	F-Measure	0.7885	0.6528	0.2933	0.5782
	Accuracy				0.6793

tests as feature selection methods are 77.41% ($f = 6786, K = 21$), 82.24% ($f = 9741, K = 17$), and 79.83% ($f = 5431, K = 21$), respectively. Corresponding to these training accuracies, the model is tested using the optimal value of K and the testing accuracies obtained using ANOVA, Kruskal-Wallis, and Friedman tests as feature selection methods are 71.70%, 73.40%, and 73.10%, respectively. The performance parameters of sf-KNN classifier, i.e., Recall, Precision, F-Measure, Accuracy are mentioned in Table 6.13.

Table 6.13: Performance parameter of sf-KNN with various feature selection methods using GSE15061 dataset.

FS method	Performance parameter	1	2	3	Average
ANOVA	**Recall**	0.6963	0.8349	0.5000	0.6771
	Precision	0.9691	0.6233	0.4894	0.6939
	F-Measure	0.8103	0.7137	0.4946	0.6729
	Accuracy				0.7172
Kruskal-Wallis	**Recall**	0.7556	0.9083	0.2609	0.6416
	Precision	0.9189	0.6111	0.7059	0.7453
	F-Measure	0.8293	0.7306	0.3810	0.6469
	Accuracy				0.7345
Friedman	**Recall**	0.7778	0.8716	0.2609	0.6367
	Precision	0.8824	0.6169	0.7059	0.7350
	F-Measure	0.8268	0.7224	0.3810	0.6434
	Accuracy				0.7310

6.4.1.4.2 Results for the GSE13159 Dataset There are 2,096 samples in the GSE13159 dataset, of which 1,397 samples are selected as training samples and 699 as testing samples. These samples are divided into 18 classes and are labeled as shown in Table 4.5.

The model sf-KNN is trained using training data by varying the value of $K \in [1, 21]$ with a span of 2 with various feature selection (FS) methods, and training accuracies are obtained. The training accuracies obtained using ANOVA, Kruskal-Wallis, and Friedman tests as feature selection methods are 81.60% $(f = 37016, K = 19)$, 80.60% $(f = 36897, K = 21)$, and 81.60% $(f = 17553, K = 19)$ respectively. Corresponding to these training accuracies, the model is tested using the optimal values of K and the testing accuracies obtained using ANOVA, Kruskal-Wallis, and Friedman test as feature selection methods are 81.12%, 80.97%, and 80.40%, respectively. The performance parameters of sf-KNN classifier, i.e., Recall, Precision, F-Measure, Accuracy are mentioned in Table 6.14.

6.4.1.4.3 Results for the GSE13204 Dataset There are 3,248 samples in the GSE13204 dataset, of which 2,165 samples are selected as training samples and 1,083 as testing samples. These samples are divided into 18 classes and are labeled as shown in Table 4.6.

The model sf-KNN is trained using training data from various FS methods by varying the value of $K \in [1, 21]$ with a span of 2 to obtain training accuracies. The training accuracies obtained using ANOVA, Kruskal-Wallis, and Friedman tests as feature selection methods are 87.90% $(f = 1423, K = 7)$, 92.52% $(f = 1427, K = 7)$, and 92.33% $(f = 1225, K = 3)$, respectively. Corresponding to these training accuracies, the model is tested using the optimal values of K, and the testing accuracies obtained using ANOVA, Kruskal-Wallis, and Friedman tests as feature selection methods are 83.75%, 83.93%, and 84.76%, respectively. The performance parameters of sf-KNN classifier, i.e., Recall, Precision, F-Measure, Accuracy are mentioned in Table 6.15.

Table 6.16 summarizes the training and testing accuracy of sf-KNN with various FS methods using different microarray datasets like GSE15061, GSE13159, and GSE13204.

From the obtained results, it is inferred that feature selection plays an important role in the classification process, and results vary by changing the feature selection methods.

The overall execution details i.e., the maximum time taken by Spark (T_{max}^{Spark}) , the maximum time taken by conventional system (T_{max}^{Conv}) and their corresponding processing efficiency of the sf-KNN classifier are tabulated in Table 6.17. Conventional time (T_{max}^{Conv}) is defined as the time taken by a conventional system where data is not distributed over different machines, i.e., data is stored on a single machine, and a dataset is sequentially processed.

The amount of time consumed by the sf-KNN classifier on the Hadoop cluster with Spark and on a conventional system is shown in Figure 6.10. From Figure 6.10, it is evident that the time taken by the sf-KNN classifier to analyze the datasets GSE13159, GSE13204, and GSE15061 (where data size is large), and the time taken by sf-KNN on the Spark is much less than on a conventional system. Hence, Spark provides effective performance in analyzing

Table 6.14: Performance parameter of sf-KNN with various feature selection methods using GSE13159 dataset.

FS method	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.8571	0.8947	0.9167	0.5000	0.8889	0.7009	0.6154	0.8333	0.9286	1.0000	0.9600	0.7101	0.6800	0.8696	0.9483	0.6829	0.6709	0.7500	0.8004
	Precision	0.6667	0.7083	0.7333	0.3200	1.0000	0.8542	0.4444	1.0000	1.0000	1.0000	0.7273	0.6806	0.5313	0.8696	0.9483	0.7568	0.8154	1.0000	0.7809
	F-Measure	0.7500	0.7907	0.8148	0.3902	0.9412	0.7700	0.5161	0.9091	0.9630	1.0000	0.8276	0.6950	0.5965	0.8696	0.9483	0.7179	0.7361	0.8571	0.7830
	Accuracy																			0.8112
Kruskal-Wallis	Recall	0.8571	0.8947	0.8333	0.5000	1.0000	0.6923	0.6154	0.8333	0.9286	1.0000	0.9600	0.7101	0.6800	0.8696	0.9483	0.6829	0.6582	1.0000	0.8147
	Precision	0.6316	0.7391	0.6667	0.3200	1.0000	0.8710	0.4211	1.0000	1.0000	1.0000	0.7273	0.6806	0.5313	0.8696	0.9483	0.7568	0.8000	1.0000	0.7757
	F-Measure	0.7273	0.8095	0.7407	0.3902	1.0000	0.7714	0.5000	0.9091	0.9630	1.0000	0.8276	0.6950	0.5965	0.8696	0.9483	0.7179	0.7222	1.0000	0.7882
	Accuracy																			0.8097
Friedman	Recall	0.9286	0.9474	0.9167	0.5625	1.0000	0.6068	0.6154	0.8333	0.9286	1.0000	0.9200	0.7101	0.6800	0.9130	0.9483	0.7073	0.6709	1.0000	0.8272
	Precision	0.6500	0.7826	0.7333	0.3214	1.0000	0.8659	0.3077	1.0000	0.9286	1.0000	0.6970	0.7000	0.5152	0.8750	0.9483	0.7632	0.8548	0.8000	0.7635
	F-Measure	0.7647	0.8571	0.8148	0.4091	1.0000	0.7136	0.4103	0.9091	0.9286	1.0000	0.7931	0.7050	0.5862	0.8936	0.9483	0.7342	0.7518	0.8889	0.7838
	Accuracy																			0.8040

Table 6.15: Performance parameter of sf-KNN with various feature selection methods using GSE13204 dataset.

FS method	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.8800	0.9756	0.9333	0.3333	1.0000	0.8070	0.5000	0.9474	0.9474	1.0000	0.9000	0.6881	0.8409	0.9677	0.9405	0.7213	0.6970	0.5000	0.8100
	Precision	0.5946	0.7407	0.8750	0.8000	0.8421	0.8961	1.0000	1.0000	1.0000	0.9870	0.8571	0.6944	0.4353	0.9375	0.9405	0.8627	0.8214	1.0000	0.8491
	F-Measure	0.7097	0.8421	0.9032	0.4706	0.9143	0.8492	0.6667	0.9730	0.9730	0.9935	0.8780	0.6912	0.5736	0.9524	0.9405	0.7857	0.7541	0.6667	0.8076
	Accuracy																			0.8375
Kruskal-Wallis	Recall	0.8400	0.9756	0.9333	0.3333	1.0000	0.8129	0.5000	0.9474	0.9474	1.0000	0.9250	0.6789	0.8409	0.9677	0.9405	0.7541	0.6970	0.5000	0.8108
	Precision	0.6000	0.7407	0.8750	0.8000	0.8421	0.8910	1.0000	1.0000	1.0000	0.9870	0.8605	0.7048	0.4353	0.9375	0.9405	0.8679	0.8214	1.0000	0.8502
	F-Measure	0.7000	0.8421	0.9032	0.4706	0.9143	0.8502	0.6667	0.9730	0.9730	0.9935	0.8916	0.6916	0.5736	0.9524	0.9405	0.8070	0.7541	0.6667	0.8091
	Accuracy																			0.8393
Friedman	Recall	0.8400	0.9268	0.9333	0.3333	1.0000	0.8012	0.5556	0.9474	0.9474	1.0000	0.9250	0.6881	0.8409	0.9677	0.9643	0.8033	0.7273	0.8333	0.8353
	Precision	0.6000	0.8085	0.8750	0.6154	0.8421	0.8954	1.0000	1.0000	1.0000	0.9870	0.8222	0.7426	0.4684	0.9375	0.9310	0.8750	0.8276	0.7143	0.8301
	F-Measure	0.7000	0.8636	0.9032	0.4324	0.9143	0.8457	0.7143	0.9730	0.9730	0.9935	0.8706	0.7143	0.6016	0.9524	0.9474	0.8376	0.7742	0.7692	0.8211
	Accuracy																			0.8476

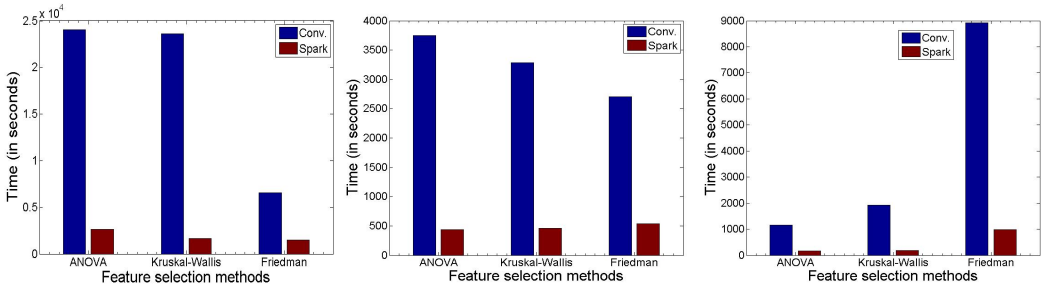
Table 6.16: Summary of training and testing accuracy (%) for various microarray datasets.

Dataset	ANOVA		Kruskal-Wallis		Friedman	
	Train Acc.	Test Acc.	Train Acc.	Test Acc.	Train Acc.	Test Acc.
GSE15061	77.41	71.72	82.24	73.45	79.83	73.10
GSE13159	81.60	81.12	80.60	80.97	81.60	80.40
GSE13204	87.90	83.75	92.52	83.93	92.33	84.76

Table 6.17: Execution details of sf-KNN classifier on Hadoop cluster (with three slaves) using Spark, and conventional system (Time is measured in seconds (s))

Dataset	Feature selection method	T_{max}^{Conv}	T_{max}^{Spark}	Processing efficiency (Conv. (s^{-1}))	Processing efficiency (Spark (s^{-1}))
GSE13159	**ANOVA**	24050	2643	1.54	14.01
	Kruskal-Wallis	23610	1676	1.56	22.02
	Friedman	6554	1500	2.68	11.73
GSE13204	**ANOVA**	3752	435	0.38	3.27
	Kruskal-Wallis	3286	457	0.43	3.123
	Friedman	2704	535	0.45	2.29
GSE15061	**ANOVA**	1151	167	5.9	40.64
	Kruskal-Wallis	1929	187	5.05	52.09
	Friedman	8918	985	6.09	55.15

large datasets.



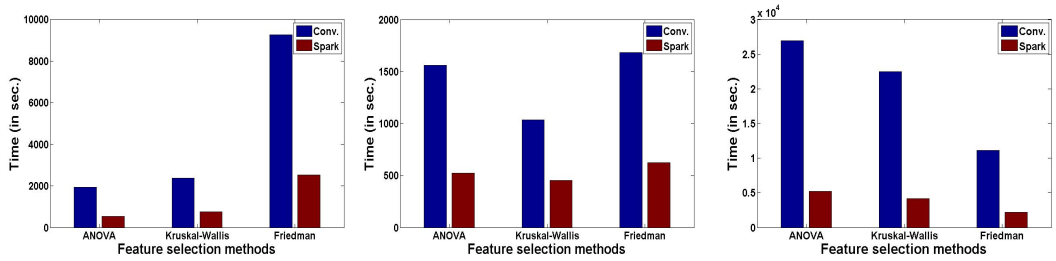
(a) GSE13159 (b) GSE13204 (c) GSE15061

Figure 6.10: Comparison of execution time in sf-KNN on Hadoop cluster (with Spark) and Conventional system (Conv).

6.4.1.5 Results of sf-ANN

In sf-ANN the weights are optimized using gradient descent algorithm. The total number of hidden nodes in the model act as the tuning parameter, which is varied within the selected range $[10, 30, 50, 90, 110, 150, 200, 300, 400]$. The value of learning rate α and momentum factor η are taken as 0.5 and 0.9 respectively. The model is trained using 3-fold CV by varying the number of hidden nodes and the overall performance is assessed. The overall execution details and processing efficiency of sf-ANN classifier is tabulated in Table 6.18.

Processing efficiency in Table 6.18 signifies the number of samples processed per second. The amount of time consumed by ANN classifier on Spark and conventional system is shown in Figure 6.11. From this figure, it is evident that the time taken to analyze the datasets by ANN classifier based on Spark framework (sf-ANN) is much less as compared to the same in conventional system. The performance parameters of sf-ANN classifier, i.e., Recall, Precision, F-Measure and Accuracy for dataset GSE15061, GSE13204, and GSE13159 are mentioned in Table 6.19, 6.20, and 6.21 respectively.



(a) GSE15061 (b) GSE13204 (c) GSE13159

Figure 6.11: Comparison in terms of execution time, between Spark and Conventional system (Conv.) for sf-ANN classifier (in testing phase)

6.4.1.6 Results of sf-RBFN using hybrid learning technique

In sf-RBFN (hybrid) K-Means clustering algorithm is applied to obtain the cluster centers, which are also the total number of hidden nodes in the model. The weights are optimized by using gradient descent algorithm. In this study the number of hidden nodes is taken as 25 and the regularization parameter λ (Section 6.3.8) which also act as the tuning parameter is varied within the range $\lambda \in [2^{-10}, 2^{5}]$ (with step size = 1 for $log\ \lambda$). The model is trained using 3-fold CV by varying λ and the overall performance is assessed. The overall execution details and processing efficiency of sf-RBFN classifier using hybrid learning technique is tabulated in Table 6.22. Processing efficiency in Table 6.22 signifies the number of samples processed per second. The amount of time consumed by RBFN (hybrid) classifier on Spark and conventional system is shown in Figure 6.12. From this figure, it is evident that the time taken to analyze the datasets by RBFN (hybrid) classifier based on Spark framework (sf-RBFN) is much less as compared to the same in conventional system. The performance parameters of sf-RBFN classifier using hybrid learning technique, i.e., Recall, Precision, F-Measure and Accuracy for dataset GSE15061, GSE13204, and GSE13159 are mentioned in Table 6.23, 6.24, and 6.25 respectively.

6.4.1.7 Results of sf-RBFN using gradient descent learning technique

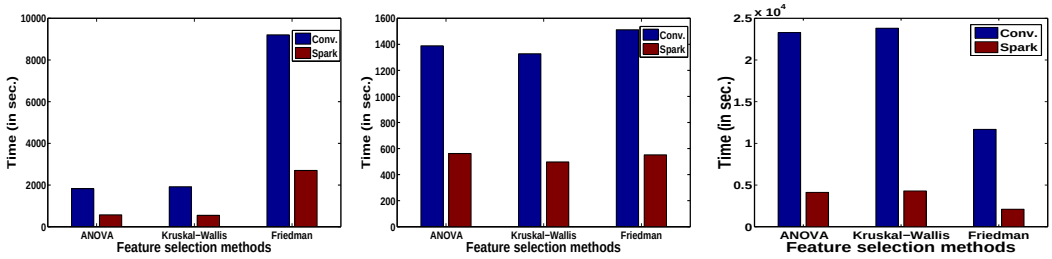
In sf-RBFN (gradient descent), both the centers and the weights are optimized by using gradient descent algorithm. In this study the number of hidden nodes is taken as 25 and the regularization parameter λ (Section 6.3.8) which also act as the tuning parameter is varied

Table 6.18: Execution details of sf-ANN based on Spark and conventional system

Dataset	FS Technique	Conv. Train Time	Conv. Test Time	Processing eff. (Conv.)	Spark Train Time	Spark Test Time	Processing eff. (Spark)	Spark Train Acc. (%)	Spark Test Acc. (%)	Hidden Nodes	Iteration
GSE13159	ANOVA	961051.70	26956.73	0.03	178410.71	5235.12	0.13	82.86	77.00	150	50
	Kruskal-Wallis	1052127.11	22481.59	0.03	206837.30	4179.35	0.17	81.31	79.75	110	50
	Friedman	502044.24	11099.44	0.06	101695.55	2176.70	0.32	79.21	74.59	200	50
GSE13204	ANOVA	62407.77	1559.37	0.69	12665.87	525.02	2.06	85.59	81.86	90	50
	Kruskal-Wallis	55553.21	1035.61	1.05	11951.45	454.59	2.38	87.23	82.48	200	50
	Friedman	54381.84	1682.12	0.64	12506.63	625.53	1.73	82.34	80.30	150	50
GSE15061	ANOVA	85560.78	1929.20	0.15	20088.51	539.76	0.54	77.39	72.72	30	50
	Kruskal-Wallis	80126.71	2384.42	0.12	18297.15	749.95	0.39	75.68	72.86	10	50
	Friedman	465459.32	9260.90	0.03	114105.80	2520.37	0.12	78.52	73.24	30	50

Table 6.19: Performance parameter of sf-ANN with various feature selection methods using GSE15061 dataset.

FS method	Performance parameter	1	2	3	Average
ANOVA	**Recall**	0.7185	0.8440	0.5000	0.6875
	Precision	0.9700	0.6301	0.5227	0.7076
	F-Measure	0.8255	0.7216	0.5111	0.6861
	Accuracy				0.7310
Kruskal-Wallis	**Recall**	0.9185	0.5596	0.6304	0.7029
	Precision	0.7168	0.8841	0.6042	0.7350
	F-Measure	0.8052	0.6854	0.6170	0.7025
	Accuracy				0.7379
Friedman	**Recall**	0.6889	0.8349	0.4565	0.6601
	Precision	0.9688	0.6149	0.4565	0.6800
	F-Measure	0.8052	0.7082	0.4565	0.6566
	Accuracy				0.7069



(a) GSE15061 (b) GSE13204 (c) GSE13159

Figure 6.12: Comparison in terms of execution time, between Spark and Conventional system (Conv.) for RBFN (hybrid) classifier (in testing phase)

within the range $\lambda \in [2^{-10}, 2^5]$ (with step size = 1 for $log\ \lambda$). The overall execution details and processing efficiency of sf-RBFN classifier using gradient descent learning technique is tabulated in Table 6.26. Processing efficiency in Table 6.26 signifies the number of samples processed per second. The amount of time consumed by RBFN (gradient descent) classifier , on Spark and conventional system is shown in Figure 6.13. From this figure, it is evident that the time taken to analyze the datasets by RBFN (gradient descent) classifier based on Spark framework (sf-RBFN) is much less as compared to the same in conventional system. The performance parameters of sf-RBFN classifier using gradient descent learning technique, i.e., Recall, Precision, F-Measure and Accuracy for dataset GSE15061, GSE13204, and GSE13159 are mentioned in Table 6.27, 6.28, and 6.29 respectively.

6.4.2 Comparative analysis

In this section, emphasis has been laid on designing classifiers based on Spark framework, which works in a distributed manner and is used for predictive analysis of large datasets. As a result, it helps in reducing the overall processing time and increasing the overall processing efficiency.

Table 6.20: Performance parameter of sf-ANN with various feature selection methods using GSE13204 dataset.

FS method	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.7600	0.8537	0.7333	0.3333	0.9375	0.7836	0.5000	0.8947	0.9474	1.0000	0.9000	0.6881	0.8409	0.9677	0.9405	0.7213	0.6667	0.6667	0.7853
	Precision	0.5135	0.6731	0.6875	0.6667	0.7500	0.8874	0.8182	0.9444	1.0000	0.9870	0.8571	0.6944	0.4405	0.9091	0.9405	0.8462	0.8073	0.8000	0.7902
	F-Measure	0.6129	0.7527	0.7097	0.4444	0.8333	0.8323	0.6207	0.9189	0.9730	0.9935	0.8780	0.6912	0.5781	0.9375	0.9405	0.7788	0.7303	0.7273	0.7752
	Accuracy																			0.8190
Kruskal-Wallis	Recall	0.7600	0.9024	0.8667	0.3333	0.9375	0.7836	0.5000	0.9474	0.9474	1.0000	0.9000	0.6881	0.8409	0.9677	0.9405	0.7213	0.6970	0.5000	0.7908
	Precision	0.5278	0.7115	0.8125	0.6667	0.7500	0.8874	0.9000	0.9474	1.0000	0.9870	0.8571	0.6944	0.4353	0.9375	0.9405	0.8627	0.8142	1.0000	0.8184
	F-Measure	0.6230	0.7957	0.8387	0.4444	0.8333	0.8323	0.6429	0.9474	0.9730	0.9935	0.8780	0.6912	0.5736	0.9524	0.9405	0.7857	0.7510	0.6667	0.7868
	Accuracy																			0.8264
Friedman	Recall	0.7600	0.8537	0.7333	0.3333	0.9375	0.7778	0.5000	0.8947	0.8947	0.9956	0.8500	0.6789	0.7727	0.9032	0.9286	0.6885	0.6515	0.5000	0.7586
	Precision	0.5000	0.6731	0.6875	0.6667	0.7143	0.8867	0.7500	0.8947	0.9444	0.9742	0.8095	0.6727	0.4198	0.9032	0.9286	0.8235	0.7963	0.6000	0.7581
	F-Measure	0.6032	0.7527	0.7097	0.4444	0.8108	0.8287	0.6000	0.8947	0.9189	0.9848	0.8293	0.6758	0.5440	0.9032	0.9286	0.7500	0.7167	0.5455	0.7467
	Accuracy																			0.8033

Table 6.21: Performance parameter of sf-ANN with various feature selection methods using GSE13159 dataset.

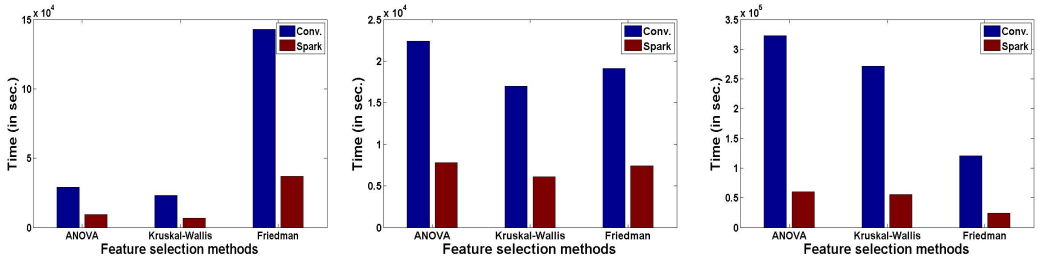
FS method	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	**Recall**	0.9286	0.9474	0.9167	0.5625	1.0000	0.6068	0.6154	0.8333	0.9286	1.0000	0.9200	0.7101	0.6800	0.9130	0.9483	0.7073	0.6709	1.0000	0.8272
	Precision	0.6500	0.7826	0.7333	0.3214	1.0000	0.8659	0.3077	1.0000	0.9286	1.0000	0.6970	0.7000	0.5152	0.8750	0.9483	0.7632	0.8548	0.8000	0.7635
	F-Measure	0.7647	0.8571	0.8148	0.4091	1.0000	0.7136	0.4103	0.9091	0.9286	1.0000	0.7931	0.7050	0.5862	0.8936	0.9483	0.7342	0.7518	0.8889	0.7838
	Accuracy																			0.8040
Kruskal-Wallis	**Recall**	0.8571	0.8947	0.8333	0.5000	1.0000	0.6923	0.6154	0.8333	0.9286	1.0000	0.9600	0.7101	0.6800	0.8696	0.9483	0.6829	0.6582	1.0000	0.8147
	Precision	0.6316	0.7391	0.6667	0.3200	1.0000	0.8710	0.4211	1.0000	1.0000	1.0000	0.7273	0.6806	0.5313	0.8696	0.9483	0.7568	0.8000	1.0000	0.7757
	F-Measure	0.7273	0.8095	0.7407	0.3902	1.0000	0.7714	0.5000	0.9091	0.9630	1.0000	0.8276	0.6950	0.5965	0.8696	0.9483	0.7179	0.7222	1.0000	0.7882
	Accuracy																			0.8097
Friedman	**Recall**	0.9286	0.9474	0.8333	0.5000	1.0000	0.5897	0.6154	0.8333	0.9286	1.0000	0.9200	0.7101	0.6800	0.9130	0.9483	0.7073	0.6709	1.0000	0.8181
	Precision	0.6190	0.7500	0.7143	0.2963	0.9000	0.8625	0.3077	1.0000	0.9286	1.0000	0.6970	0.6901	0.5152	0.8750	0.9483	0.7632	0.8548	0.8000	0.7512
	F-Measure	0.7429	0.8372	0.7692	0.3721	0.9474	0.7005	0.4103	0.9091	0.9286	1.0000	0.7931	0.7000	0.5862	0.8936	0.9483	0.7342	0.7518	0.8889	0.7730
	Accuracy																			0.7983

Table 6.22: Execution details of RBFN (hybrid) based on Spark and conventional system

Dataset	FS Technique	Conv. Train Time	Conv. Test Time	Processing eff. (Conv)	Spark Train Time	Spark Test Time	Processing eff. (Spark)	Spark Train Acc. (%)	Spark Test Acc. (%)	Spark Best Parameter	Iteration
GSE13159	ANOVA	265515.58	23278.42	0.03	51191.25	4120.65	0.17	88.11	82.12	0.5	20
	Kruskal-Wallis	281525.22	23806.81	0.03	53251.22	4290.13	0.16	81.95	80.69	0.25	20
	Friedman	129870.76	11679.75	0.06	25531.25	2104.76	0.33	79.45	79.40	0.5	20
GSE13204	ANOVA	15946.73	1388.23	0.78	3438.84	562.01	1.93	85.12	84.76	0.01	20
	Kruskal-Wallis	15696.82	1327.43	0.82	3179.27	497.15	2.18	86.45	84.95	0.003	20
	Friedman	16028.43	1511.25	0.72	3341.18	551.53	1.96	84.93	83.75	0.00005	20
GSE15061	ANOVA	21594.07	1825.74	0.16	5319.80	568.56	0.51	78.86	78.97	0.00005	20
	Kruskal-Wallis	21282.48	1916.19	0.15	5191.87	545.75	0.53	74.79	73.79	0.0001	20
	Friedman	138496.95	9203.11	0.03	33139.66	2697.96	0.11	77.55	74.48	0.003	20

Table 6.23: Performance parameter of sf-RBFN (hybrid) with various feature selection methods using GSE15061 dataset.

FS method	Performance parameter	1	2	3	Average
ANOVA	**Recall**	0.9481	0.6606	0.6304	0.7464
	Precision	0.7711	0.9000	0.6591	0.7767
	F-Measure	0.8505	0.7619	0.6444	0.7523
	Accuracy				0.7897
Kruskal-Wallis	**Recall**	0.9185	0.5596	0.6304	0.7029
	Precision	0.7168	0.8841	0.6042	0.7350
	F-Measure	0.8052	0.6854	0.6170	0.7025
	Accuracy				0.7379
Friedman	**Recall**	0.9037	0.6330	0.5435	0.6934
	Precision	0.7349	0.8519	0.5814	0.7227
	F-Measure	0.8106	0.7263	0.5618	0.6996
	Accuracy				0.7448



(a) GSE15061 (b) GSE13204 (c) GSE13159

Figure 6.13: Comparison in terms of execution time, between Spark and Conventional system (Conv.) for RBFN (gradient) classifier (in testing phase)

In this work, seven different types of classification techniques such as LoR, SVM, NB, KNN, ANN, RBFN (hybrid), RBFN (gradient) have been considered to classify the microarray dataset. Three different feature selection techniques such as ANOVA, Kruskal-Wallis, and Friedman have been considered to select the right set of features over three datasets GSE13159, GSE13204, and GSE15061 with two different performance parameters such as Accuracy and execution time (in sec). We have also considered two different platforms to execute this experiment. So for classification technique, a total number of two sets (one for each performance measure) are used, each with 18 (3 dataset * 3 feature selection techniques * 2 platforms) data points. The results of comparison analysis for performance parameters are summarized in Table 6.30.

Table 6.30 contains two sub tables. The first table shows the mean difference of accuracy parameter and second table shows the mean difference of time parameter. From Table 6.30a, we observe that RBFN (hybrid) yields better results as compared to other approaches. From Table 6.30b, we also observed that NB takes minimum time to process the datasets as compared to other techniques.

Hence, from the above experiment it is observed that the total execution time (average) of

Table 6.24: Performance parameter of sf-RBFN (hybrid) with various feature selection methods using GSE13204 dataset.

FS method	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.9200	0.9756	0.9333	0.6667	1.0000	0.7485	0.7778	0.9474	1.0000	0.9825	0.9250	0.7890	0.7727	0.9677	0.9286	0.7705	0.6667	1.0000	0.8762
	Precision	0.5610	0.7547	0.8235	0.5714	0.8889	0.9275	0.5385	1.0000	1.0000	0.9956	0.7872	0.7890	0.5763	0.9677	0.9512	0.8545	0.8544	0.4286	0.7928
	F-Measure	0.6970	0.8511	0.8750	0.6154	0.9412	0.8285	0.6364	0.9730	1.0000	0.9890	0.8506	0.7890	0.6602	0.9677	0.9398	0.8103	0.7489	0.6000	0.8207
	Accuracy																			0.8476
Kruskal-Wallis	Recall	0.9200	0.9756	0.9333	0.6667	1.0000	0.7485	0.7222	0.9474	1.0000	0.9825	0.9250	0.7982	0.7727	0.9677	0.9286	0.7869	0.6742	1.0000	0.8750
	Precision	0.5750	0.7547	0.8235	0.5714	0.8889	0.9209	0.5200	1.0000	1.0000	0.9956	0.8043	0.7909	0.5763	0.9677	0.9512	0.8571	0.8641	0.4286	0.7939
	F-Measure	0.7077	0.8511	0.8750	0.6154	0.9412	0.8258	0.6047	0.9730	1.0000	0.9890	0.8605	0.7945	0.6602	0.9677	0.9398	0.8205	0.7574	0.6000	0.8213
	Accuracy																			0.8495
Friedman	Recall	0.8800	0.9268	0.8667	0.6250	0.8750	0.7427	0.6667	0.8947	0.9474	0.9737	0.9000	0.7982	0.7727	0.9677	0.9524	0.7869	0.6742	0.8333	0.8380
	Precision	0.5789	0.7451	0.6842	0.5357	0.8235	0.9137	0.4800	1.0000	0.9474	0.9911	0.8000	0.7632	0.5965	0.9677	0.9524	0.8571	0.8558	0.3333	0.7681
	F-Measure	0.6984	0.8261	0.7647	0.5769	0.8485	0.8194	0.5581	0.9444	0.9474	0.9823	0.8471	0.7803	0.6733	0.9677	0.9524	0.8205	0.7542	0.4762	0.7910
	Accuracy																			0.8375

Table 6.25: Performance parameter of sf-RBFN (hybrid) with various feature selection methods using GSE13159 dataset.

FS method	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.7857	0.8947	0.6667	0.1250	0.2222	0.8462	0.0769	0.7500	0.7857	1.0000	0.7200	0.9130	0.6400	0.9130	0.9483	0.6341	0.8101	0.5000	0.6795
	Precision	0.8462	0.8095	1.0000	0.6667	1.0000	0.7734	0.5000	1.0000	0.9167	1.0000	0.9000	0.6176	0.5333	0.8750	0.9483	0.8667	0.7711	0.4000	0.8014
	F-Measure	0.8148	0.8500	0.8000	0.2105	0.3636	0.8082	0.1333	0.8571	0.8462	1.0000	0.8000	0.7368	0.5818	0.8936	0.9483	0.7324	0.7901	0.4444	0.7006
	Accuracy																			0.8212
Kruskal-Wallis	Recall	0.7143	0.8421	0.5833	0.1875	0.2222	0.8462	0.1538	0.6667	0.7857	1.0000	0.6800	0.8986	0.6000	0.8261	0.9310	0.6098	0.7975	0.5000	0.6580
	Precision	0.8333	0.8000	0.8750	0.5000	0.6667	0.7857	0.5000	1.0000	0.9167	1.0000	0.8500	0.6139	0.5000	0.8261	0.9153	0.8065	0.7778	0.3333	0.7500
	F-Measure	0.7692	0.8205	0.7000	0.2727	0.3333	0.8148	0.2353	0.8000	0.8462	1.0000	0.7556	0.7294	0.5455	0.8261	0.9231	0.6944	0.7875	0.4000	0.6808
	Accuracy																			0.8069
Friedman	Recall	0.7143	0.7895	0.5000	0.1875	0.2222	0.8205	0.2308	0.6667	0.7857	0.9933	0.6400	0.8841	0.6000	0.7826	0.9310	0.5854	0.7848	0.7500	0.6594
	Precision	0.8333	0.7895	0.7500	0.4286	0.6667	0.8000	0.5000	0.8000	0.7857	0.9867	0.7619	0.6100	0.5172	0.8182	0.9000	0.8000	0.7470	0.6000	0.7275
	F-Measure	0.7692	0.7895	0.6000	0.2609	0.3333	0.8101	0.3158	0.7273	0.7857	0.9900	0.6957	0.7219	0.5556	0.8000	0.9153	0.6761	0.7654	0.6667	0.6766
	Accuracy																			0.7940

Table 6.26: Execution details of sf-RBFN (gradient descent) based on Spark and conventional system

Dataset	FS Technique	Conv. Train Time	Conv. Test Time	Processing eff. (Conv)	Spark Train Time	Spark Test Time	Processing eff. (Spark)	Spark Train Acc. (%)	Spark Test Acc. (%)	Spark Best Parameter	Iteration
GSE13159	ANOVA	2710352.54	322695.48	0.0022	504088.66	60337.10	0.0116	85.00	82.92	0.01	20
	Kruskal-Wallis	3089404.59	271517.23	0.0026	633498.20	55659.19	0.0126	82.31	81.33	0.01	20
	Friedman	1515983.96	120888.30	0.0058	333422.34	24480.19	0.0286	81.74	78.38	0.25	20
GSE13204	ANOVA	232609.40	22397.18	0.0484	46734.62	7803.63	0.1388	82.45	83.05	0.5	20
	Kruskal-Wallis	214047.43	16960.08	0.0639	40870.27	6122.56	0.1769	87.74	86.38	0.5	20
	Friedman	236950.73	19100.66	0.0567	46669.20	7400.55	0.1463	88.55	85.33	0.5	20
GSE15061	ANOVA	336491.95	29057.29	0.0100	83513.53	9221.19	0.0314	83.00	78.31	0.0001	20
	Kruskal-Wallis	355243.11	23098.49	0.0126	92290.34	6797.39	0.0427	84.55	80.33	0.00005	20
	Friedman	1514040.32	142838.40	0.0020	426587.48	37032.19	0.0078	79.74	75.38	0.003	20

Table 6.27: Performance parameter of sf-RBFN (gradient) with various feature selection methods using GSE15061 dataset.

FS method	Performance parameter	1	2	3	Average
ANOVA	**Recall**	0.7333	0.8991	0.6304	0.7543
	Precision	0.9900	0.6759	0.6444	0.7701
	F-Measure	0.8426	0.7717	0.6374	0.7505
	Accuracy				0.7793
Kruskal-Wallis	**Recall**	0.7185	0.8624	0.6087	0.7299
	Precision	0.9898	0.6620	0.5600	0.7373
	F-Measure	0.8326	0.7490	0.5833	0.7217
	Accuracy				0.7552
Friedman	**Recall**	0.7037	0.8532	0.5435	0.7001
	Precision	0.9896	0.6370	0.5208	0.7158
	F-Measure	0.8225	0.7294	0.5319	0.6946
	Accuracy				0.7345

these proposed methods on Spark is reduced by approximately 78.35% than the conventional system.

6.5 Summary

In this chapter, various classifiers based on Spark framework are designed to classify large microarray datasets. The proposed approach works in a distributed manner on scalable clusters, and its performance increases with increase in data size. From the obtained results, it is inferred that sf-RBFN (hybrid) classifier provides better accuracy with ANOVA test. The three major contributions of this chapter:

- Harnessing the power of distributed computing for better storage and faster processing of datasets.
- Design of Spark based classifiers.
- Comparative analysis of processing time between Conventional system and Spark for selecting features and classifying the datasets.

Table 6.28: Performance parameter of sf-RBFN (gradient) with various feature selection methods using GSE13204 dataset.

FS method	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.8800	0.9756	1.0000	0.4167	0.9375	0.8187	0.5000	0.9474	0.9474	1.0000	0.9000	0.6881	0.8409	0.9677	0.9524	0.7213	0.6970	0.5000	0.8161
	Precision	0.5789	0.7407	0.8824	0.8333	0.8333	0.9032	1.0000	1.0000	1.0000	0.9870	0.8571	0.7009	0.4353	0.9677	0.9524	0.8627	0.8364	1.0000	0.8540
	F-Measure	0.6984	0.8421	0.9375	0.5556	0.8824	0.8589	0.6667	0.9730	0.9730	0.9935	0.8780	0.6944	0.5736	0.9677	0.9524	0.7857	0.7603	0.6667	0.8144
	Accuracy																			0.8421
Kruskal-Wallis	Recall	0.8800	0.9756	0.9333	0.3333	1.0000	0.8070	0.5000	0.9474	0.9474	1.0000	0.9000	0.6881	0.8409	0.9677	0.9405	0.7213	0.6970	0.5000	0.8100
	Precision	0.5946	0.7407	0.8750	0.8000	0.8421	0.8961	1.0000	1.0000	1.0000	0.9870	0.8571	0.6944	0.4353	0.9375	0.9405	0.8627	0.8214	1.0000	0.8491
	F-Measure	0.7097	0.8421	0.9032	0.4706	0.9143	0.8492	0.6667	0.9730	0.9730	0.9935	0.8780	0.6912	0.5736	0.9524	0.9405	0.7857	0.7541	0.6667	0.8076
	Accuracy																			0.8375
Friedman	Recall	0.7600	0.9024	0.8667	0.3333	0.9375	0.7836	0.5000	0.9474	0.9474	1.0000	0.9000	0.6881	0.8409	0.9677	0.9405	0.7213	0.6970	0.5000	0.7908
	Precision	0.5278	0.7115	0.8125	0.6667	0.7500	0.8874	0.9000	0.9474	1.0000	0.9870	0.8571	0.6944	0.4353	0.9375	0.9405	0.8627	0.8142	1.0000	0.8184
	F-Measure	0.6230	0.7957	0.8387	0.4444	0.8333	0.8323	0.6429	0.9474	0.9730	0.9935	0.8780	0.6912	0.5736	0.9524	0.9405	0.7857	0.7510	0.6667	0.7868
	Accuracy																			0.8264

Table 6.29: Performance parameter of sf-RBFN (gradient) with various feature selection methods using GSE13159 dataset.

FS method	Performance parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Average
ANOVA	Recall	0.9286	0.9474	0.8333	0.5625	1.0000	0.7179	0.6923	1.0000	0.9286	1.0000	1.0000	0.7246	0.7200	0.8696	0.9483	0.6829	0.6582	1.0000	0.8452
	Precision	0.6500	0.7200	0.6667	0.3750	1.0000	0.8936	0.4500	1.0000	1.0000	1.0000	0.7353	0.7353	0.5806	0.9091	0.9483	0.7568	0.8125	1.0000	0.7907
	F-Measure	0.7647	0.8182	0.7407	0.4500	1.0000	0.7962	0.5455	1.0000	0.9630	1.0000	0.8475	0.7299	0.6429	0.8889	0.9483	0.7179	0.7273	1.0000	0.8100
	Accuracy																			0.8269
Kruskal-Wallis	Recall	0.8571	0.8947	0.9167	0.5000	0.8889	0.7009	0.6154	0.8333	0.9286	1.0000	0.9600	0.7101	0.6800	0.8696	0.9483	0.6829	0.6709	0.7500	0.8004
	Precision	0.6667	0.7083	0.7333	0.3200	1.0000	0.8542	0.4444	1.0000	1.0000	1.0000	0.7273	0.6806	0.5313	0.8696	0.9483	0.7568	0.8154	1.0000	0.7809
	F-Measure	0.7500	0.7907	0.8148	0.3902	0.9412	0.7700	0.5161	0.9091	0.9630	1.0000	0.8276	0.6950	0.5965	0.8696	0.9483	0.7179	0.7361	0.8571	0.7830
	Accuracy																			0.8112
Friedman	Recall	0.8571	0.8947	0.8333	0.5000	1.0000	0.6923	0.4615	0.8333	0.7857	0.9933	0.9600	0.7101	0.6800	0.8696	0.9483	0.6829	0.6582	1.0000	0.7978
	Precision	0.6316	0.7083	0.6667	0.3200	0.9000	0.8710	0.3333	0.9091	1.0000	1.0000	0.7273	0.6712	0.5313	0.8696	0.9483	0.7568	0.8000	1.0000	0.7580
	F-Measure	0.7273	0.7907	0.7407	0.3902	0.9474	0.7714	0.3871	0.8696	0.8800	0.9966	0.8276	0.6901	0.5965	0.8696	0.9483	0.7179	0.7222	1.0000	0.7707
	Accuracy																			0.8026

Table 6.30: Performance comparison of various classifiers

(a) Comparision of accuracy

	sf-LoR	sf-SVM	sf-NB	sf-KNN	sf-ANN	sf-RBFN (hybrid)	sf-RBFN (gradient)
sf-LoR	0.00	-3.31	-2.20	-4.69	-3.63	-5.77	-5.62
sf-SVM	3.31	0.00	1.11	-1.38	-0.32	-2.46	-2.31
sf-NB	2.20	-1.11	0.00	-2.49	-1.43	-3.57	-3.42
sf-KNN	4.69	1.38	2.49	0.00	1.06	-1.08	-0.93
sf-ANN	3.63	0.32	1.43	-1.06	0.00	-2.14	-1.99
sf-RBFN (hybrid)	5.77	2.46	3.57	1.08	2.14	0.00	0.15
sf-RBFN (gradient)	5.62	2.31	3.42	0.93	1.99	-0.15	0.00

(b) Comaparison of execution time

	sf-LoR	sf-SVM	sf-NB	sf-KNN	sf-ANN	sf-RBFN (hybrid)	sf-RBFN (gradient)
sf-LoR	0.00	146.07	459.73	-4211.68	-4814.84	-4076.84	-65259.91
sf-SVM	-146.07	0.00	313.65	-4357.76	-4960.91	-4222.92	-65405.98
sf-NB	-459.73	-313.65	0.00	-4671.41	-5274.56	-4536.57	-65719.64
sf-KNN	4211.68	4357.76	4671.41	0.00	-603.15	134.84	-61048.23
sf-ANN	4814.84	4960.91	5274.56	603.15	0.00	737.99	-60445.07
sf-RBFN (hybrid)	4076.84	4222.92	4536.57	-134.84	-737.99	0.00	-61183.07
sf-RBFN (gradient)	65259.91	65405.98	65719.64	61048.23	60445.07	61183.07	0.00

Chapter 7

Conclusions and Future Work

Analysis of microarray high-dimensional data is of great concern to physicians/scientists for early diagnosis of cancer. The microarray contains huge amount of information. To extract relevant information and analyze the data in a reasonable time is very essential. For the sake of analysis in an efficient manner, machine learning techniques are applied. As the data size of microarray becomes huge in nature, various machine learning techniques on scalable platforms are employed to analyze this data efficiently.

To start with, in this thesis, a brief review on the literature available for the analysis of microarray data using machine learning techniques has been carried out. The survey includes various criteria chosen by the authors such as the feature selection and classification methods employed, and the datasets used. The results on the survey work done for microarray data classification conclude that a good number of researchers and practitioners have considered statistical tests as techniques for feature selection and the various machine learning techniques for classification of the dataset. Subsequently, the implementation of the existing feature selection methods and classifiers, which are most frequently applied to classify the microarray datasets are implemented and the results are compared using three datasets viz., Leukemia, Breast, and Ovarian cancer. From the obtained results, it is revealed that feature selection methods play a significant role in the classification of microarray data. In the next chapter, various kernel based classifiers like ELM, RVM, and KFIS with linear, polynomial, RBF, and Tansig kernel functions are proposed and their performances are compared with the SVM. As the size of dataset increases the traditional machine learning techniques are not suitable enough to process efficiently within a definite span of time. To mitigate these issues, distributed and scalable platform like Hadoop is considered for storage (HDFS) and processing (MapReduce and Spark) of data in a distributed manner. To validate the efficacy of the distributed and scalable systems, the microarray high-dimensional datasets with various sizes have been considered. Hence, the existing methodologies as mentioned in the literature have been implemented on scalable platforms to analyze the microarray datasets.

Chapter 4 deals with the implementation of scalable feature selection methods like ANOVA, Kruskal-Wallis, and Friedman tests. The proposed methods are implemented with MapReduce and Spark on Hadoop cluster and their performance is measured. From

the obtained results, it is concluded that the processing of datasets to select the relevant and significant features on Spark is faster than MapReduce and conventional system (i.e., $T_{max}^{Spark} \leq T_{max}^{MR}, T_{max}^{Conv}$). Hence, from the above experiment, it is observed that the total execution time (average) of these proposed methods on Spark is reduced by approximately 81.94% and 46.78% than the conventional system and MapReduce respectively; and the execution time on MapReduce is reduced by 66.06% than conventional system. After selecting the relevant and significant features, various scalable classifiers are proposed in the subsequent chapters to classify the high-dimensional multi-class datasets. In Chapter 5 a scalable proximal support vector machine classifier has been proposed, and implemented using MapReduce and Spark frameworks. The proposed classifier is executed on Hadoop as well as conventional system and the performance is compared. From this chapter, it is concluded that Spark is more efficient than MapReduce and conventional system to analyze the datasets.

Finally in chapter 6, a scalable implementation of various classifiers has been proposed to classify the microarray data. The proposed models are implemented using Spark framework on the top of Hadoop cluster, and their efficiencies are measured with the conventional system and the results are compared. The processing efficiencies of these models on Spark are much greater than that on conventional system. It is also observed that the overall execution time (average) of these proposed methods on Spark is reduced by approximately 78.35% than the conventional system. Therefore, the efficiency to process the datasets using Spark is increased by approximately 78.35%. From the obtained results, It is concluded that to process the high-dimensional data with big sizes (GBs, TBs, etc.), the distributed and scalable cluster like Hadoop is better choice for the researchers.

Scope for Further Research

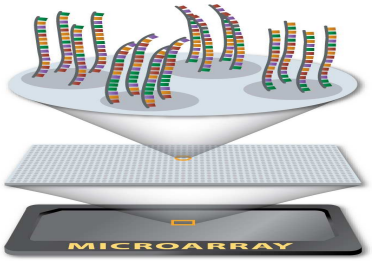
The analysis of microarray high-dimensional data using distributed and scalable platform described in this thesis unwraps some interesting research directions. It is known that the curse of dimensionality is a major issue in analyzing the high-dimensional data. Hence various feature selection/extraction techniques can be implemented to process the high-dimensional Big data based on scalable framework like Spark. After feature selection/extraction, to classify the high-dimensional big data, various machine learning classifiers based on kernel methods, ensembles of the classifiers, deep learning, neuro-fuzzy techniques, decision tree, etc. can be implemented on the same framework like Spark and their performance can be investigated. The work can also be extended in the direction of real time analytics using Spark streaming or Storm.

Chapter A

Microarray Data

In the era of twentieth century, scientists came up with several ways to study the genes such as mapping them, making mutation, cloning, sequencing, and analyzing the protein they encode. But, it took a lot of times to study the gene one by one. All living organisms have plenty of genes (e.g., Human $\geq 50,000$ genes). Hence, it would take a huge amount of time to analyze each human gene one at a time. Microarray is a technology with the size of a microscope slide, or even smaller where scientists can study many genes at a time or they can learn about every gene in a single experiment. It contains thousands of spots and each spot contains the strands of DNA sequence corresponding to a single gene. The cell types can be differentiated by measuring the gene expression of different cells indicated on the microarray chip. Figures A.1a, A.1b, A.1c show the Microarray chip, structure of one spot on Microarray data and the values of each spot after scanning the Microarray chip respectively.

(a) Microarray chip



(b) DNA spot on Microarray

ID_REF	VALUE_DS
AFFX-BioB-5_at	1126.74
AFFX-BioB-M_at	1167.66
AFFX-BioB-3_at	840.345
AFFX-BioC-5_at	2572.97
AFFX-BioC-3_at	3291.99
AFFX-BioDn-5_at	6171.86
AFFX-BioDn-3_at	11723.4
AFFX-CreX-5_at	27573.8
AFFX-CreX-3_at	28086.4
AFFX-DapX-5_at	350.522
AFFX-DapX-M_at	777.328
AFFX-DapX-3_at	1236
AFFX-LysX-5_at	58.9167
AFFX-LysX-M_at	113.665
AFFX-LysX-3_at	188.708
AFFX-PheX-5_at	130.262
AFFX-PheX-M_at	108.407
AFFX-PheX-3_at	256.371
AFFX-ThrX-5_at	82.8532
AFFX-ThrX-M_at	136.71

Total number of rows: **54675**

(c) Sample of Microarray data

Figure A.1: Microarray Data

References

[1] G. Russo, C. Zegar, and A. Giordano, ‘‘Advantages and limitations of microarray technology in human cancer,’’ *Oncogene*, vol. 22, no. 42, pp. 6497–6507, 2003.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] B. P. RAO, ‘‘Brief notes on big data: A cursory look,’’ 2015.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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

[16] 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.

[17] 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.

[18] 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.

[19] T. White, *Hadoop: The definitive guide*. ” O’Reilly Media, Inc.”, 2012.

[20] D. Borthakur, ‘‘The hadoop distributed file system: Architecture and design,’’ *Hadoop Project Website*, vol. 11, no. 2007, pp. 1–21, 2007.

[21] 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.

[22] 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.

[23] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, ‘‘Spark: cluster computing with working sets.’’ *HotCloud*, vol. 10, pp. 10–10, 2010.

[24] 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.

[25] 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.

[26] J.-G. Zhang and H.-W. Deng, ‘‘Gene selection for classification of microarray data based on the bayes error,’’ *BMC bioinformatics*, vol. 8, no. 1, pp. 370–378, October 2007.

[27] X. Hang, ‘‘Cancer classification by sparse representation using microarray gene expression data,’’ in *IEEE International Conference on Bioinformatics and Biomeidcine Workshops (BIBMW),PA, USA*. IEEE, November 2008, pp. 174–177.

[28] 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.

[29] 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.

[30] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, ‘‘Support vector machine classification and validation of cancer tissue samples using microarray expression data,’’ *Bioinformatics*, vol. 16, no. 10, pp. 906–914, May 2000.

[31] L. Li, C. R. Weinberg, T. A. Darden, and L. G. Pedersen, ‘‘Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the ga/knn method,’’ *Bioinformatics*, vol. 17, no. 12, pp. 1131–1142, June 2001.

[32] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini, ‘‘Tissue classification with gene expression profiles,’’ *Journal of Computational Biology*, vol. 7, no. 3-4, pp. 559–583, July 2004.

[33] D. V. Nguyen and D. M. Rocke, ‘‘Tumor classification by partial least squares using microarray gene expression data,’’ *Bioinformatics*, vol. 18, no. 1, pp. 39–50, March 2002.

[34] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, ‘‘Gene selection for cancer classification using support vector machines,’’ *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[35] 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.

[36] 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.

[37] J.-H. Cho, D. Lee, J. H. Park, and I.-B. Lee, ‘‘Gene selection and classification from microarray data using kernel machine,’’ *Federation of European Biochemical Societies (FEBS) letters*, vol. 571, no. 1, pp. 93–98, July 2004.

[38] Y. Lee and C.-K. Lee, ‘‘Classification of multiple cancer types by multicategory support vector machines using gene expression data,’’ *Bioinformatics*, vol. 19, no. 9, pp. 1132–1139, December 2003.

[39] T. K. Paul and H. Iba, ‘‘Selection of the most useful subset of genes for gene expression-based classification,’’ in *Congress on Evolutionary Computation (CEC), Portland, Oregon*, vol. 2. IEEE, June 2004, pp. 2076–2083.

[40] E. B. Huerta, B. Duval, and J.-K. Hao, ‘‘A hybrid ga/svm approach for gene selection and classification of microarray data,’’ in *Applications of Evolutionary Computing*. Springer, 2006, pp. 34–44.

[41] J. Ye, T. Li, T. Xiong, and R. Janardan, ‘‘Using uncorrelated discriminant analysis for tissue classification with gene expression data,’’ *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 1, no. 4, pp. 181–190, October 2004.

[42] J. J. Liu, G. Cutler, W. Li, Z. Pan, S. Peng, T. Hoey, L. Chen, and X. B. Ling, ‘‘Multiclass cancer classification and biomarker discovery using ga-based algorithms,’’ *Bioinformatics*, vol. 21, no. 11, pp. 2691–2697, March 2005.

[43] E. Alba, J. Garcia-Nieto, L. Jourdan, and E.-G. Talbi, ‘‘Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms,’’ in *IEEE Congress on Evolutionary Computation, CEC, Singapore*. IEEE, 2007, pp. 284–290.

[44] L. Yu and H. Liu, ‘‘Redundancy based feature selection for microarray data,’’ in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, USA*. ACM, 2004, pp. 737–742.

[45] C. Ding and H. Peng, ‘‘Minimum redundancy feature selection from microarray gene expression data,’’ *Journal of bioinformatics and computational biology*, vol. 3, no. 02, pp. 185–205, June 2005.

[46] S. B. Cho and H.-H. Won, ‘‘Cancer classification using ensemble of neural networks with multiple significant gene subsets,’’ *Applied Intelligence*, vol. 26, no. 3, pp. 243–250, June 2007.

[47] W.-H. Yang, D.-Q. Dai, and H. Yan, ‘‘Generalized discriminant analysis for tumor classification with gene expression data,’’ in *International Conference on Machine Learning and Cybernetics, Beijing, China*. IEEE, 2006, pp. 4322–4327.

[48] Z. Wang, V. Palade, and Y. Xu, ‘‘Neuro-fuzzy ensemble approach for microarray cancer gene expression data analysis,’’ in *Evolving Fuzzy Systems, 2006 International Symposium on*. IEEE, 2006, pp. 241–246.

[49] S. Pang, I. Havukkala, Y. Hu, and N. Kasabov, ‘‘Classification consistency analysis for bootstrapping gene selection,’’ *Neural Computing and Applications*, vol. 16, no. 6, pp. 527–539, March 2007.

[50] G.-Z. Li, X.-Q. Zeng, J. Y. Yang, and M. Q. Yang, ‘‘Partial least squares based dimension reduction with gene selection for tumor classification,’’ in *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering (BIBE), Boston, MA*. IEEE, 2007, pp. 1439–1444.

[51] F. Yue, K. Wang, and W. Zuo, ‘‘Informative gene selection and tumor classification by null space lda for microarray data,’’ in *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Springer, 2007, pp. 435–446.

[52] S. Li, X. Wu, and X. Hu, ‘‘Gene selection using genetic algorithm and support vectors machines,’’ *Soft computing*, vol. 12, no. 7, pp. 693–698, May 2008.

[53] 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.

[54] Y. Wang, I. V. Tetko, M. A. Hall, E. Frank, A. Facius, K. F. Mayer, and H. W. Mewes, ‘‘Gene selection from microarray data for cancer classification-a machine learning approach,’’ *Computational biology and chemistry*, vol. 29, no. 1, pp. 37–46, November 2005.

[55] R. Ruiz, J. C. Riquelme, and J. S. Aguilar-Ruiz, ‘‘Incremental wrapper-based gene selection from microarray data for cancer classification,’’ *Pattern Recognition*, vol. 39, no. 12, pp. 2383–2392, December 2006.

[56] J. Zhu and T. Hastie, ‘‘Classification of gene microarrays by penalized logistic regression,’’ *Biostatistics*, vol. 5, no. 3, pp. 427–443, January 2004.

[57] H. Huynh, J.-J. Kim, and Y. Won, ‘‘Classification study on dna microarray with feedforward neural network trained by singular value decomposition,’’ *International Journal of Bio-Science and Bio-Technology*, vol. 1, no. 1, pp. 17–24, 2009.

[58] R. Díaz-Uriarte and S. A. De Andres, ‘‘Gene selection and classification of microarray data using random forest,’’ *BMC bioinformatics*, vol. 7, no. 1, p. 3, June 2006.

[59] 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.

[60] H. Yu, G. Gu, H. Liu, J. Shen, and C. Zhu, ‘‘A novel discrete particle swarm optimization algorithm for microarray data-based tumor marker gene selection,’’ in *Computer Science and Software Engineering, 2008 International Conference on*, vol. 1. IEEE, December 2008, pp. 1057–1060.

[61] Y. Chen and Y. Zhao, ‘‘A novel ensemble of classifiers for microarray data classification,’’ *Applied soft computing*, vol. 8, no. 4, pp. 1664–1669, September 2008.

[62] H. Liu, L. Liu, and H. Zhang, ‘‘Ensemble gene selection for cancer classification,’’ *Pattern Recognition*, vol. 43, no. 8, pp. 2763–2772, August 2010.

[63] 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.

[64] E. Huerta, B. Duval, and J.-K. Hao, ‘‘Fuzzy logic for elimination of redundant information of microarray data,’’ *Genomics, Proteomics & Bioinformatics*, vol. 6, no. 2, pp. 61–73, June 2008.

[65] X. Liu, A. Krishnan, and A. Mondry, ‘‘An entropy-based gene selection method for cancer classification using microarray data,’’ *BMC bioinformatics*, vol. 6, no. 1, p. 76, March 2005.

[66] K.-H. Liu and D.-S. Huang, ‘‘Cancer classification using rotation forest,’’ *Computers in Biology and Medicine*, vol. 38, no. 5, pp. 601–610, May 2008.

[67] A. Sharma and K. K. Paliwal, ‘‘Cancer classification by gradient lda technique using microarray gene expression data,’’ *Data & Knowledge Engineering*, vol. 66, no. 2, pp. 338–347, August 2008.

[68] 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.

[69] J. Shim, I. Sohn, S. Kim, J. W. Lee, P. E. Green, and C. Hwang, ‘‘Selecting marker genes for cancer classification using supervised weighted kernel clustering and the support vector machine,’’ *Computational Statistics & Data Analysis*, vol. 53, no. 5, pp. 1736–1742, March 2009.

[70] J.-H. Hong and S.-B. Cho, ‘‘A probabilistic multi-class strategy of one-vs.-rest support vector machines for cancer classification,’’ *Neurocomputing*, vol. 71, no. 16, pp. 3275–3281, October 2008.

[71] Z. Chen, J. Li, and L. Wei, ‘‘A multiple kernel support vector machine scheme for feature selection and rule extraction from gene expression data of cancer tissue,’’ *Artificial Intelligence in Medicine*, vol. 41, no. 2, pp. 161–175, 2007.

[72] M. F. Akay, ‘‘Support vector machines combined with feature selection for breast cancer diagnosis,’’ *Expert systems with applications*, vol. 36, no. 2, pp. 3240–3247, 2009.

[73] Q. Shen, W.-m. Shi, and W. Kong, ‘‘New gene selection method for multiclass tumor classification by class centroid,’’ *Journal of Biomedical Informatics*, vol. 42, no. 1, pp. 59–65, February 2009.

[74] H. Yu, G. Gu, H. Liu, J. Shen, and J. Zhao, ‘‘A modified ant colony optimization algorithm for tumor marker gene selection,’’ *Genomics, Proteomics & Bioinformatics*, vol. 7, no. 4, pp. 200–208, December 2009.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] D. P. Berrar, C. S. Downes, and W. Dubitzky, ‘‘Multiclass cancer classification using gene expression profiling and probabilistic neural networks,’’ in *Proceedings of the Pacific Symposium on Biocomputing, Hawaii, USA*, vol. 8. World Scientific, 2002, pp. 5–16.

[80] 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.

[81] F. F. Gonzalez-Navarro and L. A. Belanche-Muñoz, ‘‘Feature selection for microarray gene expression data using simulated annealing guided by the multivariate joint entropy,’’ *Computación y Sistemas*, vol. 18, no. 2, pp. 275–293, 2014.

[82] F. F. González-Navarro and L. A. Belanche-Muñoz, ‘‘Parsimonious selection of useful genes in microarray gene expression data,’’ in *Software Tools and Algorithms for Biological Systems*. Springer, 2011, pp. 45–55.

[83] R. Cai, Z. Hao, X. Yang, and W. Wen, ‘‘An efficient gene selection algorithm based on mutual information,’’ *Neurocomputing*, vol. 72, no. 4, pp. 991–999, January 2009.

[84] L. Wang, J. Zhu, and H. Zou, ‘‘Hybrid huberized support vector machines for microarray classification and gene selection,’’ *Bioinformatics*, vol. 24, no. 3, pp. 412–419, November 2008.

[85] H.-L. Bu, G.-Z. Li, and X.-Q. Zeng, ‘‘Reducing error of tumor classification by using dimension reduction with feature selection,’’ *Lecture Notes in Operations Research*, vol. 7, no. 124, pp. 232–241, August 2007.

[86] J.-H. Hong and S.-B. Cho, ‘‘Cancer classification with incremental gene selection based on dna microarray data,’’ in *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, (CIBCB’08), Sun Valley, Idaho*. IEEE, 2008, pp. 70–74.

[87] R. Hewett and P. Kijsanayothin, ‘‘Tumor classification ranking from microarray data,’’ *BMC genomics*, vol. 9, no. Suppl 2, p. S21, September 2008.

[88] H. Yu, S. Hong, X. Yang, J. Ni, Y. Dan, and B. Qin, ‘‘Recognition of multiple imbalanced cancer types based on dna microarray data using ensemble classifiers,’’ *BioMed research international*, vol. 2013, pp. 1–13, July 2013.

[89] A. Osareh and B. Shadgar, ‘‘An efficient ensemble learning method for gene microarray classification,’’ *BioMed research international*, vol. 2013, July 2013.

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

[91] S. KR, ‘‘Microarray data classification using support vector machine,’’ *International Journal of Biometrics and Bioinformatics (IJBB)*, vol. 5, no. 1, p. 10, 2011.

[92] K.-J. Kim and S.-B. Cho, ‘‘Prediction of colon cancer using an evolutionary neural network,’’ *Neurocomputing*, vol. 61, pp. 361–379, October 2004.

[93] GLOBOCAN, ‘‘Cancer incidence and mortality worldwide: Iarc,’’ http://globocan.iarc.fr/factsheets/populations/factsheet.asp?uno=900, 2008.

[94] Y. Saeys, I. Inza, and P. Larrañaga, ‘‘A review of feature selection techniques in bioinformatics,’’ *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, August 2007.

[95] M. Kumar and S. Kumar Rath, ‘‘Classification of microarray data using kernel fuzzy inference system,’’ *International Scholarly Research Notices*, vol. 2014, no. Article ID 769159, p. 18 pages, August 2014.

[96] M. Kumar, N. K. Rath, A. Swain, and S. K. Rath, ‘‘Feature selection and classification of microarray data using mapreduce based anova and k-nearest neighbor,’’ *Procedia Computer Science*, vol. 54, pp. 301–310, 2015.

[97] L. Deng, J. Pei, J. Ma, and D. L. Lee, ‘‘A rank sum test method for informative gene discovery,’’ in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 410–419.

[98] H. Liu and R. Setiono, ‘‘Chi2: Feature selection and discretization of numeric attributes,’’ in *2012 IEEE 24th International Conference on Tools with Artificial Intelligence, Vassilopoulos (Ed.). Herndon, Virginia*. IEEE Computer Society, 1995, pp. 388–388.

[99] M. Ben-Bassat, ‘‘Pattern recognition and reduction of dimensionality,’’ *Handbook of Statistics*, vol. 2, pp. 773–910, 1982.

[100] L. E. Raileanu and K. Stoffel, ‘‘Theoretical comparison between the gini index and information gain criteria,’’ *Annals of Mathematics and Artificial Intelligence*, vol. 41, no. 1, pp. 77–93, May 2004.

[101] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[102] T. M. Mitchell, ‘‘Machine learning. 1997,’’ *Burr Ridge, IL: McGraw Hill*, vol. 45, 1997.

[103] M. Kumar and S. K. Rath, ‘‘Microarray data classification using fuzzy k-nearest neighbor,’’ in *International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, November 2014, pp. 1032–1038.

[104] M. Warren and P. Walter, ‘‘A logical calculus of ideas immanent in nervous activity,’’ *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, December 1943.

[105] C. Bishop, ‘‘Improving the generalization properties of radial basis function neural networks,’’ *Neural computation*, vol. 3, no. 4, pp. 579–588, 1991.

[106] D. F. Specht, ‘‘Probabilistic neural networks,’’ *Neural networks*, vol. 3, no. 1, pp. 109–118, 1990.

[107] C. Cortes and V. Vapnik, ‘‘Support-vector networks,’’ *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[108] K. J. Yogendra and K. B. Santosh, ‘‘Min-max normalization based data perturbation method for privacy protection,’’ *International Journal of Computer and Communication Technology*, vol. 2, no. 8, pp. 45–50, October 2001.

[109] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri *et al.*, ‘‘Molecular classification of cancer: class discovery and class prediction by gene expression monitoring,’’ *science*, vol. 286, no. 5439, pp. 531–537, October 1999.

[110] L. J. van’t Veer, H. Dai, M. J. Van De Vijver, Y. D. He, A. A. Hart, M. Mao, H. L. Peterse, K. van der Kooy, M. J. Marton, and S. G. Witteveen, Anke T, ‘‘Gene expression profiling predicts clinical outcome of breast cancer,’’ *nature*, vol. 415, no. 6871, pp. 530–536, January 2002.

[111] E. F. Petricoin III, A. M. Ardekani, B. A. Hitt, P. J. Levine, V. A. Fusaro, S. M. Steinberg, G. B. Mills, C. Simone, D. A. Fishman, E. C. Kohn *et al.*, ‘‘Use of proteomic patterns in serum to identify ovarian cancer,’’ *The lancet*, vol. 359, no. 9306, pp. 572–577, February 2002.

[112] D. S. Broomhead and L. David, ‘‘Multivariable functional interpolation and adaptive networks,’’ *Complex Systems*, vol. 2, no. 3, pp. 321–355, 1988.

[113] J. Moody and J. Darken C, ‘‘Fast learning in networks of locally-tunes processing units,’’ *Neural Computation*, vol. 1, no. 2, pp. 281–294, Summer 1989.

[114] D. F. Specht, ‘‘Probabilistic neural networks,’’ *Neural Networks*, vol. 3, no. 1, pp. 109 – 118, June 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/089360809090049Q

[115] Y. F. Leung and D. Cavalieri, ‘‘Fundamentals of cdna microarray data analysis,’’ *TRENDS in Genetics*, vol. 19, no. 11, pp. 649–659, November 2003.

[116] 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.

[117] G. Lee, C. Rodriguez, and A. Madabhushi, ‘‘Investigating the efficacy of nonlinear dimensionality reduction schemes in classifying gene and protein expression studies,’’ *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, vol. 5, no. 3, pp. 368–384, October 2008.

[118] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2003.

[119] G.-B. Huang, D. H. Wang, and Y. Lan, ‘‘Extreme learning machines: a survey,’’ *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[120] M. Tipping, ‘‘Relevance vector machine,’’ Oct. 14 2003, uS Patent 6,633,857.

[121] T. Fletcher, ‘‘Relevance vector machines explained,’’ *University College London: London, UK*, 2010.

[122] L.-X. Wang and J. M. Mendel, ‘‘Generating fuzzy rules by learning from examples,’’ *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 22, no. 6, pp. 1414–1427, November/December 1992.

[123] T. Takagi and M. Sugeno, ‘‘Fuzzy identification of systems and its applications to modeling and control,’’ *Systems, Man and Cybernetics, IEEE Transactions on*, no. 1, pp. 116–132, Jan/Feb 1985.

[124] S. N. Sivanandam, S. Sumathi, and S. N. Deepa, *Introduction to Fuzzy Logic using MATLAB*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[125] B. Schölkopf, A. Smola, and K.-R. Müller, ‘‘Nonlinear component analysis as a kernel eigenvalue problem,’’ *Neural computation*, vol. 10, no. 5, pp. 1299–1319, March 1998.

[126] Y. K. Jain and S. K. Bhandare, ‘‘Min max normalization based data perturbation method for privacy protection,’’ *International Journal of Computer & Communication Technology (IJCCT)*, vol. 2, no. 8, pp. 45–50, October 2011.

[127] M. Tipping, ‘‘Sparse Bayesian Learning and The Relevance Vector Machine,’’ *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[128] J. C. MacKay, ‘‘The Evidence Framework Applied to Classification Networks,’’ *Neural Computation*, vol. 4, no. 5, pp. 720–736, 1992.

[129] D.-W. Kim, K. Lee, D. Lee, and K. H. Lee, ‘‘A kernel-based subtractive clustering method,’’ *Pattern Recognition Letters*, vol. 26, no. 7, pp. 879–891, May 2005.

[130] S. Chiu, ‘‘Fuzzy model identification based on cluster estimation,’’ *Journal of intelligent and Fuzzy systems*, vol. 2, no. 3, pp. 267–278, June 1994.

[131] B. Schölkopf, R. Herbrich, and A. J. Smola, ‘‘A generalized representer theorem,’’ in *Computational learning theory*. Springer, 2001, pp. 416–426.

[132] G. Kimeldorf and G. Wahba, ‘‘Some results on tchebycheffian spline functions,’’ *Journal of Mathematical Analysis and Applications*, vol. 33, no. 1, pp. 82–95, January 1971.

[133] R. Rosipal and L. J. Trejo, ‘‘Kernel partial least squares regression in reproducing kernel hilbert space,’’ *The Journal of Machine Learning Research*, vol. 2, pp. 97–123, December 2002.

[134] L. I. Kuncheva, ‘‘How good are fuzzy if-then classifiers?’’ *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 30, no. 4, pp. 501–509, August 2000.

[135] M. Sokolova and G. Lapalme, ‘‘A systematic analysis of performance measures for classification tasks,’’ *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, July 2009.

[136] C. Catal, ‘‘Performance evaluation metrics for software fault prediction studies,’’ *Acta Polytechnica Hungarica*, vol. 9, no. 4, pp. 193–206, 2012.

[137] E. F. Petricoin III, A. M. Ardekani, B. A. Hitt, P. J. Levine, V. A. Fusaro, S. M. Steinberg, G. B. Mills, C. Simone, D. A. Fishman, E. C. Kohn *et al.*, ‘‘Use of proteomic patterns in serum to identify ovarian cancer,’’ *The lancet*, vol. 359, no. 9306, pp. 572–577, February 2002.

[138] 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.

[139] L. J. van’t Veer, H. Dai, M. J. Van De Vijver, Y. D. He, A. A. Hart, M. Mao, H. L. Peterse, K. van der Kooy, M. J. Marton, A. T. Witteveen *et al.*, ‘‘Gene expression profiling predicts clinical outcome of breast cancer,’’ *nature*, vol. 415, no. 6871, pp. 530–536, November 2002.

[140] M. Consortium, ‘‘The microarray quality control (maqc)-ii study of common practices for the development and validation of microarray-based predictive models,’’ *Nature biotechnology*, vol. 28, no. 8, pp. 827–838, July 2010.

[141] T. Haferlach, A. Kohlmann, L. Wieczorek, G. Basso, G. Te Kronnie, M.-C. Béné, J. De Vos, J. M. Hernández, W.-K. Hofmann, and K. I. Mills, ‘‘Clinical utility of microarray-based gene expression profiling in the diagnosis and subclassification of leukemia: report from the international microarray innovations in leukemia study group,’’ *Journal of Clinical Oncology*, vol. 28, no. 15, pp. 2529–2537, May 2010.

[142] A. Kohlmann, T. J. Kipps, L. Z. Rassenti, J. R. Downing, S. A. Shurtleff, K. I. Mills, A. F. Gilkes, W.-K. Hofmann, G. Basso, M. C. Dell'Orto *et al.*, ''An international standardization programme towards the application of gene expression profiling in routine leukaemia diagnostics: the microarray innovations in leukemia study prephase,'' *British journal of haematology*, vol. 142, no. 5, pp. 802–807, 2008.

[143] A. Kühnl, N. Gökbuget, A. Stroux, T. Burmeister, M. Neumann, S. Heesch, T. Haferlach, D. Hoelzer, W.-K. Hofmann, E. Thiel *et al.*, ''High baalc expression predicts chemoresistance in adult b-precursor acute lymphoblastic leukemia,'' *Blood*, vol. 115, no. 18, pp. 3737–3744, 2010.

[144] K. I. Mills, A. Kohlmann, P. M. Williams, L. Wieczorek, W.-m. Liu, R. Li, W. Wei, D. T. Bowen, H. Loeffler, J. M. Hernandez *et al.*, ''Microarray-based classifiers and prognosis models identify subgroups with distinct clinical outcomes and high risk of aml transformation of myelodysplastic syndrome,'' *Blood*, vol. 114, no. 5, pp. 1063–1072, 2009.

[145] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2000.

[146] J. A. Suykens and J. Vandewalle, ''Least squares support vector machine classifiers,'' *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[147] O. L. Mangasarian and E. W. Wild, ''Proximal support vector machine classifiers,'' in *Proceedings Knowledge Discovery and Data Mining KDD, San Francisco, CA, USA*. Citeseer, 2001.

[148] Y. Tang and H. H. Zhang, ''Multiclass proximal support vector machines,'' *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 339–355, 2006.

[149] G. Fung and O. L. Mangasarian, ''Incremental support vector machine classification.'' in *SDM*. SIAM, 2002, pp. 247–260.

[150] G. M. Fung and O. L. Mangasarian, ''Multicategory proximal support vector machine classifiers,'' *Machine Learning*, vol. 59, no. 1-2, pp. 77–97, 2005.

[151] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[152] C.-Y. Lin, C.-H. Tsai, C.-P. Lee, and C.-J. Lin, ''Large-scale logistic regression and linear support vector machines using spark,'' in *IEEE International Conference on Big Data, Washington DC, USA*. IEEE, 2014, pp. 519–528.

[153] R. Hecht-Nielsen, ''Theory of the backpropagation neural network,'' in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1989, pp. 593–605.

[154] K. Hornik, M. Stinchcombe, and H. White, ''Multilayer feedforward networks are universal approximators,'' *Neural networks*, vol. 2, no. 5, pp. 359–366, March 1989.

[155] A. Bors, ''Introduction of the radial basis function (rbf) networks,'' in *Online symposium for electronics engineers*, vol. 1, no. 1, 2001, pp. 1–7.

[156] D. S. Broomhead and D. Lowe, ''Radial basis functions, multi-variable functional interpolation and adaptive networks,'' DTIC Document, Tech. Rep., 1988.

[157] J. Moody and C. J. Darken, ''Fast learning in networks of locally-tuned processing units,'' *Neural computation*, vol. 1, no. 2, pp. 281–294, 1989.

[158] M. Kumar and S. K. Rath, ''Classification of microarray using mapreduce based proximal support vector machine classifier,'' *Knowledge-Based Systems*, vol. 89, pp. 584–602, 2015.

Index

Dissemination

Internationally indexed journals (*Web of Science, SCI, Scopus, etc.*)[1]

1. **Mukesh Kumar**, Nitish Kumar Rath, and Santanu Kumar Rath. ‘‘Analysis of microarray leukemia data using an efficient MapReduce-based K-nearest-neighbor classifier.” *Journal of biomedical informatics, Elsevier* 60 (2016): pp. 395-409, DOI: http://dx.doi.org/10.1016/j.jbi.2016.03.002.

2. **Mukesh Kumar**, and Santanu Kumar Rath. ‘‘Classification of microarray using MapReduce based proximal support vector machine classifier” *Knowledge-Based Systems, Elsevier* 89 (2015): pp. 584-602, DOI: http://dx.doi.org/10.1016/j.knosys.2015.09.005.

3. **Mukesh Kumar**, Sandeep Singh and Santanu Kumar Rath, ‘‘Classification of Microarray Data using Extreme Learning Machine Classifier,” *International Journal of Information Processing (IJIP)*, vol. 9, no. 3, pp. 1-16, 2015.

4. **Mukesh Kumar** and Santanu Kumar Rath, ‘‘Classification of Microarray Data Using Kernel Fuzzy Inference System,” *International Scholarly Research Notices, Hindawi*, vol. 2014, Article ID 769159, 18 pages, 2014. DOI:10.1155/2014/769159

Other journals and Book chapters [1]

1. **Mukesh Kumar**, Santanu Kumar Rath, ‘‘Classification of microarray data using machine learning techniques,” *Emerging Trends in Applications and Infrastructures for Computational Biology, Bioinformatics, and Systems Biology: Systems and Applications*, 1st Edition, Book ISBN :9780128042038 Publisher: Elsevier (MK imprints).

[1] Articles already published, in press, or formally accepted for publication.

Conferences [1]

1. Ransingh Biswajit Ray, **Mukesh Kumar**, Anand Tirkey, and Santanu Kumar Rath. ''Scalable Information Gain Variant on Spark Cluster for Rapid Quantification of Microarray." *Procedia Computer Science*, Elsevier 93 (2016): pp. 292-298, Kochi, India.

2. Ransingh Biswajit Ray, **Mukesh Kumar**, and Santanu Kumar Rath. ''Fast in-memory cluster computing of sizeable microarray using spark." 5^{th} *International Conference on Recent Trends in Information Technology (ICRTIT)*, pp. 1-6. IEEE, 2016, Channai, India.

3. Ransingh Biswajit Ray, **Mukesh Kumar**, and Santanu Kumar Rath. ''Fast Computing of Microarray Data Using Resilient Distributed Dataset of Apache Spark." *In Recent Advances in Information and Communication Technology* 2016, pp. 171-182. Springer International Publishing, 2016, Bangkok.

4. **Mukesh Kumar**, Nitish Kumar Rath, Amitav Swain, and Santanu Kumar Rath. ''Feature Selection and Classification of Microarray Data using MapReduce based ANOVA and K-Nearest Neighbor.", *In Eleventh International Multi Conference on Information Processing (IMCIP'15)*,*Procedia Computer Science* Elsevier, 54 (2015): pp. 301-310, Bangalore, India.

5. **Mukesh Kumar**, and Santanu Kumar Rath. ''Meta-heuristic search based gene selection and classification of microarray data." *In* 12^{th} *Annual IEEE India Conference (INDICON)*, pp. 1-6. IEEE, 2015, Delhi, India.

6. **Mukesh Kumar**, Sandeep Singh, and Santanu Kumar Rath. ''Classification of Microarray Data using Functional Link Neural Network", 3^{rd} *International Conference on Recent Trends in Computing (ICRTC),* Noida, India, 2015.

7. **Mukesh Kumar** and Santanu Kumar Rath, ''Microarray data classification using Fuzzy K-Nearest Neighbor", *In* 1^{st} *International Conference on Contemporary Computing and Informatics (IC3I), 2014* pp. 1032-1038. IEEE, 2014, Mysore, India.

8. **Mukesh Kumar** and Santanu Kumar Rath, ''Classification of microarray data using Fuzzy inference system",. *In* 4^{th} *International Conference on Recent Trends in Information Technology (ICRTIT),* pp. 1-8, April, 2014, IEEE, Channai, India.

Article under preparation [2]

1. **Mukesh Kumar**, Ransingh Biswajit Ray, and Santanu Kumar Rath. ‘‘A scalable implementation of Neural Networks for Big Data classification using Spark cluster.” *IEEE Transaction on Big Data*, IEEE (Communicated)

2. **Mukesh Kumar**, Nitish Kumar Rath, and Santanu Kumar Rath. ‘‘MapReduce based Feature Selection and Classification of Microarray Data”, *International Journal of Machine Learning and Cybernetics (JMLC),* Springer (Communicated)

[2] Articles under review, communicated, or to be communicated.

Resume

Mukesh Kumar

Department of Computer Science and Engineering,
National Institute of Technology Rourkela,
Rourkela – 769 008, Odisha, India.
Mob: +91 89841 42557
Email: mkyadav262@gmail.com

Qualification

- PhD (CSE) (*Continuing*)
  National Institute of Technology Rourkela
- M.Tech. (CSE)
  National Institute of Technology Rourkela, [8.63 CGPA]
- B. Tech. (CSE)
  Cochin University of Science and Technology (CUSAT), Kochi, [74.50%]
- Internship at ABB, INCRC, Bangalore, INDIA [01^{st} Mar 2016 – 01^{st} Sep 2016]

Publications

- Journals: 07
- Conferences: 12
- Book Chapters: 01

Permanent Address

At-Naurangia, PO- Semrahia, PS- Lakhaura
East Champaran 845 302, Bihar.

Date of Birth

08^{th} October 1987