2021-05-01

Gene Selection And Classification In High-Throughput Biological Data With Integrated Machine Learning Algorithms And Bioinformatics Approaches

Abhijeet R Patil
University of Texas at El Paso

GENE SELECTION AND CLASSIFICATION IN HIGH-THROUGHPUT BIOLOGICAL DATA WITH INTEGRATED MACHINE LEARNING ALGORITHMS AND BIOINFORMATICS APPROACHES

ABHIJEET R PATIL

Doctoral Program in Computational Science

APPROVED:

Ming-Ying Leung, Ph.D., Chair

Sourav Roy, Ph.D., Co-Chair

Jonathon E Mohl, Ph.D.

Jorge Cervantes, M.D., Ph.D.

Stephen Crites, Ph.D.
Dean of the Graduate School

to my

MOTHER and FATHER

with love

GENE SELECTION AND CLASSIFICATION IN HIGH-THROUGHPUT BIOLOGICAL DATA WITH INTEGRATED MACHINE LEARNING ALGORITHMS AND BIOINFORMATICS APPROACHES

by

ABHIJEET R PATIL, M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Computational Science Program

THE UNIVERSITY OF TEXAS AT EL PASO

May 2021

Acknowledgements

I would like to express my most sincere gratitude to my advisor, Dr. Ming-Ying Leung for her wise guidance, constant encouragement, and advice throughout my research. I would like to thank my co-advisor, Dr. Sourav Roy for his constant support, supervision in my study and thesis. I sincerely appreciate their generous time and ideas. I am indebted and thankful to both of my advisors.

I would also like to thank my committee members; Dr. Jorge Cervantes of Texas Tech Health Science Center and Dr. Jonathon E Mohl of Mathematical Sciences at UTEP. Their encouragement, insightful comments, additional guidance were invaluable to my research.

I will forever be thankful to my former mentor, Dr. Sangjin Kim. I have greatly benefited from his expertise. I would have not reached this far without receiving help and support from him.

I would also like to thank my former dissertation committee member Dr. Lin Li for his valuable inputs during the first two years of my study.

I also want to thank faculty, staff, and students in the Computational Science program for their help and encouragement.

Finally, I would like to convey my deepest feelings of gratitude towards my parents and my wife for their constant inspiration, loving support and encouragement in all my decisions. I undoubtedly could not have done this without you all.

Abstract

With the rise of high throughput technologies in biomedical research, large volumes of expression profiling, methylation profiling, and RNA-sequencing data are being generated. These high-dimensional data have large number of features with small number of samples, a characteristic called the "curse of dimensionality." The selection of optimal features, which largely affects the performance of classification algorithms in machine learning models, has led to challenging problems in bioinformatics analyses of such high-dimensional datasets. In this work, I focus on the design of two-stage frameworks of feature selection and classification and their applications in multiple sets of colorectal cancer data.

The first algorithm developed was a combination of resampling based least absolute shrinkage and selection operator (lasso) feature selection (RLFS) and ensembles of regularized regression models (ERRM) capable of handling data with high correlation structures. The ERRM boosted the prediction accuracy with the top-ranked features obtained from RLFS. The second algorithm was a modified adaptive lasso method with normalized weights from various feature selection methods. Here, the genes were ranked based on their levels of statistical significance. The scores of the ranked genes were normalized and assigned as proposed weights to the adaptive lasso method to obtain the most significant genes known to be biologically related to the cancer type and helped attain higher classification performance. Lastly, we introduced a resampling of group lasso (glasso) feature selection method capable of ignoring the features unrelated to the response variable considering the group correlation among the features. These features, when applied on various classifiers, showed an increase in the classification accuracy.

We applied the above algorithms on both simulated and real data to show that our methods have better performance compared to existing ones. In the real data application, we combined machine learning with various bioinformatics tools, such as STRINGdb and Cytoscape, to explore 13 sets of microarray and RNA-seq data to identify hub genes in col-

orectal cancer. The results could be useful for suggesting further studies to reveal potential biomarkers that might lead to better cancer diagnoses and treatments.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the advances of high throughput technology in biomedical research, large volumes of high-dimensional data have been generated by different types of omics experiments. These data, including those from microarrays [Datta et al., 2007, Su et al., 2020, Wang et al., 2008] and RNA sequencing (RNA-seq) platforms [Stark et al., 2019], as well as DNA methylation studies [Bock, 2012, Moore et al., 2013, Sun and Wang, 2012] are being deposited [Hrdlickova et al., 2017, Kukurba and Montgomery, 2015, Moore et al., 2013, Sobek et al., 2006] in public databases such as gene expression omnibus (GEO) and the cancer genome atlas (TCGA).

The most common problem among all these different biological data types is the "curse of dimensionality", where the size of features becomes much larger than the number of samples ($p >> n$). Some of the other common challenges while dealing with high-throughput data is that they contain many redundant and unwanted features. To address these challenges, many two-stage machine learning frameworks of feature selection and classification have been proposed over the past few years [Elyasigomari et al., 2017, Kim and Kim, 2019, Sun et al., 2019]. In the first stage, feature selection (FS) methods are used to remove the redundant and unwanted noisy features, reducing the dimensionality of data. Next, significant features are given to classification algorithms which help in boosting the prediction performance.

There are various FS methods that rank the features upon their importance based on certain scores [Chandrashekar and Sahin, 2014, Kim and Kim, 2019, Kim and Halabi, 2016, Li et al., 2017a]. The optimal set of features are selected using sure independence screening (SIS) [Fan and Lv, 2008] condition. Some of the most popular rank-based FS methods used in bioinformatics are Information Gain [Quinlan, 1993], Chi-square

[Iguyon and Elisseeff, 2003], Fisher score [UM, 2013], and Minimum Redundancy Maximum Relevance [Peng et al., 2005]. Another type of FS method is called subset methods [Ditzler et al., 2015]. These methods select the subset of features with some pre-determined threshold based on some criteria. However, these methods need more computational time in high-dimensional data settings and lead to an NP-hard problem [Su and Yang, 2008]. Some of the popular subset methods found in literature includes Boruta [Kursa and Rudnicki, 2010], Fisher score [UM, 2013] and Relief [Urbanowicz et al., 2018].

For the classification of high-throughput data, many popular parametric and non-parametric algorithms can be used. The parametric methods includes regularized regression models with different penalties such as least absolute shrinkage and selection operator (LASSO) [Tibshirani, 1996], Ridge [Marquardt and Snee, 1975], elastic net [Wang et al., 2019], smoothly clipped absolute deviation (SCAD) [Fan and Li, 2001], and minimax concave penalty (MCP) [Zhang, 2010]. The LASSO and Ridge methods are based on L1 and L2 penalties, respectively. Elastic net is a combination of L1 and L2 penalties. SCAD and MCP are based on non-concave and concave penalties. These regularized models are predominantly common in high-throughput studies [Hastie et al., 2009]. The non-parametric models include random forests [Breiman, 2001], AdaBoost [Freund, 2001], support vector machines [Hearst et al., 1998], and naive Bayes [Rish, 2001]. The random forests and AdaBoost methods were built on decision trees, and the support vector machines were based on the idea of hyperplanes.

The above-discussed FS methods and individual classification algorithms have been widely used in the field of machine learning, and bioinformatics
[Chandrashekar and Sahin, 2014, Hastie et al., 2009]. However, in a highly correlated gene expression data set, most FS and classification methods do not perform well in terms of gene selection and classification accuracy [Bourgon et al., 2010a, Kim and Kim, 2019, Lu et al., 2011a]. Besides, there are several unique challenges associated with each high-throughput data type. For example, DNA methylation data has group correlation

among the CpG sites and most of the machine learning algorithms do not perform well [Patil et al., 2020a]. In RNA-seq data, where expression levels are in the form of counts, many machine learning algorithms cannot be applied directly, and the observed values needs to be transformed to a continuous form. This dissertation will focus on addressing these various challenges in the microarray, RNA-seq, and DNA methylation data by developing machine learning methods.

Additionally, we also aim at gaining biological insights into different stages of colorectal cancer using various bioinformatics analyses. Here, I will first identify the hub genes, then perform enrichment analysis, and finally build protein-protein interaction networks to look for potential biomarkers that may help promote early diagnosis and treatment of colorectal cancer.

1.1 Specific aims

The "curse of dimensionality" is a major issue while solving any high dimensional problem in multi-omics research [Almugren and Alshamlan, 2019, Kumar et al., 2015, Michiels et al., 2011]. Most of the existing machine learning algorithms do not perform well on high-throughput data [Bourgon et al., 2010a, Lu et al., 2011a]. Choosing the best performing gene selection and classification method on microarray and RNA-seq experimental data becomes challenging [Kim and Kim, 2019]. This challenge prompted me to study, investigate, and develop several gene selection and classification algorithms that achieve best performance in terms of selecting significant genes and classification accuracy. Three specific aims are described below to overcome the challenges in different biological data sets.

Specific Aim 1. Development and application of machine learning algorithms in microarray gene expression data

In my studies on cancer microarray data to attain best classification performance, the first algorithm I developed focused on a two-stage approach of gene selection and classifica-

tion. In the first stage, I developed a resampling technique on lasso method to identify the most differentially expressed genes. Next, these significantly expressed genes were considered as important features and applied on the classifiers. Here, I developed the ensembles model that was built with regularized regression models such as lasso, adaptive lasso, elastic net, SCAD, and MCP. To compare the performance of the proposed methods with the current popular machine learning models, I conducted simulation studies with different correlation scenarios based on compound symmetry and applied the gene expression data that was obtained from GEO.

The second algorithm I developed focused on modifying the weights of adaptive lasso method. The traditional adaptive lasso method use the weights from ridge regression. Previous studies have also modified the weights by using the marginal maximum likelihood estimation method to generate weights and assign them to adaptive lasso method. These studies have shown an increase in performance of prediction on colon cancer data. We introduce a new variant of adaptive lasso by assigning normalized weights using the ranking-based gene selection methods for selecting key genes. The prediction accuracy was expected to be increased with the key genes selected in model. I generated synthetic data using the auto-regressive correlation and also used the colon cancer microarray data to validate the performance of the proposed models.

Specific Aim 2. Improving the classification performance in DNA-methylation data

The DNA methylation (DNAm) data is slightly different compared to normal gene expression data. The expression values lies between 0 and 1 and are highly correlated. There is a lack of literature on the application of machine learning algorithms on the DNAm data [Patil et al., 2020a, Zhuang et al., 2012]. In this part, I first analyzed the performance of the popular machine learning algorithms on both simulation and a real prostate cancer data set from GEO. Most of the algorithms could not perform well because of the group correlation among the data. Therefore, I developed a resampling-based group lasso method and applied it to both synthetic and real DNAm data to compare the performance of the

proposed and existing models.

Specific Aim 3. Identification of hub genes and key pathways in colorectal cancer

Colorectal cancer (CRC) is the third most common cancer that contributes to cancer-related morbidity. However, the expression of genes in different phases of CRC is largely unknown. There is also little known about the role of stress-survival pathways in CRC. We sought to discover the hub genes and identify their role in several key pathways, including oxidative stress and apoptosis in different stages of CRC. I collected eleven microarray and two RNA-seq datasets related to the colorectal adenoma, adenocarcinoma, and carcinoma groups. There is a need for investigation on how the genes are differentially expressed, functionally enriched, and interact with each other. The hub DEGs revealed from this study may be biomarkers and may explain the CRC development and progression mechanisms. Also, to validate the performance of the resampling-based lasso feature selection (RLFS) algorithm from specific Aim 1, I applied the RLFS algorithm on 13 datasets related to CRC to identify the top significant genes and compared the results with the differentially expressed genes found through traditional approach of linear models for microarray data (LIMMA) [Ritchie et al., 2015].

1.2 Overview of dissertation

The dissertation is organized as follows. Chapter 2, describes the background information about different types of high-throughput biological data, feature selection methods, and classification algorithms. In Chapter 3, I will present the proposed two-stage gene selection and classification algorithm from the article [Patil and Kim, 2020]. Another proposed algorithm developed for microarray data [Patil et al., 2020c], which uses adaptive-lasso with weights based on normalized filtering scores, is discussed in Chapter 4. Then in Chapter 5, the proposed resampling-based group-lasso ranking algorithm for handling DNA methylation data is discussed [Patil et al., 2020b]. In Chapter 6, the identification of hub genes in

colorectal cancer through machine learning and bioinformatics analysis is explained. Here, the main goal is to identify potential biomarkers in colorectal cancer. Finally, in Chapter 7, overall conclusions and future research are presented.

Chapter 2

Literature Review

This chapter first describes the types of high-dimensional biological data. Next, the popular feature selection and supervised classification algorithms used in high-dimensional data are discussed. The various performance metrics used to evaluate the performance of various classification models are explained. Finally, we explain the various bioinformatics approaches used for analyzing genes.

2.1 Biological data

2.1.1 Expression profiling by microarray

Expression profiling by microarray is one of the standard approaches to understand gene expression, single nucleotide polymorphisms (SNPs), and diagnosis of disease [Agapito, 2019]. It strengthens the capability by comparing the expression and regulation of more than ten thousand genes in parallel [Agapito, 2019]. Different microarray technologies, such as, short oligonucleotides (Affymetrix, NimblGen), long oligonucleotides (Agilent, Illumina), and spotted cDNA are used to generate gene expression data [Page et al., 2007]. HG-U133_Plus_2, Agilent-014850, GPL15207 are some of the microarray platforms used for studies in humans. These expression profiling platforms measures the mRNA expression levels of thousands of genes or the entire genome in a cell at any given moment [Metsis et al., 2004]. The expression profiling data is used to study the effects of some treatment and diseases [Fielden and Zacharewski, 2001]. For example, the gene expression patterns can be compared between the infected cells and healthy cells or tissues through the microarray-based gene expression profiling technique. Some of the key

steps involved in analyzing the microarray data are (1) feature extraction, (2) quality control, (3) normalisation, (4) differential expression analysis, and (5) biological interpretation [Huerta and Burke, 2016]. The expression profiling platforms yields tens of thousands of features referred as genes on often tens of patients. A typical microarray data include around less than hundred samples and the count of genes ranges from 28000 to 57000.

2.1.2 Methylation profiling by microarray

DNA methylation (DNAm) can be defined as the addition of a methyl (CH3) group to DNA and it occurs specifically at the cytosine part of the cytosine-guanine dinucleotides (CpG) [Moore et al., 2013, Robertson, 2005]. DNAm plays a key and active role in regulating gene expression and modifying the function of regulatory elements [Moore et al., 2013]. The DNAm makes the cell specialized and maintain the unique traits throughout the life of an organism, equips a mechanism for response to external stimuli, and represses the detrimental expression of viral genes and other non-host DNA elements [Moore et al., 2013, Portela and Esteller, 2010]. Abnormal DNA methylation, including hypermethylation and hypomethylation, can lead to the dysregulation of cellular processes and its effect on gene expression values have been indicated in many diseases, including cancers [Portela and Esteller, 2010], type 2 diabetes [Dayeh et al., 2014], and chronic kidney disease [Ko et al., 2013]. The Illumina Infinium HumanMethylation27 Beadchip assay (IIHM27K) is one of the most widely used microarrays for the genome-wide DNA methylation analysis [Baker, 2010]. It quantitatively measures 27578 CpG sites (CpG dinucleotides) that spans over 14475 consensus coding sequence genes [Pidsley et al., 2016]. The CpG sites maps to the promoter regions of the genes with an average coverage of 2 CpG sites per gene and extensive coverage ranging from 3 to 20 CpG sites per gene for cancer-related genes [Bibikova et al., 2009]. The genome-wide span of IIHM27K and the 12-sample per array format showed a key advancement over the previously designed methods that were low-throughput and limited to a small number of genetic loci [Pidsley et al., 2016]. The size of the DNAm data typically includes around 27000 features known as CpG sites and

fewer than 100 samples which leads to a high-dimensional problem.

2.1.3 Expression profiling by high throughput sequencing

Expression profiling with high-throughput sequencing is a next-generation sequencing (NGS) technology. One of the most commonly used advanced NGS technology in genomic research is RNA sequencing (RNA-seq). Many studies have been conducted to compare the differences between the differentially expressed genes generated from microarray and RNA-seq [Xu et al., 2013, Hung and Weng, 2017, Zhao et al., 2014]. RNA-seq is also an alternative to microarray gene expression profiling [Merrick et al., 2013]. RNA-seq technology will be used more because it allows sequencing of whole transcriptome while the microarray allows only predefined transcripts/gene profiles through hybridization [Qian et al., 2014, Mantione et al., 2014]. However, the RNA-seq technology when compared with microarray has several disadvantages. For example, although there are many computational tools available there is a lack of standardized protocols available for optimal analysis [Rao et al., 2019, Chandramohan et al., 2013, Hayer et al., 2015]. Furthermore, RNA-seq datasets are much larger compared to microarray datasets. Therefore, the bioinformatics analysis becomes computationally intensive [Robinson and Oshlack, 2010, Rao et al., 2019, Esteller, 2011]. The Illumina HiSeq 2500 and MGISEQ-2000 are the most popular sequencing technology used to perform whole genome sequencing and generate RNA-seq data [Korostin et al., 2020]. There are several key steps carried out in RNA-seq data generation and analysis: (1) RNA isolation and extraction, (2) RNA-seq library preparation, (3) quality analysis and pre-processing of RNA-seq data, (4) alignment to the reference genome, (5) Measuring transcript abundance, and finally (6) differential expression analysis [Chatterjee et al., 2018].

2.2 Feature selection methods

Feature selection (FS) is a process of removing the noisy and redundant features from the data. It helps in boosting the performance of a classification algorithm not just in terms of accuracy, sensitivity, and specificity but also in reducing the computational time.

The FS methods can be divided into three different categories namely filter, embedded, and wrapper methods.

2.2.1 Filter Methods

The filter-based approaches are independent of the classification methods; therefore, they are computationally faster than the wrapper and embedded methods. The relevant features are selected based on distances, entropy, and uncertainty. There are many algorithms developed. Some of the best examples include Relief [Urbanowicz et al., 2018], which uses the distance-based metric function. The ReliefF [Kononenko, 1994], which is a modified version of Relief, is developed to handle the multi-class problems. The minimum redundancy maximum relevance (MRMR) [Peng et al., 2005] and mutual information-based feature selection method (MIFS) [Battiti, 1994] are the FS methods that rely on mutual information criteria. The mutual information is calculated between the individual feature and response label. The FS method conditional mutual information maximization (CMIM) [Fleuret, 2004] recursively chooses the features that provide the maximum mutual information with the response class. The Information gain, gain ratio, and symmetrical uncertainty are the FS methods that are based on the entropy models [Quinlan, 1986, Quinlan, 1993].

2.2.2 Embedded Methods

Embedded methods incorporate the FS process inside the classification method, which helps in performing the feature selection and classification simultaneously. It helps reduce the computational time than the wrapper method; however, it is expensive when compared to filter-based methods. The embedded method includes the pruning method,

built-in mechanism, and regularization methods. In the pruning method, all the features are considered during the training phase for building the classification model. The features with lower correlation with the response variable are removed recursively using the support vector machines. In the built-in FS method, the important features are ranked based on their importance. The variable importance (varImp) measure in the random forest (RF) method is the best example of a built-in measure. In the regularized methods, the penalized regression (PLR) models are based on penalties and are popular in high-dimensional data for variable selection and classification purposes. Some of the examples of the sparse variable selection models include lasso, adaptive lasso, elastic net, smooth clipped absolute deviation (SCAD), and minimax concave penalty (MCP).

2.2.3 Wrapper Methods

Wrapper selects the feature subsets using classification algorithm and searching techniques. The examples of the former approach include forward selection, backward elimination, and recursive feature elimination. The latter approach includes hill climbing and best-first search strategies using the decision trees and naive Bayes classifiers. The wrapper methods are computationally expensive because the features are selected based on the performance of the classifiers or the searching techniques, which is a recursive process.

In the following sections, I will discuss some of the popular used filter methods in gene expression studies such as fisher score, information gain, chi-square, and minimum redundancy maximum relevance [Pirooznia et al., 2008, Dash, 2020, Alkuhlani et al., 2017, Hira and Gillies, 2015, Duda et al., 1998].

2.2.4 Fisher score

It is supervised FS method. It assigns ranks to each feature based on the weights that are calculated distances between the data points and classes [UM, 2013]. The weights for the samples from the same class types are assigned similar values, and the weights for samples

from different categories are assigned different values [Dash, 2020]. The Fisher score for each feature is defined below:

$$F(j) = \frac{\sum_{k=1}^{c} n_k(\mu_k^j - \mu^j)^2}{\sum_{k=1}^{c} n_k(\sigma_k^j)^2} \tag{2.1}$$

$F(j)$ is the calculated fisher score for each feature j, class c= 0 or 1, μ_k^j is the mean of k-th class, σ_{iy} is the variance of k-th class, μ^j is the mean and σ^j is the variance of the whole dataset corresponding to feature j.

2.2.5 Information Gain

The information gain (IG) method [Quinlan, 1993] is simple, and one of the widely used FS methods. This univariate FS method is used to assess the quantity of information shared between the training feature set $x_j = (x_{j1}, x_{j2}, ..., x_{jp})$ for $j = 1, ..., t$, where t is the number of training samples, for $g = 1, 2, ...p$, where g is the feature in p number of features, and the response variable y_j. It provides an ordered ranking of all the features having a strong correlation with the response variable that helps to obtain good classification performance.

The information gain between the gth feature in x_j and the response variable y_j is given as follows:

$$\mathrm{IG}(x_j; y_j) = \mathrm{H}(x_j) - \mathrm{H}(x_j|y_j), \tag{2.2}$$

where $H(x_j)$ is entropy of x_j and $H(x_j|y_j)$ is entropy of x_j given y_j. The entropy [Li et al., 2017b] of x_j is defined by the following equation:

$$\mathrm{H}(x_j) = \sum_{g \in x_j} \pi(g) log(\pi(g)), \tag{2.3}$$

where g indicates discrete random variable x_j and $\pi(g)$ gives the probability of g on all values of x_j.

Given the random variable y_j, the conditional entropy of x_j is:

$$\mathrm{H}(x_j|y_j) = \sum_{y \in y_j} \pi(y) \sum_{g \in x_j} \pi(g|y) log(\pi(g|y)), \tag{2.4}$$

where $\pi(y)$ is the prior probability of y_j; $\pi(g|y)$ is conditional probability of g in a given y that shows the uncertainty of x_j given y_j.

$$\mathrm{IG}(x_j; y_j) = \sum_{g \in x_j} \sum_{y \in y_j} \pi(g, y) log \frac{\pi(g, y)}{\pi(g)\pi(y)}, \tag{2.5}$$

where $\pi(g, y)$ is the joint probability of g and y . IG is symmetric such that $\mathrm{IG}(x_j; y_j) = \mathrm{IG}(y_j; x_j)$, and is zero if the variables x_j and y_j are independent.

2.2.6 Chi-Square Test

The chi-square test (Chi2) belongs to the category of the non-parametric test, which is used mainly in determining the significant relation between two categorical variables. As part of the pre-processing step, we used the "equal interval width" approach to transform the numerical variables into categorical counterparts. The "equal interval width" algorithm first divides the data into q intervals of equal size. The width of each interval is defined as: $w = (max - min)/q$ and the interval boundaries are determined by: $min + w, min + 2w, ..., min + (q-1)w$.

The general rule in Chi2 is that the features have a strong dependency on the class labels selected, and the features independent of the class labels are ignored.

From the training set, $x_j = (x_{j1}, ...x_{jp})$, $g = 1, 2, ...p$, where g is every feature in p number of features. Given a particular feature g with r different feature values [Li et al., 2017b], the Chi2 score of that particular feature can be calculated as:

$$\tilde{\chi}^2(g) = \sum_{j=1}^{r} \sum_{s=1}^{p} \frac{(O_{js} - E_{js})^2}{E_{js}}, \tag{2.6}$$

where O_{js} is the number of instances with the j^{th} feature value given feature g. In addition, $E_{js} = \frac{O_{*s} O_{j*}}{O}$, where O_{j*} indicates the number of data instances with the feature value given feature g, O_{*s} denotes the number of data instances in r, and p is total number of features.

When two features are independent, the O_{js} is closer to the expected count E_{js}; consequently, we will have smaller Chi2 score. On the other hand, the higher Chi2 score implies

that the feature is more dependent on the response and it can be selected for building the model during training.

2.2.7 Minimum Redundancy Maximum Relevance

The minimum redundancy and maximum relevance method (MRMR) is built on optimization criteria of mutual information (redundancy and relevance); hence, it is also defined under mutual information based methods. If a feature has uniformly of expressions or if they are randomly distributed in different classes, its mutual information with such classes is null [Peng et al., 2005]. If a feature is expressed deferentially for different classes, it should have strong mutual information. Hence, we use mutual information as a measure of the relevance of features. MRMR also reduces the redundant features from the feature set. For a given set of features, it tries to measure both the redundancy among features and relevance between features and class vectors.

The redundancy and relevance are calculated based on mutual information, which is as follows: We know that, in the training set x_j, $g = 1, ..., p$ represents every feature in x_j and y_j is the response variable.

$$\mathrm{I}(g, y) = \sum_{g \in x_j} \sum_{y \in y_j} log \frac{\pi(g, y)}{\pi(g)\pi(y)}, \tag{2.7}$$

In the following equation, for simplicity, let us consider the training set x_j as X and response variable y_j as Y. The objective function is shown below:

$$J_{\mathrm{MRMR}}(X_S, Y) = \frac{1}{|S|} \sum_{i \in S} I(X_i, Y) - \frac{1}{|S|^2} \sum_{i,j \in S} I(X_i, X_j), \tag{2.8}$$

where S is the subset of selected features and X_i is the ith feature. The first term is a measure of relevance that is the sum of mutual information of all the selected features in the set S with respect to the output Y. The second term is measure of redundancy that is the sum of the mutual information between all the selected features in the subset S. By optimizing the Equation (2.8), we are maximizing the first term and minimizing the second term simultaneously.

2.2.8 Random forest variable importance

There are several types of variable importance measures by the RF algorithm. The first is derived based on statistical random permutation tests, and the other measure is calculated based on the training of the RF classifier [Strobl et al., 2007, Guyon et al., 2002].

The RF classifier, while training the model, performs the feature selection process implicitly by choosing a small number of important variables for classification purposes, which leads to higher performance. The implicit feature selection measure carried out by the RF algorithm is termed as Gini importance and is used as a symbol of feature relevance [Menze et al., 2009]. The scores are assigned to each feature and are ranked based on the importance measure given by the RF classifier.

At every node O in trees T, the ideal split is investigating with the Gini impurity $i(O)$. The Gini impurity $i(O)$ is a computational measure of how good a ideal split is dividing the training data samples of two classes in a given node. Among the total number of training samples X_t, Let $R_k = \frac{S_k}{X_t}$ be the small amount of data samples S_k from class $k = 0, 1$ at node O, The i_O is computed as follows:

$$i(O) = 1 - {R_1}^2 - {R_0}^2 \tag{2.9}$$

The decrease of Gini Δi that comes from dividing and sending the training data samples to two sub-nodes O_l and O_r where $R_l = \frac{S_l}{X_t}$ and $R_m = \frac{S_m}{X_t}$ are the corresponding fractions and the threshold t_ϕ on feature ϕ is calculated as:

$$\Delta i(O) = i(O) \; - \; R_l \; i(O_l) \; - R_m \; i(O_m) \tag{2.10}$$

During the exhaustive search conducted across all the variables ϕ present at the node and on all the thresholds t_ϕ, the pair ϕ, t_ϕ that leads to a maximal Δi is solved [Menze et al., 2009]. The decrease in Gini impurity [Menze et al., 2009] is obtained from this ideal split $\Delta i_\phi(O, E)$ and collected for all the nodes o in the binary trees E, individually for all the features ϕ.

$$GI(\phi) = \sum_{E} \sum_{O} \Delta i_{\phi}(O, E) \tag{2.11}$$

2.3 Supervised classification methods

There is a broad range of algorithms that can be applied for labeled data such as tree-based methods, discriminant analysis, and PLR models. The popular tree-based methods include random forest and adaptive boosting [Chen and Ishwaran, 2012]. The random forests are built on the concept of decision trees. The idea is to operate as an ensemble method instead of relying on a single method. It is based on the concept of bagging and majority voting. The Adaptive boosting method is an ensemble learning technique where it improves the single weak boosting algorithm through an iterative process. The support vector machines detect the maximum margin hyperplane by maximizing the distance between the hyperplane and the closest dot. The maximum margin indicates that the classes are well separable and correctly classified. The PLR models are built on the concept of penalties including L1 and L2.

In the following sub-sections, we will discuss the tree-based methods, support vector machines, and PLR models [Bielza et al., 2011].

2.3.1 Logistic Regression

Logistic regression (LR) is perhaps one of the primary and popular models used while dealing with binary classification problems [Liao and Chin, 2007]. Logistic regression for dealing with more than two classes is called multinomial logistic regression. The primary focus here is on the binary classification. Given the set of inputs, the output is a predicted probability that the given input point belongs to a particular class. The output is always within [0, 1]. Logistic regression is based on the assumption that the original input space can be divided into two separate regions, one for each class, by a plane. This plane helps to discriminate between the dots belonging to different classes and is called as linear

discriminant or linear boundary.

One of the limitations is the number of parameters that can be estimated needs to be smaller and should not exceed the number of samples.

2.3.2 Regularized Regression Models

Regularization is a technique used in logistic regression by employing penalties to overcome the limitations of dealing with high-dimensional data. Here, we discuss the PLR models such as lasso, adaptive lasso, elastic net, SCAD, and MCP. These five methods are included in the proposed ensembles of regularized regression models (ERRM) and also tested as independent classifiers for comparing performance with the ERRM.

Let us consider the training set as $xj = (x_{j1}, x_{j2}, ..., x_{jf})$ for $j = 1, ..., t$, where t is the number of training samples, the response variable y_j for the training set. The logistic regression equation:

$$\log\left(\frac{\pi(y_j = 1|x_j)}{1 - \pi(y_j = 1|x_j)}\right) = \beta_0 + \beta x_j, \tag{2.12}$$

where x_j is the training data, $j = 1....t$ and $\beta = (\beta_1...\beta_f)^T$ with f denoted as features in training data.

From logistic regression Equation (2.12), the log-likelihood estimator is shown as below:

$$l(\beta, y_j) = \sum_{j=1}^{t} \{y_j \log(\pi(y_j = 1|x_j)) + (1 - y_j)\log(1 - \pi(y_j = 1|x_j))\}. \tag{2.13}$$

Logistic regression offers the benefit by simultaneous estimation of the probabilities $\pi(x_j)$ and 1-$\pi(x_j)$ for each class. The criterion for prediction is $I\{\pi(y_j = 1|x_j) \geq 0.5\}$, where $I(\cdot)$ is an indicator function.

The parameters for PLR are estimated by minimizing above function:

$$\hat{\beta}_{\text{PLR}} = \underset{\beta}{\operatorname{argmin}} \left[- l(\beta, y_j) + p(\beta) \right], \tag{2.14}$$

where $p(\beta)$ is a penalty function, $l(\beta, y_j)$ is the log-likelihood function.

Lasso is a widely used method in variable selection and classification purposes in high dimensional data. It is one of the five methods used in the proposed ERRM for classification purposes. The lasso PLR method is defined below:

$$\hat{\beta}_{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \left[-l(\beta, y_j) + \lambda \sum_{j=1}^{f} |\beta_j| \right] \tag{2.15}$$

where f is the reduced number of features; λ is the tuning parameter that controls the strength of the L1 penalty.

The oracle property [Fan and Li, 2001] has consistency in variable selection and asymptotic normality. The lasso works well in subset selection; however, it lacks the oracle property. To overcome this, different weights are assigned to different coefficients: this describes a weighted lasso called adaptive lasso. The adaptive lasso (ALASSO) penalty is shown below:

$$\hat{\beta}_{\text{ALASSO}} = \underset{\beta}{\operatorname{argmin}} \left[-l(\beta, y_j) + \lambda \sum_{j=1}^{f} w_j |\beta_j| \right], \tag{2.16}$$

where f is the reduced number of features, λ is the tuning parameter that controls the strength of the L2 penalty, and w_j is the weight vector based on ridge estimator. The ridge estimator [Marquardt and Snee, 1975] uses the L2 regularization method which obtains the size of coefficients by adding the L2 penalty.

The elastic net (ENET) [Zou and Hastie, 2005] is the combination of lasso which uses the L1 penalty, and ridge which uses the L2 penalty. The sizable number of variables is obtained, which helps in avoiding the model turning into an excessively sparse model.

The ENET penalty is defined as:

$$\hat{\beta}_{\text{ENET}} = \underset{\beta}{\operatorname{argmin}} \left[-l(\beta, y_j) + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^{f} |\beta_j|^2 + \alpha \sum_{j=1}^{f} |\beta_j| \right) \right], \tag{2.17}$$

where λ is the tuning parameter that controls the penalty, f is the number of features, α is the mixing parameter between ridge $\alpha = 0$ and lasso $\alpha = 1$.

The smoothly clipped absolute deviation penalty (SCAD) [Fan and Li, 2001] is a sparse logistic regression model with a non-concave penalty function. It improves the properties of

the L1 penalty. The regression coefficients are estimated by minimizing the log-likelihood function:

$$\hat{\beta}_{\text{scad}} = \underset{\beta}{\text{argmin}} \left[-l(\beta, y_j) + \lambda \sum_{j=1}^{f} p_\lambda(\beta_j) \right]. \tag{2.18}$$

In Equation (2.18) the $p_\lambda(\beta_j)$ is defined by:

$$|\beta_i| \mathrm{I}_{(|\beta_j| \leq \lambda)} + \left(\frac{\{(c^2-1)\lambda^2 - (c\lambda - |\beta_j|)_+^2\} I(\lambda \leq |\beta_j|)}{2(c-1)} \right), \; c > 2 \text{ and } \lambda \geq 0 \; . \tag{2.19}$$

Minimax concave penalty (MCP) [Zhang, 2010] is very similar to the SCAD. However, the MCP relaxes the penalization rate immediately, while for SCAD, the rate remains smooth before it starts decreasing. The MCP equation is given as follows:

$$\hat{\beta}_{\text{mcp}} = \underset{\beta}{\text{argmin}} \left[-l(\beta, y_j) + \lambda \sum_{j=1}^{f} p_\lambda(\beta_j) \right]. \tag{2.20}$$

In Equation (2.20) the $p_\lambda(\beta_j)$ is defined as:

$$\Big(\frac{2c\lambda|\beta_j| - \beta_j^2}{2c} \Big) \mathrm{I}(|\beta_j| \leq c\lambda) + \Big(\frac{c\lambda^2}{2} \Big) \mathrm{I}(|\beta_j| > c\lambda), \text{ for } \lambda \geq 0 \text{ and } c > 1. \tag{2.21}$$

2.3.3 Random Forests

The random forest (RF) [Breiman, 2001] is an interpretive and straightforward method commonly used for classification purposes in bioinformatics. It is also known for its variable importance ranking in high dimensional data sets. RF is built on the concept of decision trees. Decision trees are usually more decipherable when dealing with binary responses. The idea of RF is to operate as an ensemble instead of relying on a single model. RF is a combination of a large number of decision trees where each tree has some random subset of features obtained from the data by allowing repetitions. This process is called bagging. The majority voting scheme is applied by aggregating all the tree models and obtaining one final prediction.

2.3.4 Support Vector Machines

Support vector machines (SVM) [Hearst et al., 1998] are well known amongst most of the mainstream algorithms in supervised learning. The main goal of a SVM is to choose a hyperplane that can best divide the data in the high dimensional space. This helps to avoid overfitting. The SVM detects the maximum margin hyperplane, the hyperplane that maximizes the distance between the hyperplane, and the closest dots [Li et al., 2010]. The maximum margin indicates that the classes are well separable and correctly classified. It is represented as a linear combination of training points. As a result, the decision boundary function for classifying points as to hyperplane only involves dot products between those points.

2.3.5 Adaboost

Adaboost is also known as adaptive boosting (AB) [Freund, 2001]. It improves the performance of a particular weak boosting classifier through an iterative process. This ensemble learning algorithm can be extensively applied to classification problems. The primary objective here is to assign more weights to the patterns that are harder to classify. Initially, the same weights are assigned to each training item. The weights of the wrongly classified items are incremented while the weights of the correctly classified items are decreased in each iteration. Hence, with the additional iterations and more classifiers, the weak learner is bound to cast on the challenging samples of the training set.

2.3.6 Naive Bayes

Naive Bayes is a probabilistic method that takes the benefits of probability theory and the Bayes theorem [Patil et al., 2020a, Friedman et al., 1997]. The rule of thumb is that all the features being classified needs to be independent. Given the reduced training data after FS stage, Let Y_t be denoted as the random variable indicating the class and X_r denotes random variable indicating the observed values. y is response and x represents a given

value in X_r. Given a new test sample X_k from the testing data X_t to classify, each class probability can be accessed by using the Bayes theorem:

$$p(Y_t = y|X_r = x) = \frac{p(Y_t = y)p(X_r = x|Y_t = y)}{p(X_r = x)} \tag{2.22}$$

Since, $p(X_r = x)$ is a common factor for a given sample, it can be ignored during the classification process [Soria et al., 2011]. The Naive Bayes classifier can be obtained by imposing the class-conditional independence among the features:

$$p(X_r = x|Y_t = y) = \prod_{t=1}^{u} p(X_r = x|Y_t = y) \tag{2.23}$$

We can use the probability density value $f(X_r = x|Y_t = y)$ replacing the probability value $p(X_r = x|Y_t = y)$. The class-conditional probability density $f(.|Y_t = y)$ for each attribute value and prior $p(Y_t = y)$ can be obtained from the learning phase. The final Naive Bayes classification model can be calculated based on the kernel density estimation as follows:

$$nb^* = \underset{y \in \Omega}{\operatorname{argmax}}\, p(Y_t = y) \prod_{t=1}^{u} f(X_r = x|Y_t = y) \tag{2.24}$$

2.4 Performance metrics

We evaluated the results of combinations of FS methods with the classifier using accuracy and geometric mean (Gmean). The metrics are detailed with respect to true positive (TP), true negative (TN), false negative (FN), and false positive (FP). The equations for accuracy and Gmean are as follows:

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} \\ \text{Gmean} &= \sqrt{\text{Sensitivity} \times \text{Specificity}}, \end{aligned} \tag{2.25}$$

where the sensitivity and specificity are given by:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad \text{and} \quad \text{Specificity} = \frac{TN}{TN + FP}. \tag{2.26}$$

2.5 Functional enrichment analysis

Functional enrichment analysis helps to gain mechanistic insight into the differentially expressed gene lists and co-expressed gene sets generated from multi-omics experiments [Müller et al., 2019, Zhang et al., 2020]. Identifying the relationship between genes in different function classification system allows to explore biological questions. It is important to characterize gene-function relations by mining the functional associations among gene sets [Zhang et al., 2020]. There are various types of databases that are designed for identifying gene function classification. The most popular databases that provide gene-function interpretations for the biologists are gene ontology (GO) [Carbon et al., 2019], Kyoto encyclopedia of genes and genomes (KEGG) [Kanehisa et al., 2012], Reactome [Fabregat et al., 2016], and DISEASE [Pletscher-Frankild et al., 2015]. There are several analytic approaches developed to decipher the biological significance of the specific gene sets from different gene-function databases [Zhang et al., 2020]. Over-representation analysis (ORA) is most widely used in exploring the gene sets [Khatri et al., 2012]. Pathway analysis tools are developed based on certain key statistical methods such as Fisher's exact, Hypergeometric, Chi-square, and Bayesian test [Huang et al., 2009a]. DAVID bioinformatics is the most commonly used database for gene set enrichment analysis [Huang et al., 2009b]. There are many other tools available that are also used for enrichment analysis such as Allenricher, Enrichr, GO-Elite, clusterProfilers, FunSet, g:Profiler, gene set enrichment analysis (GSEA), Ingenuity pathway analysis (IPA), Cytoscape, EnrichmentMap [Müller et al., 2019, Shi Jing et al., 2015, Zhang et al., 2020].

2.6 Protein-protein interactions

Protein-protein Interactions (PPI) are used for knowing the fundamental processes and molecular organization in living cells [Peng et al., 2017]. High-throughput experiments such as yeast two-hybrid assay [Ito et al., 2002], affinity purification mass spectrometry (AP-MS) [Köcher and Superti-Furga, 2007], genome-wide PPIs are developed for many organisms such as, Drosophila melanogaster and Homo sapiens [Giot et al., 2003, Rolland et al., 2014]. There are many publicly available PPI databases that collect the interactions derived from various sources such as high-throughput experiments, low-throughput experiments, and computational predictions [Dong and Provart, 2018]. These methods have their own limitations. High-throughput and computational predictions including machine learning algorithms outputs large number of PPIs quickly but are lower quality interactions [Braun et al., 2009, Li and Ilie, 2017, Sarkar and Saha,]. Low-throughput experiments generally develop high quality interactions but the error-rate that occur during curation process can also be high [Cusick et al., 2009].

The interactions collected from various sources are stored into databases. There are several PPI databases available such as botony array resource (BAR) [Toufighi et al., 2005], BioGrid [Stark et al., 2006], BioPlex [Huttlin et al., 2015], InnateDB [Breuer et al., 2013], Mentha [Calderone et al., 2013], and STRING [Szklarczyk et al., 2017]. STRING is the most widely used PPI database. The PPI networks are constructed using various computation tools that retrieve the interactions from PPI databases. Some of the computational tools used for network construction tools includes Cytoscape [Shannon et al., 2003], igraph, and NetworkX [Dong and Provart, 2018]. Among these tools Cytoscape is an open source tool that is widely used for molecular visualization and analysis [Dong and Provart, 2018].

Chapter 3

A two-stage resampling-based feature selection and ensembles-based classification approach in high-throughput data

The content of this chapter has been published in [Patil and Kim, 2020]

3.1 Introduction

The two-stage approach of handling gene expression data has gained popularity over the past years [Sun et al., 2019, Kim and Kim, 2019, Elyasigomari et al., 2017]. The feature selection (FS) approach is the first stage where most of the lowly expressed genes are removed. The significantly expressed genes are passed on to the classifiers that help improve the classifiers' prediction performance. Information gain (IG) [Quinlan, 1993], minimum redundancy maximum relevance (MRMR) [Peng et al., 2005], Chi-square (Chi2) [Iguyon and Elisseeff, 2003] are some of the FS methods popularly used in literature to reduce the dimensionality of the data by removing noisy and irrelevant features. There are several non-parametric and parametric algorithms widely used for the classification of microarray data. Some non-parametric models include random forests (RF), adaptive boosting(AdaBoost), and support vector machines (SVM) which are non-parametric models [Breiman, 2001, Hearst et al., 1998, Freund, 2001]. The parametric models have gained popularity in high-throughput studies [Hastie et al., 2009]. Some of the parametric models include least absolute selection shrinkage selection operator (LASSO) [Tibshirani, 1996] and ridge [Marquardt and Snee, 1975] that are based on L1 and L2 penalties. Adaptive lasso (ALASSO) which takes ridge method weights. Elastic net

(ENET) [Wang et al., 2019] that is a combination of L1 and L2 penalty. The other two parametric models are non-concave and concave penalty-based smooth clipped absolute deviation (SCAD) [Fan and Li, 2001] and minimum concave penalty (MCP) [Zhang, 2010], respectively. Although these individual FS and classification methods perform well, they lack in achieving higher prediction accuracy. To overcome the drawbacks of these individual methods, several ensemble-based classification algorithms have been proposed in literature [Dietterich, 2000, Opitz and Maclin, 1999]. The ensemble-based methods are bagging and aggregating methods that are employed to improve the accuracy of several "weak" classifiers [Datta et al., 2010, Breiman, 1996, Freund and Schapire, 1997]. Another tree-based method of classification by ensembles from random partitions (CERP) showed good performance but was computer-intensive [Ahn et al., 2007]. The ensembles of logistic regression models (LORENS) for microarray data were proven to be more efficient for classification [Ahn et al., 2007]. However, these models suffered from a decrease in performance because of the smaller number of significant variables in high-dimensional space due to random partitioning. To address these issues, there is a need to develop a novel two-stage approach FS and classification algorithm and compare the proposed two-stage method's performance with the other combinations of popular FS and classifiers by performing extensive simulation studies and a real data application. It is important to filter our redundant and irrelevant features through FS methods, which further reduces computational time and boosts classification accuracy.

This chapter introduces the combination of an ensemble classifier with an FS method—the resampling-based lasso feature selection (RLFS) method for ranking features and ensemble of regularized regression models (ERRM) for classification purposes. The resampling approach was proven to be one of the best FS screening steps in a high-dimensional data setting [Patil and Kim, 2020]. The RLFS uses the selection probability with lasso penalty, and the threshold for selecting the top-ranked features is set using the b-SIS condition, and these select features were applied to the ERRM to achieve the best prediction accuracy. The ERRM uses five individual regularization models, LASSO, ALASSO, ENET, SCAD,

and MCP. We compare the performance of two-stage RLFS-ERRM combination with other combinations of FS and classifiers. Some of the individual FS methods include IG, Chi2, and MRMR. The individual classifiers include LASSO, ALASSO, ENET, SCAD, MCP, ADABOOST, RF, LR, SVM[Patil and Kim, 2020]. These individual FS and classification methods are discussed earlier in Chapter 2

3.2 Data

3.2.1 Synthetic Data Setup

The data were generated based on a random multivariate normal distribution where the mean was assigned as 0, and the variance-covariance matrix $\sum_x$ adapts a compound symmetry structure with the diagonal items set to 1 and the off-diagonal items being ρ values.

$$\sum_x = \begin{pmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{pmatrix}_{p \times p} . \tag{3.1}$$

The class labels were generated using the Bernoulli trails with the following probability:

$$\pi_i(y_i = 1|x_i) = \frac{\exp(x_i\beta)}{1 + \exp(x_i\beta)}. \tag{3.2}$$

The data matrix $x_i \sim N_p(0, \sum_x)$ was generated using the random multivariate normal distribution, and the response variable y_i was generated by binomial distribution, as shown in Equations (3.1) and (3.2) respectively. For sufficient comparison of the performance of the model and subsidizing the effects of the data splits, all of the regularized regression models were built using the 10-fold cross-validation procedure, and the averages were taken over 100 partitioning times referred to as 100 iterations in this paper. The data generated are high-dimensional in nature with the number of samples, $n = 200$ and total features, $p = 1000$. The true regression coefficients were set to 25, which were generated using uniform distribution with the minimum and maximum values 2 and 4, respectively.

With this setup of high-dimensional data, we simulated three different types of data, each with correlation structures $\rho = 0.2$, 0.5, and 0.8 respectively. These values show the low, intermediate, and high correlation structures in the datasets which are significantly similar to what we usually see in the gene expression or others among many types of data in the field of bioinformatics [Kim and Kim, 2019]. At first, the data were divided randomly into training and testing sets with 75% and 25% of samples respectively; 75% of the training data was given to the FS methods, which ranked the genes concerning their importance, and then the top-ranked genes were selected based on b-SIS condition. The selected genes were applied in all the classifiers. For standard comparison and mitigating the effects of the data splitting, all of the regularized regression models were built using the 10-fold cross-validation; the models were assessed for testing the performance with the testing data using different evaluation metrics, and averages were taken over 100 splitting times referred to as 100 iterations.

3.2.2 Experimental Data Setup

To test the performance of the proposed combination of ERRM with RLFS, and compare it with the rest of the combinations of FS and classifiers, the gene expression data SMK-CAN-187 were analyzed. The data include 187 samples and 19,993 genes obtained from smokers, which included 90 samples from those with lung cancer and 97 samples from those without lung cancer. This data is high-dimensional, with the number of genes being 19,993. The preprocessing procedures are necessary to handle these high-dimensional data. At first, the data were randomly divided into training and testing sets with 75% and 25% of samples respectively. As the first filtering step, 75% of the training data were given to the marginal maximum likelihood estimator (MMLE), to overcome the redundant noisy features, and the genes were ranked based on their level of significance. The ranked significant genes were next applied to the FS methods along with the proposed RLFS method as the second filtering step, and a final list of truly significant genes was obtained. These significant genes were applied to all the classification models along with the proposed ERRM classifier.

All of the models were built using the 10-fold cross-validation. The average classification accuracy and Gmean of our proposed framework were tested using the test data. The above procedure was repeated for 100 times and the averages were taken.

3.3 The two-stage proposed RLFS-ERRM framework

3.3.1 The Resampling-Based Lasso Feature Selection

From [Kim and Kim, 2019], we see that the resampling-based FS is relatively more efficient in comparison to the other existing FS methods in gene expression data. The RLFS method is based on the lasso penalized regression method and the resampling approach employed to obtain the ranked important features using the frequency.

The least absolute shrinkage and selection operator (LASSO) [Tibshirani, 1996] estimator is based on L1-regularization. The L1-regularization method limits the size of coefficients pushes the unimportant regression coefficients to zero by using the L1 penalty. Due to this property, variable selection is achieved. It plays a crucial role in achieving better prediction accuracy along with the gene selection in bioinformatics.

$$\hat{\beta}_{\text{lasso}} = \underset{\beta}{\text{argmin}} \left[-\sum_{j=1}^{t} \{ y_j \log(\pi(y_j = 1|x_j)) + (1 - y_j) \log(1 - \pi(y_j = 1|x_j)) \} + \lambda \sum_{j=1}^{p} |\beta_j| \right]. \tag{3.3}$$

The selection probability $S(f_m)$ of the features based on the lasso is shown in the below equation.

$$S(f_m) = \frac{1}{R} \sum_{i=1}^{R} \frac{1}{L} \sum_{j=1}^{L} \mathrm{I}(\beta_{ijm} \neq 0), \text{ for } m = 1, 2, ..., p. \tag{3.4}$$

The b-SIS criteria to select the top k ranked features is defined by,

$$\left\lceil b \times \frac{n}{log(n)} \right\rceil, \tag{3.5}$$

where R is defined by the total number of resampling, L is total number of λ values, f_m is the feature indexed as i, p is total number of features, n is total number of samples,

and β_{ijm} is defined as regression coefficient of mth feature and $I()$ indicator variable. Each R number of resamples and L number of values of λ are considered to build the variable selection model. The 10-fold cross validation is considered while building the model.

After ranking the features using the RLFS method, we employ the b-SIS approach to select the top features based on Equation (3.5) where b is set to two. The number of true important variables selected among the top b-SIS ranked features is calculated in each iteration and the average of this is taken over 100 iterations.

3.3.2 The Ensembles of Regularized Regression Models

LASSO, ALASSO, ENET, SCAD, and MCP are the five individual regularized regression models included as base learners in our ERRM. The role of bootstrapped aggregation or bagging is to reduce the variance by averaging over an "ensemble" of trees, which will improve the performance of weak classifiers. $B = B_1^k, ..., B_M^k$ is the number of random bootstrapped samples obtained from reduced training set x_r with corresponding class label y_j. The five regularized regression models are trained on each bootstrapped sample B named sub-training data, leading to $5 \times B$ models. These five regularized models are then trained using the 10-fold cross-validation to predict the classes on the out of bag samples called sub-testing data where the best model fit in each of the five regularized regression model is obtained. Henceforth, in each of the five regularized models, the best model is selected and the testing data x_k is applied to obtain the final list of predicted classes for each of these models. For binary classification problems, in addition to accuracy, the sensitivity and specificity are primarily sought. The E evaluation metrics are computed for each of these best models of five regularized models. In order to get an optimized classifier using all the evaluation measures E is essential, and this is achieved using weighted rank aggregation. Here, each of the regularized models is ranked based on the performance of E evaluation metrics. The models are ranked based on the increasing order of performance; in the case of a matching score of accuracy for two or more models, other metrics such as sensitivity and specificity are considered. The best performing model among the five

models is obtained based on these ranks. This procedure is repeated to obtain the best performing model in each of the tree T. Finally, the majority voting procedure is applied over the T trees to obtain a final list of predicted classes. The test class label is applied to measure the final E measures for assessing the performance of the proposed ensembles. The Algorithm 1 defines the proposed ERRM procedure.

Algorithm 1 Proposed ERRM

Step 1: Obtain new training data x_r with most informative features using the proposed RLFS method.

Step 2: Draw bootstrap samples from x_r and apply them to each of the regularized methods to be fitted with 10-fold cross validation.

Step 3: Apply out of bag samples (OOB) not used in bootstrap samples to the above fitted models to choose the best model using E evaluation metrics.

Step 4: Repeat steps 2 and 3 until getting 100 bootstrap models.

Step 5: Apply testing set x_k to each of 100 models to aggregate votes of classification.

Step 6: Predict classification of each sample by the rule of majority voting in the testing set.

The complete workflow of the proposed RLFS-ERRM framework is shown in Figure 3.1.

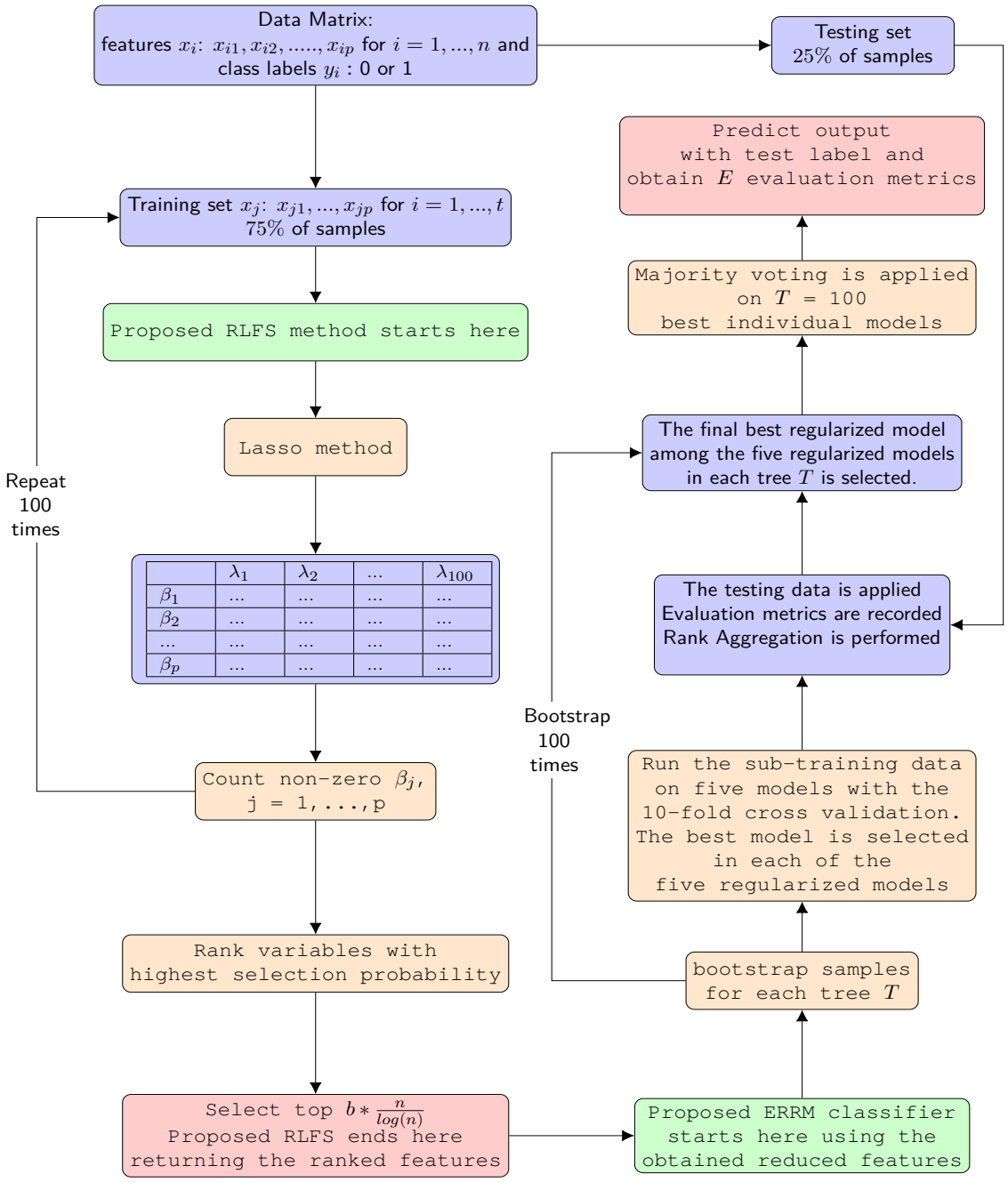

Figure 3.1: The complete workflow depicting the proposed combination of RLFS-ERRM framework.

3.4 Results

3.4.1 Simulation Results

The data is generated based on a random multivariate normal distribution where the mean is assigned as 0 and the variance-covariance matrix $\sum_X$ adapts a compound symmetry structure with the diagonal items set to 1 and the off-diagonal items being ρ values.

$$\sum_X = \begin{pmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{pmatrix}_{p \times p} \tag{3.6}$$

The class labels are generated using the Bernoulli trails with the following probability:

$$\pi_i(y_i = 1|x_i) = \frac{\exp(x_i\beta)}{1 + \exp(x_i\beta)} \tag{3.7}$$

The data matrix $x_i \sim N_p(0, \sum_x)$ is generated using the random multivariate normal distribution and the response variable y_i is generated by binomial distribution as shown in (3.6) and (4.2) respectively. For sufficient comparison of the performance of the model and subsidizing the effects of the data splits, all of the regularized regression models were built using the 10-fold cross validation procedure, the averages were taken over 100 partitioning times referred as 100 iterations in this paper. The data generated are high-dimensional in nature with the number of samples, n = 100 and total features, p = 1000. The true regression coefficients are set to 25 which are generated using uniform distribution with the min and max values 2 and 4, respectively.

With this set up of high-dimensional data, we simulated three different types of data each with correlation structures ρ = 0.2, 0.5 and 0.8 respectively. These values show the low, medium and high correlation structures in the data sets which are significantly similar to what we usually see in the gene expression or others among many types of data in the field of bioinformatics [Kim and Kim, 2019].

The prediction performance of any given model is largely dependent on the type of features. The features having an effect on the class will help in attaining the best prediction accuracies. In Figure 3.2 we see the RLFS method with the top-ranked features based on the k-SIS criterion includes the more true number of important features than other existing FS methods such as IG, Chi2, and MRMR used for comparison in this study. The proposed RLFS is performing consistently better across low, medium and highly correlated simulated data and the positive effect of having more number of true important variables is seen in all three simulation scenarios and further explained in detail.

Simulation Scenario (S1): low correlation 0.2

The predictors are generated having a low correlation structure with $\rho = 0.2$. The proposed classifier ERRM is performing better than existing classifier on all the FS methods: proposed RLFS, IG, Chi2 and MRMR. In addition, the proposed combination of RLFS method and ERRM classifier, with the accuracy and Gmean, each of which is 0.8606 and 0.8626 with the standard deviations (SD) of 0.049 and 0.073 respectively, is relatively better in comparison to other combinations of FS method and classifier such as RLFS-LASSO, RLFS-ALASSO, RLFS-ENET and the other remaining combinations as observed in Figure 3.3. The combination of the FS method IG with proposed ERRM with accuracy and SD of 0.8476 and 0.052 is also seen performing better than IG-LASSO, IG-ALASSO, IG-ENET, IG-SCAD, IG-MCP, IG-ABOOST, IG-RF, IG-LR, and IG-SVM. Similarly, the combination of Chi2-ERRM with an accuracy of 0.8538 and a SD of 0.053 is seen better than FS method Chi2 with the other remaining classifiers. The results are reported in Table 3.1. The combination of MRMR-ERRM has an accuracy of 0.8550 and Gmean of 0.8552 is better than the combination of FS method MRMR with the rest of the nine classifiers. The performance of proposed ERRM shows that ensembles approach is better than using individual classifiers.

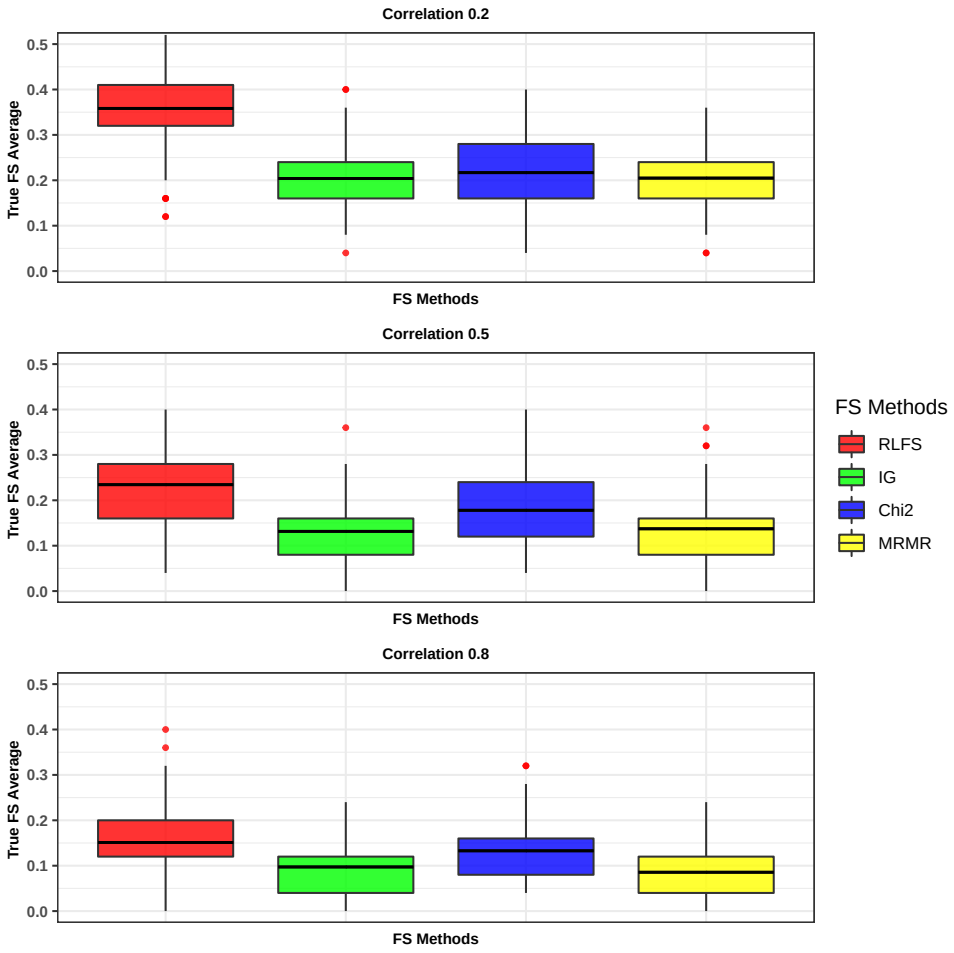

Figure 3.2: True number of features selected among top k-SIS ranked features and the average of this taken over 100 iterations for three different scenarios. RLFS- Resampling-based lasso feature selection; IG- Information Gain; Chi2- Chi-square; MRMR- Minimum redundancy maximum relevance;

All the classifiers with the features obtained from RLFS method achieved best accuracies in comparison to other FS methods. The combination of RLFS with SVM showed the second-best performance. It explains the characteristic of SVM working well with the proposed RLFS as it selected the best features in comparison to other FS methods. In general, the SVM classifier showed very competitive performance with all the FS methods.

The ENET method which is a combination of L1 and L2 penalty showed best perfor-

mance among all the regularized regression models with all the FS methods, and the best accuracy obtained with RLFS. This can be justified because of the known characteristic of ENET performing well in the high dimensional setting.

In summary, when we compare the average accuracies across all the combinations of FS method and classifiers, the proposed combination of RLFS-ERRM is having better performance than the other existing combinations of the FS and classifier without the proposed FS method RLFS and classifier ERRM itself. For example, the existing FS methods IG, Chi2, and MRMR with the eight existing individual classifiers performance are lower than the proposed RLFS-ERRM combination shown in the Table 3.1.

Simulation Scenario (S2): medium correlation 0.5

The predictor variables are generated using medium correlation structure with $\rho = 0.5$. The proposed combination of RLFS method and ERRM classifier, with the accuracy and Gmean, each of which is 0.9256 and 0.9266 with the standard deviations (SD) of 0.037 and 0.053. respectively, attained relatively better performance compared to other combinations of FS method and classifier such as RLFS-LASSO, RLFS-ALASSO, RLFS-ENET and the other remaining combinations. The results are shown in Table 3.2. From Figure 3.4, we see that the proposed ensemble classifier ERRM with other FS methods such as IG, Chi2, and MRMR is performing best compared to the other nine individual classifiers.

The SVM and ENET classifiers with the RLFS method attained accuracies of 0.9256 and 0.9244 respectively. These accuracies are almost similar to the combination of ERRM-RLFS but the combination of ERRM-RLFS, with the Gmean of 0.9266 and SD of 0.053, outperforms the SVM and ENET classifiers. The average SD of the proposed combination of the ERRM-RLFS is smaller than other combination of FS method and classifier. This shows the robustness of the proposed combination. The accuracies of SVM and ENET with the IG method are 0.9128 and 0.9150 respectively. These are relatively low compared to the accuracy of 0.9184 achieved by ERRM with IG. Similarly, the ERRM with the Chi2 method having an accuracy of 0.9160 showed relatively better performance than the

competitive classifiers ENET and SVM which acquired an accuracy of 0.9122 and 0.9098 respectively. Further, the ERRM classifier with the MRMR method having an accuracy of 0.9174 showed better performance than ENET, SVM, and other top-performing individual classifiers. While the SVM and ENET classifiers showed promising performance on the RLFS that had a good number of important features failed to show the same consistency on the other FS methods where there was more noise. On the other hand, the ensembles ERRM showed robust behavior, with being able to withstand the noise that helps in attaining better prediction accuracies and Gmean, not only with the RLFS method but also with other FS methods such as IG, Chi2, and MRMR.

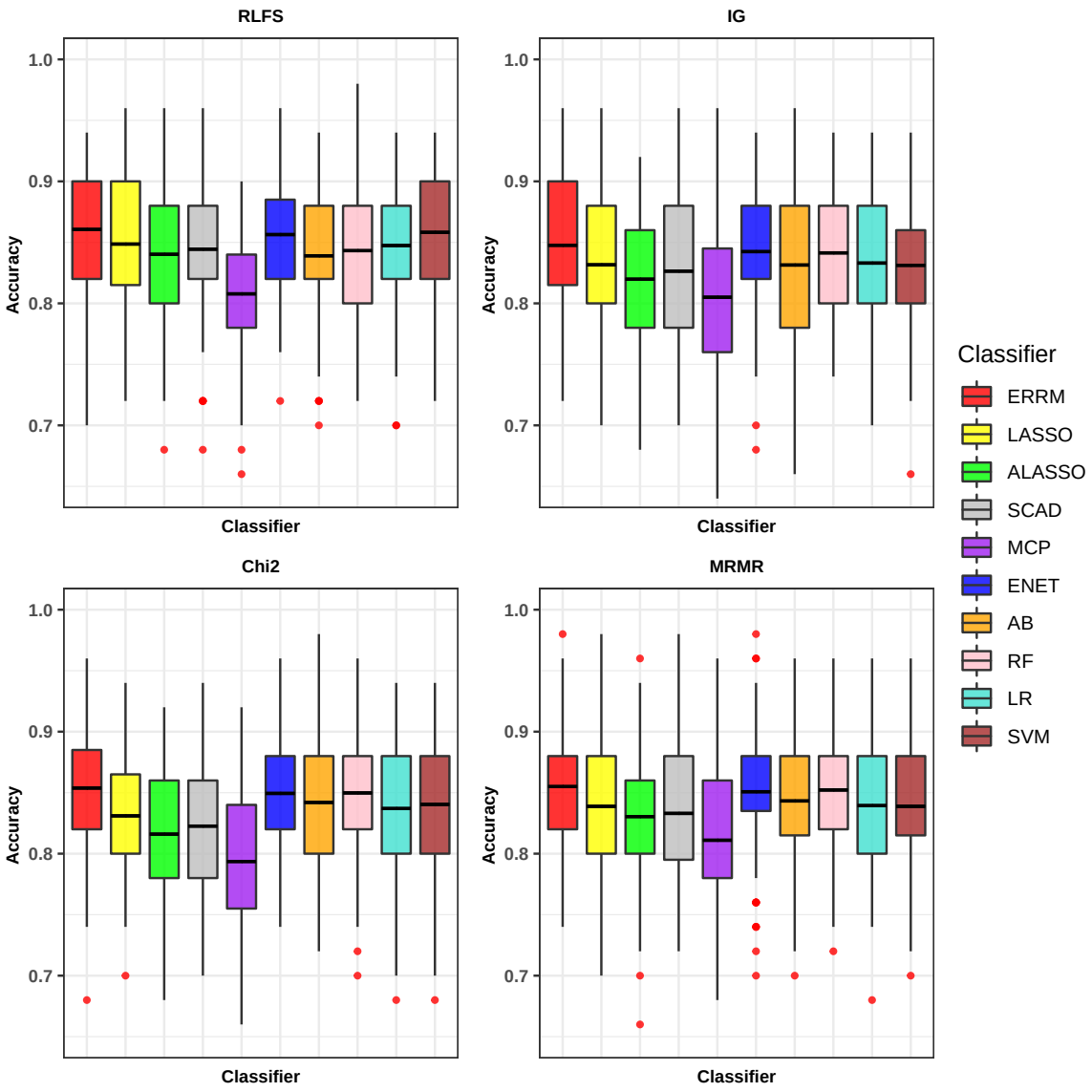

Figure 3.3: Boxplot showing the accuracies of Classifiers with FS methods in simulation scenario: S1. RLFS- Resampling-based lasso feature selection; IG- Information Gain; Chi2- Chi-square; MRMR- Minimum redundancy maximum relevance; ERRM- Ensembles of regularized regression models; LASSO: Least absolute selection shrinkage operator; ALASSO: Adaptive lasso; SCAD: Smooth clipped absolute deviation; MCP: Minimum concave penalty; ENET: Elastic Net; LR: Logistic regresion; RF: Random Forest; SVM: Support vector machine; AB: AdaBoost

Table 3.1: Average values taken over 100 iterations in simulation scenario: S1

FS	**Proposed RLFS**		**IG**		**Chi2**		**MRMR**	
+	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**
Classifier	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**
Proposed ERRM	**0.8606 (0.049)**	**0.8626 (0.073)**	0.8476 (0.052)	0.8483 (0.079)	0.8538 (0.053)	0.8551 (0.071)	0.8550 (0.049)	0.8552 (0.075)
LASSO	0.8486 (0.052)	0.8504 (0.075)	0.8316 (0.054)	0.8335 (0.083)	0.8310 (0.052)	0.8323 (0.071)	0.8388 (0.051)	0.8393 (0.077)
ALASSO	0.8402 (0.054)	0.8416 (0.077)	0.8198 (0.051)	0.8217 (0.079)	0.8160 (0.053)	0.8171 (0.075)	0.8304 (0.051)	0.8313 (0.079)
ENET	0.8564 (0.048)	0.8584 (0.072)	0.8424 (0.054)	0.8441 (0.081)	0.8494 (0.046)	0.8509 (0.067)	0.8508 (0.052)	0.8508 (0.077)
SCAD	0.8440 (0.054)	0.8457 (0.080)	0.8264 (0.057)	0.8283 (0.086)	0.8226 (0.061)	0.8239 (0.077)	0.8330 (0.056)	0.8336 (0.081)
MCP	0.8078 (0.049)	0.8095 (0.081)	0.8050 (0.062)	0.8074 (0.088)	0.7936 (0.060)	0.7952 (0.085)	0.8110 (0.060)	0.8126 (0.082)
ABOOST	0.8390 (0.051)	0.8224 (0.077)	0.8314 (0.060)	0.8328 (0.080)	0.8422 (0.054)	0.8435 (0.075)	0.8432 (0.054)	0.8437 (0.075)
RF	0.8432 (0.057)	0.8467 (0.084)	0.8414 (0.052)	0.8435 (0.078)	0.8498 (0.053)	0.8520 (0.075)	0.8522 (0.051)	0.8534 (0.077)
LR	0.8474 (0.050)	0.8489 (0.076)	0.8330 (0.053)	0.8346 (0.080)	0.8370 (0.054)	0.8380 (0.073)	0.8394 (0.051)	0..8394 (0.080)
SVM	0.8582 (0.049)	0.8595 (0.070)	0.8312 (0.052)	0.8320 (0.083)	0.8404 (0.054)	0.8416 (0.074)	0.8388 (0.049)	0.8378 (0.084)

Table 3.2: Average values taken over 100 iterations in simulation scenario: S2

FS	**Proposed RLFS**		**IG**		**Chi2**		**MRMR**	
+	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**
Classifier	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**
Proposed ERRM	**0.9256 (0.037)**	**0.9266 (0.053)**	0.9184 (0.039)	0.9195 (0.059)	0.9160 (0.038)	0.9165 (0.056)	0.9174 (0.042)	0.9176 (0.056)
LASSO	0.9146 (0.037)	0.9155 (0.053)	0.9034 (0.045)	0.9046 (0.061)	0.9020 (0.043)	0.9029 (0.063)	0.9066 (0.045)	0.9065 (0.062)
ALASSO	0.9056 (0.039)	0.9062 (0.056)	0.8956 (0.044)	0.8966 (0.065)	0.8948 (0.046)	0.8954 (0.065)	0.8984 (0.046)	0.8982 (0.062)
ENET	0.9244 (0.038)	0.9253 (0.052)	0.9150 (0.044)	0.9163 (0.061)	0.9122 (0.039)	0.9130 (0.060)	0.9158 (0.043)	0.9155 (0.058)
SCAD	0.9102 (0.041)	0.9110 (0.060)	0.8974 (0.046)	0.8986 (0.0630	0.8964 (0.045)	0.8972 (0.065)	0.9030 (0.045)	0.9030 (0.059)
MCP	0.8850 (0.047)	0.8855 (0.066)	0.8798 (0.050)	0.8813 (0.068)	0.8772 (0.045)	0.8782 (0.065)	0.8738 (0.049)	0.8738 (0.070)
ABOOST	0.9158 (0.035)	0.9166 (0.050)	0.9014 (0.046)	0.9027 (0.065)	0.9102 (0.040)	0.9112 (0.060)	0.9072 (0.047)	0.9075 (0.062)
RF	0.9148 (0.039)	0.9166 (0.055)	0.9186 (0.041)	0.9199 (0.059)	0.9154 (0.042)	0.9167 (0.060)	0.9116 (0.043)	0.9127 (0.060)
LR	0.9124 (0.037)	0.9127 (0.054)	0.9054 (0.043)	0.9063 (0.061)	0.9018 (0.045)	0.9024 (0.063)	0.9092 (0.043)	0.9084 (0.060)
SVM	0.9256 (0.038)	0.9261 (0.054)	0.9128 (0.038)	0.9135 (0.056)	0.9098 (0.043)	0.9099 (0.061)	0.9126 (0.045)	0.9120 (0.062)

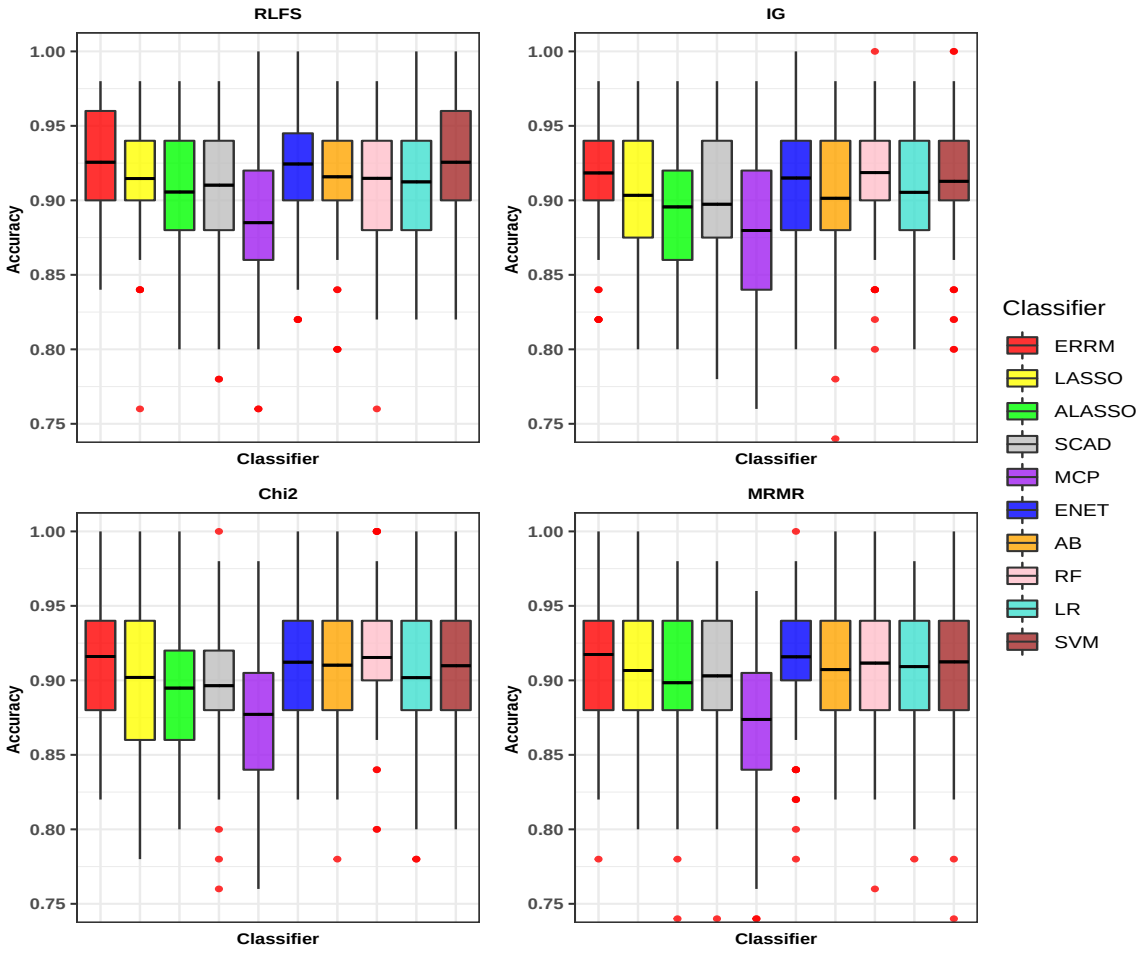

Figure 3.4: Boxplot showing the accuracies of Classifiers with FS methods in simulation scenario: S2

Simulation Scenario (S3): medium correlation 0.8

The data is generated based on high correlation data structure with $rho = 0.8$. The performance of proposed combination of RLFS-ERRM is relatively better than the other combinations of the FS methods and classifiers. The results for simulation scenario S3 are shown in Figure 3.5. The average accuracy and Gmean for all the FS and classifiers are noted in the Table 3.3. The SVM and ENET classifiers with all the FS methods showed a little better performance among all individual the classifiers. However, the accuracy and

Gmean attained by the proposed ensemble classifier ERRM with the FS methods RLFS, IG and Chi2 was relatively better compared to the individual classifiers with FS methods. The best performance is achieved by the proposed RLFS-ERRM combination with accuracy of 0.9586 and Gmean of 0.9596. The second best performing combination is MRMR-SVM. The lowest performance in terms of accuracy and the Gmean is shown by Chi2-MCP. The MCP classifier is having the lowest accuracies with all the FS methods. This explains that the MCP does not perform well when the predictor variables are highly correlated

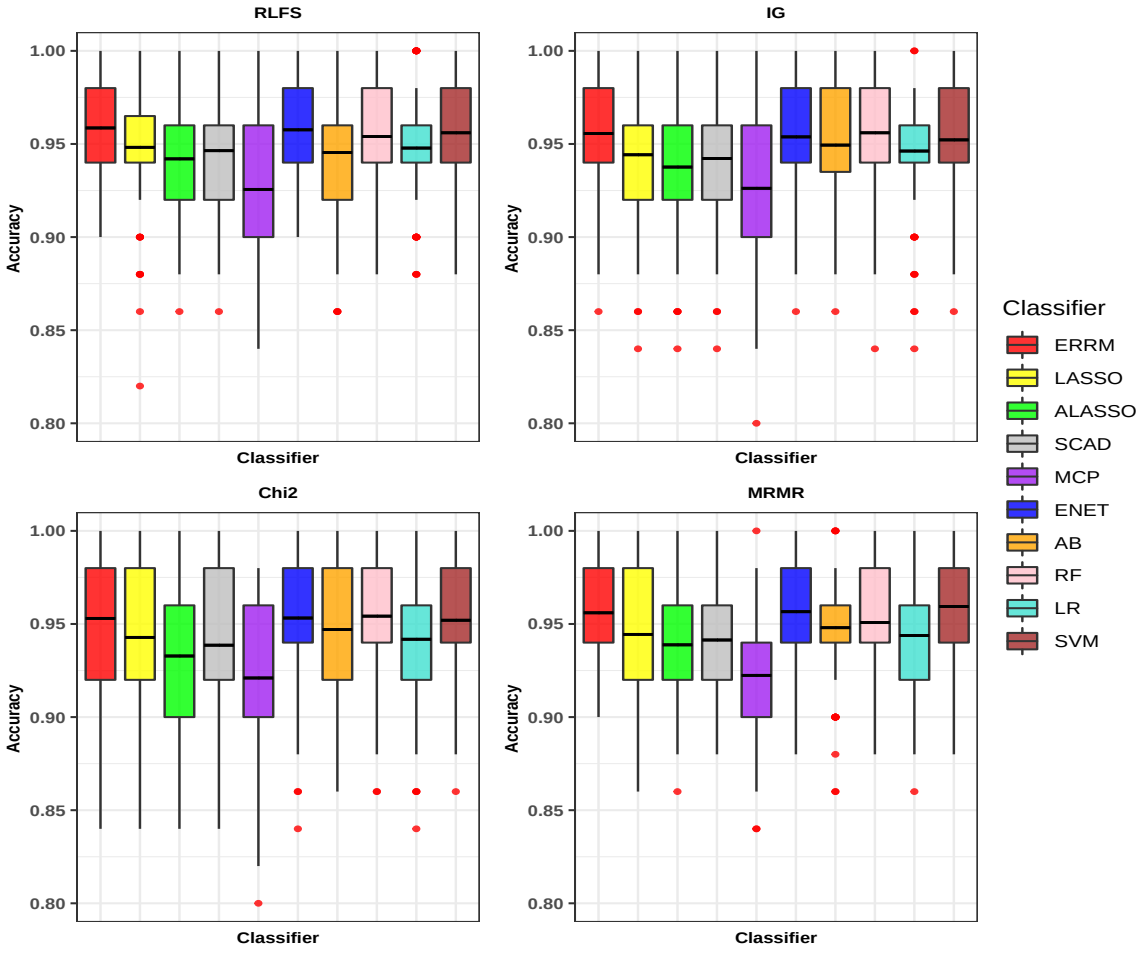

Figure 3.5: Boxplot showing the accuracies of Classifiers with FS methods in simulation scenario: S3

In summary, when we compare all the combinations of the FS methods and classifiers,

the proposed combination of the RLFS-ERRM is seen performing relatively better than any other existing combination of FS and classifiers.

Table 3.3: Average values taken over 100 iterations in simulation scenario: S3

FS	**Proposed RLFS**		**IG**		**Chi2**		**MRMR**	
+	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**
Classifier	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**
Proposed ERRM	**0.9586 (0.025)**	**0.9596 (0.039)**	0.9556 (0.027)	0.9565 (0.041)	0.9530 (0.034)	0.9544 (0.045)	0.9560 (0.024)	0.9558 (0.037)
LASSO	0.9482 (0.033)	0.9493 (0.050)	0.9442 (0.030)	0.9194 (0.045)	0.9428 (0.037)	0.9447 (0.051)	0.9444 (0.032)	0.9442 (0.042)
ALASSO	0.9420 (0.031)	0.9425 (0.051)	0.9376 (0.030)	0.9379 (0.047)	0.9328 (0.041)	0.9342 (0.056)	0.9388 (0.033)	0.9389 (0.047)
ENET	0.9576 (0.025)	0.9587 (0.039)	0.9538 (0.029)	0.9546 (0.042)	0.9532 (0.034)	0.9546 (0.045)	0.9566 (0.024)	0.9562 (0.036)
SCAD	0.9464 (0.031)	0.9475 (0.049)	0.9422 (0.030)	0.9428 (0.045)	0.9386 (0.043)	0.9401 (0.055)	0.9414 (0.031)	0.9408 (0.043)
MCP	0.9256 (0.040)	0.9270 (0.062)	0.9262 (0.038)	0.9269 (0.055)	0.9210 (0.041)	0.9221 (0.058)	0.9224 (0.034)	0.9223 (0.048)
ABOOST	0.9454 (0.032)	0.9469 (0.047)	0.9494 (0.030)	0.9501 (0.044)	0.9470 (0.034)	0.9482 (0.046_	0.9480 (0.029)	0.9481 (0.040)
RF	0.9540 (0.030)	0.9557 (0.043)	0.9560 (0.029)	0.9565 (0.043)	0.9542 (0.032)	0.9556 (0.044)	0.9508 (0.027)	0.9510 (0.039)
LR	0.9478 (0.029)	0.9482 (0.045)	0.9462 (0.030)	0.9469 (0.044)	0.9418 (0.038)	0.9432 (0.050)	0.9438 (0.028)	0.9437 (0.041)
SVM	0.9560 (0.027)	0.9568 (0.041)	0.9522 (0.030)	0.9527 (0.043)	0.9520 (0.031)	0.9526 (0.042)	0.9594 (0.026)	0.9587 (0.037)

3.4.2 Experimental Results

To test the performance of the proposed combination of ERRM with RLFS, and compare with the rest of the combinations of FS and classifiers, the gene expression data SMK-CAN-187 is analyzed. The data includes 187 samples and 19993 genes obtained from smokers, which included 90 samples with lung cancer and 97 samples without lung cancer. This data is high-dimensional in nature with the number of genes being 19993. The pre-processing procedures were performed to handle this high-dimensional data. As first filtering step to overcome the redundant noisy features, the marginal maximum likelihood estimation (MMLE) was performed, the genes are ranked based on their level of significance. These ranked significant genes are further applied on the FS methods as the final filtering step and final list of truly significant genes are obtained. These significant genes are applied to all the classification models. The average classification accuracy and Gmean of our proposed framework is tested. At first, the data was divided into training and testing set with 70% and 30% of samples respectively. 70% of the training data is given to the FS methods which rank the genes with respect to their importance and then the top-ranked genes are selected based on k-SIS condition. The selected genes are applied in all the classifiers. For standard comparison and mitigating the effects of the data splitting, all of the regularized regression models were built using the 10-fold cross validation, the models were assessed for testing the performance using different metrics and averages were taken over 100 splitting times referred as 100 iterations.

Figure 3.6 shows the box plot of average accuracies taken over 100 iterations for all the combinations of FS and classifiers. Each of the sub-figures in the figure shows the classifiers with the corresponding FS method. In simulation studies, we set the number of true important significant Chi2 when we generate the synthetic data and then after applying the FS methods, we estimate the average number of true important features. However, the significant genes cannot be known apriori. But, based on the simulation results in Section 3.4.1, where we saw that a higher number of significant features in the model leads to better prediction accuracy. From this, we can see that if the accuracy of

classifiers is better then there is a higher number of significant genes in the model. The average number of true important features showed in Figure 3.2 proved that the RLFS method selects more significant features. Likewise, the RLFS has more significant genes than the other FS methods, when we see in Table 3.4, the performance of all the individual classifiers when applied on the RLFS method, the accuracy and Gmean are relatively much better than the accuracies of the individual classifiers when applied on the IG, Chi2, and MRMR methods.

When we look at the performance of all the classifiers with the IG method in comparison to other FS method, there is much variation in the accuracies. The SVM classifier which attained the accuracy of 0.7026 with the RLFS method drops to 0.6422 with IG method, this shows that the IG is a very poorly performed FS method among all.

The proposed combination of the RLFS with ERRM classifier achieved the highest average accuracy of 0.7161 and Gmean of 0.7127. The RLFS method is also a top-performed FS method on all individual classifiers. However, among the other FS methods, the MRMR method when applied to all the individual classifiers showed relatively much better performance than the application of IG and Chi2 methods to the individual classifiers.

In summary, the proposed combination of the RLFS-ERRM outperformed the rest of the combination of the FS method and classifier. We can also note that the proposed combination is much better than other combinations of the existing FS and classifiers. For example, the proposed RLFS-ERRM is observed to have accuracy of 0.7161 whereas the other combinations such as Chi2-SVM is recorded with 0.6422. This explains that the proposed framework performs much better when the predictor variables are significantly correlated. The second best performed method is the RLFS-ENET combination. The lowest performance among all the combinations of FS method and classifier is shown by SVM classifier with the IG method.

Table 3.4: Average values taken over 100 iterations in SMK-CAN-187 data

FS	**Proposed RLFS**		**IG**		**Chi2**		**MRMR**	
+	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**	**Acc**	**Gmean**
Classifier	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**	**(SD)**
Proposed ERRM	**0.7161 (0.053)**	**0.7127 (0.082)**	0.6789 (0.056)	0.6791 (0.091)	0.6807 (0.056)	0.6808 (0.091)	0.7035 (0.056)	0.7024 (0.087)
LASSO	0.7073 (0.064)	0.7058 (0.087)	0.6726 (0.060)	0.6725 (0.095)	0.6680 (0.057)	0.6680 (0.090)	0.6859 (0.061)	0.6871 (0.097)
ALASSO	0.6878 (0.065)	0.6869 (0.091)	0.6715 (0.060)	0.6714 (0.094)	0.6696 (0.064)	0.6698 (0.092)	0.6800 (0.059)	0.6803 (0.092)
ENET	0.7138 (0.061)	0.7116 (0.085)	0.6733 (0.057)	0.6722 (0.093)	0.6733 (0.052)	0.6726 (0.090)	0.6998 (0.061)	0.6992 (0.095)
SCAD	0.7114 (0.054)	0.7098 (0.083)	0.6735 (0.056)	0.6732 (0.090)	0.6670 (0.058)	0.6669 (0.091)	0.6894 (0.059)	0.6901 (0.091)
MCP	0.6880 (0.010)	0.6870 (0.082)	0.6673 (0.057)	0.6663 (0.089)	0.6647 (0.059)	0.6639 (0.092)	0.6866 (0.057)	0.6874 (0.089)
ABOOST	0.6991 (0.064)	0.6958 (0.087)	0.6673 (0.054)	0.6634 (0.086)	0.6605 (0.058)	0.6583 (0.094)	0.6929 (0.050)	0.6897 (0.083)
RF	0.6975 (0.056)	0.6933 (0.089)	0.6729 (0.045)	0.6691 (0.078)	0.6738 (0.054)	0.6703 (0.090)	0.6942 (0.055)	0.6902 (0.088)
LR	0.7001 (0.065)	0.6987 (0.089)	0.6761 (0.058)	0.6662 (0.097)	0.6770 (0.059)	0.6769 (0.094)	0.7008 (0.058)	0.7000 (0.086)
SVM	0.7026 (0.058)	0.7014 (0.086)	0.6422 (0.059)	0.6430 (0.099)	0.6459 (0.066)	0.6477 (0.105)	0.6668 (0.058)	0.6658 (0.092)

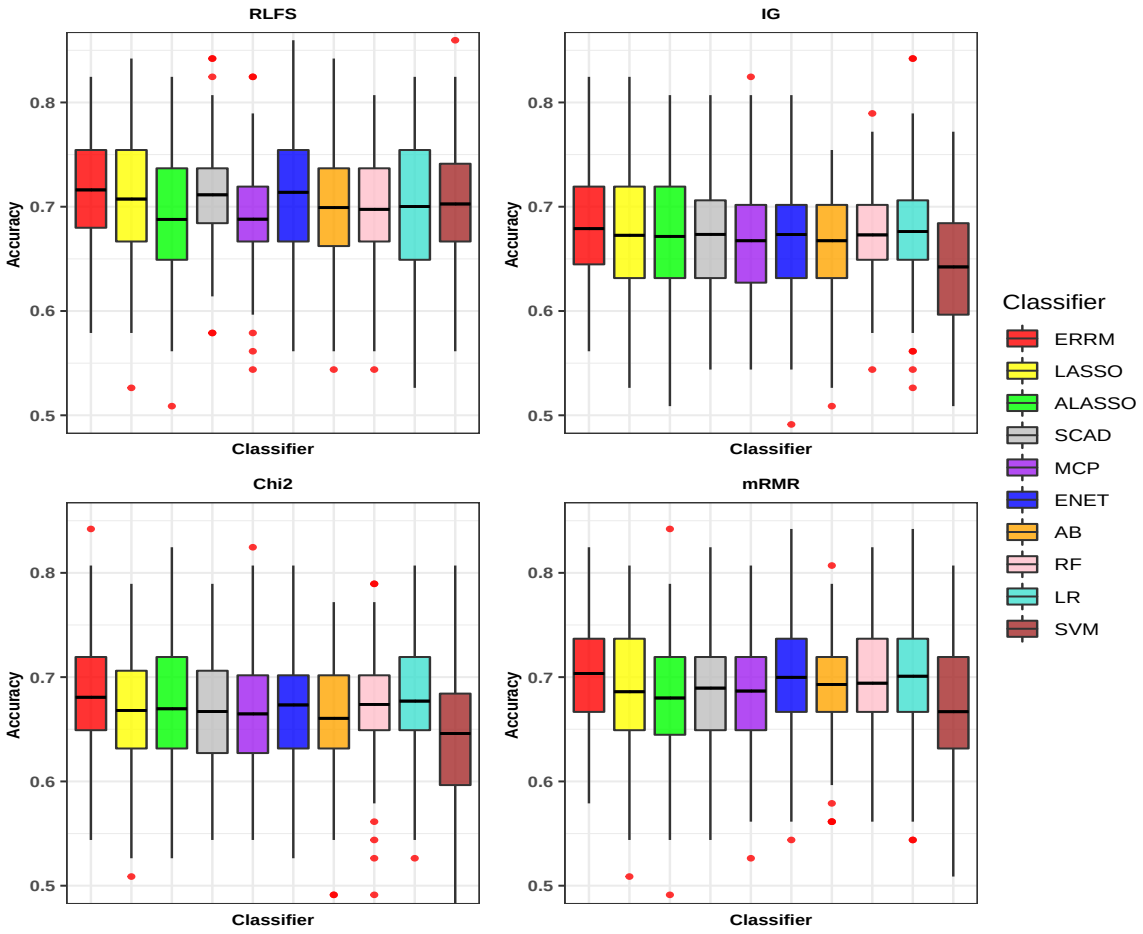

Figure 3.6: Boxplot showing the accuracies of Classifiers with FS methods in SMK-CAN-187 data

For assessing the importance of bootstrapping and FS screening of the proposed framework, we measured the performance of ERRM without FS screening. The results shows the ensembles method with and without bootstrapping procedure. In the former approach the bootstrapped samples were used and in the latter technique resampled samples were used. In Table 3.5 we see that the ERRM with bootstrapping procedure attains relatively better accuracy in comparison to other ERRM without bootstrapping.

Table 3.5: Comparison of proposed ERRM with and without bootstrapping

	Bootstrapping	Accuracy (SD)	Gmean (SD)
ERRM with FS screening	**Yes**	0.7129 (0.053)	0.7093 (0.091)
ERRM without FS screening	**No**	0.6947 (0.057)	0.6944 (0.089)

The performance of the regularized regression models used in the proposed ensembles algorithm is tested with the FS screening step and without FS screening step. In the former approach, the regularized regression models were built and tested using the proposed RLFS screening step with selected amount of significant features, whereas in the latter approach, the regularized models used all the features for building the model. The FS screening step is necessary for boosting the performance of the model in terms of accuracy and geometric-mean. The results are shown in Table 3.6.

Table 3.6: Comparison of regularized regression models used in the ERRM with and without FS screening

	FS screening	Accuracy (SD)	Gmean (SD)
LASSO	**Yes**	0.7073 (0.064)	0.7058 (0.087)
	No	0.6740 (0.061)	0.6752 (0.125)
ALASSO	**Yes**	0.6878 (0.065)	0.6869 (0.091)
	No	0.6740 (0.061)	0.6752 (0.125)
ENET	**Yes**	0.7138 (0.061)	0.7116 (0.085)
	No	0.6740 (0.061)	0.6752 (0.125)
SCAD	**Yes**	0.7114 (0.054)	0.7098 (0.083)
	No	0.6740 (0.061)	0.6752 (0.125)
MCP	**Yes**	0.6880 (0.010)	0.6870 (0.082)
	No	0.6740 (0.061)	0.6752 (0.125)

3.5 Discussion

In this research, we proposed the combination of RLFS and ERRM that are used for feature selection and classification respectively. The RLFS method ranks the features by employing the lasso method with resampling approach and the k-SIS criteria to set the threshold for selecting the optimal number of features and these features are applied on the ERRM classifier, which uses bootstrapping and rank aggregation, to select the best performing model across the bootstrapped samples and helps in attaining the best prediction accuracy in a high dimensional setting. The ensembles framework ERRM is built using five different regularized regression models. The regularized regression models are known for their best performance in terms of variable selection and prediction accuracy on the gene expression

data.

To show the performance of our framework we used three different simulation scenarios with low, medium and high correlation structures that matches the gene expression data. To further illustrate our point, we also used SMK-CAN-187 data. Figure 3.2 shows the boxplots of the average number of true important features, where the RLFS shows higher detection power than the other FS methods such as IG, Chi2, and MRMR. From the results, of both simulation studies and experimental data, we show that all the individual classifiers with RLFS method performed much better compared to the IG, Chi2, and MRMR. We also observed that all the individual classifiers showed a lot of instability with the other three FS methods. This means that the individual classifiers do not work well with more noise and less true important variables in the model. The SVM and ENET classifiers with all the FS methods performed a little better among all the classifiers. However, the performance was relatively still low in comparison to the proposed ERRM classifier with every FS method. The tree-based ensemble methods RF and ABOOST with RLFS also attained good accuracies but were not the best compared to the ERRM classifier.

The proposed ERRM method is assessed with the FS screening and without FS screening step along with the bootstrapping option. The ERRM with FS screening and bootstrapping approach works better than ERRM without the FS screening and bootstrapping technique. Also, the results from Table 3.5 shows that the ensembles with bootstrapping is better approach on both the filtered and unfiltered data. On comparing the performance of the individual regularized regression models used in the ensembles, the individual models with proposed RLFS screening step showed comparatively better accuracy in comparison to the individual regularized regression models without the FS screening. This means that using the reduced number of significant features with RLFS is better approach instead of using all the features from the data.

The ERRM showed better overall performance not only with the RLFS but also with the other FS methods compared in this study. This means that the ERRM is robust and works much better on the highly correlated gene expression data. The rule of thumb in

attaining the best prediction accuracy is that more of the true important variables, better the prediction accuracy. Thus, from the results of simulation and experimental data we see that the proposed combination of RLFS-ERRM has a superior compared to the other existing combinations of FS and classifiers as seen in the Tables 3.1, 3.2, 3.3 and 3.4. The proposed ERRM classifier showed the best performance across all the FS methods, with the highest performance achieved with RLFS method. However, the minor drawback in the framework is that the performance of ERRM would have been much better if the RLFS method had better selection average in retaining the significant features.

3.6 Conclusion

In this research, through extensive simulation studies, we showed the superior performance of RLFS with the k-SIS condition has a higher selection average in detecting the true important features than other competitive FS methods. The ensemble classifier ERRM also showed better average prediction accuracy with the RLFS, IG, Chi2, and MRMR compared to other classifiers with these FS methods. On comparing the ensembles, the proposed ERRM with bootstrapping showed better accuracy in comparison to the ERRM without bootstrapping. In both the simulation study and the experimental data SMK-CAN-187, the superior performance was achieved by the proposed combination of RLFS -ERRM compared to all other combinations of FS and classifiers.

Chapter 4

Adaptive lasso with weights based on normalized filtering scores in molecular big data

The content of this chapter has been published in [Patil et al., 2020c]

4.1 Introduction

DNA microarray technology is used to determine the expression levels of thousands of genes at the same time [Honrado et al., 2006] and used in applications of machine learning [Kim and Halabi, 2016]. The microarray gene expression data [Bourgon et al., 2010b, Lu et al., 2011b] is widely used for cancer classification and detection [Algamal and Lee, 2015]. The microarray datasets are high-dimensional with a very limited number of samples. Some of the problems associated with these datasets include redundancy, noise, and dimensionality [Kim and Kim, 2019]. To overcome these problems, many rank-based feature selection (RFS) techniques have been developed over the recent years. The RFS methods lessen the dimensionality of the data through ignoring irrelevant genes and provides a consistently improved performance when applied to the classification methods.

In RFS methods, the genes are ranked with regards to their importance that is determined through certain criteria [Kim and Kim, 2019, Patil and Kim, 2020], and the optimal number of top significant genes can be selected with the sure independence screening (SIS) [Fan and Lv, 2008] condition. Some of the popular RFS methods include IG [Quinlan, 1993], FS [UM, 2013], mmle [Algamal and Lee, 2019], and CS [Iguyon and Elisseeff, 2003]. The advantages of RFS methods is that they are computa-

tionally less intensive because they do not involve classification step during the training phase of ranking significant features [Ferreira and Figueiredo, 2012].

The non-parametric and parametric algorithms can be utilized for classification purposes in the microarray data. The non-parametric methods include random forest and support vector machines [Breiman, 2001, Hearst et al., 1998, Patil and Kim, 2020]. The parametric models comprise different variants of sparse regularized regression (SRR) models which are popular in the high throughput microarray data analysis [Hastie et al., 2009]. The SRR models are known for performing variable selection and classification at the same time. The popular SRR models include the lasso [Tibshirani, 1996], alasso [Zou, 2006], and elastic net [Zou and Hastie, 2005, Wang et al., 2019]. Most of these models suffer from prediction accuracy when the genes are having high correlation among them. Lasso has received great success but it suffers from bias and leads to inconsistency in selecting significant genes [Tibshirani, 1996]. The alasso has received much attention to overcome this issue. It consistently selects the significantly related variables and has the oracle properties [Fan and Li, 2001]. However, there is still a lack of performance by the alasso with standard ridge weights in highly correlated data.

Motivated by these problems, we aim to build an estimator which not just improves the prediction accuracy but it also gives the best set of truly important genes responsible for a disease. In this study, we propose alasso with weights from RFS methods. The RFS methods such as FS, IG, and CS are applied to the data to rank the genes based on their level of importance and a screening procedure, SIS, to select the top most important genes. Through this procedure many genes not pertinent to a disease are filtered out and henceforth the computational time required during the training phase by the classifiers is reduced. The ranked important variables from the RFS methods have corresponding values and each of these values are modified as proposed weights and then given to the alasso method to attain the best measure of various performances with important subset of genes. The proposed adaptive lasso variants will be compared with some of the existing adaptive lasso models. The different existing feature selection methods such as ridge estimator,

MMLE, FS, IG, and CS are discussed in Chapter 2. The traditional lasso and Adaptive lasso methods have also been discussed earlier in Chapter 2.

4.2 Data

4.2.1 Synthetic data

The synthetic data is produced through a multivariate random normal distribution using the auto-regressive correlation framework. The correlation matrix Σ is defined as follows:

$$\Sigma = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1g} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{2g} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{s1} & \rho_{s2} & \cdots & \rho_{sg} \end{pmatrix} \tag{4.1}$$

The binary response variables are generated using the following equation:

$$\pi(Y_o = 1|t) = \frac{\exp(t\beta)}{1 + \exp(t\beta)} \tag{4.2}$$

t is represented as true important variables, and the response variable Y_o is generated by Bernoulli trials as shown in Equation (4.2). Each of the generated synthetic data is having $s = 200$ and $g = 1000$ genes. The total number of true regression coefficients assigned as β were 6. The β was weak signal and is defined as follows: $\beta = (1.036480, -1.073300, -1.250950, 1.138730, -1.128370, 1.145260)$ [Pi and Halabi, 2018]. The pairwise correlation between i^{th} and j^{th} feature is defined as $\rho_{ij} = \rho^{|i-j|}$ where $i, j = 1, 2, ..., g$. We generated data with correlations set as $\rho = 0.2$, 0.5, and 0.8 respectively through this high-dimensional set up.

The microarray data is usually having high correlation and the synthetic data generated allows us to evaluate the proposed methods on such types of correlated data. Firstly, we partition the synthetic data into 70% of train data and 30% of testing data. The training data is applied to the RFS methods, the genes are ranked according to their significance

level, and the top-ranked significant genes are chosen with the SIS. The chosen genes are further given to the proposed alasso method and other methods used in this study. For measuring the model performance in detail and eliminate the effects of data splittings, the 10-fold cross-validation (CV) was enforced on all of the SRR models, and the computations were made by taking averages from 100 random partitioning times.

4.2.2 Real data

The data included 62 samples and 2000 genes, which includes 40 samples with colon cancer and 22 normal samples. The data is partitioned into 70% of data samples for training and 30% of data samples for testing purposes. The training data is used by RFS methods to ignore the unrelated genes and rank the important genes. The top-most important genes are selected based on SIS and are given to the various classifiers and the proposed alasso method with a 10-fold CV employed during model building process. Lastly the test data is used to estimate the various performance measures. The data was randomly split in every iteration and the mean values were taken over iterations.

The microarray data can be given by the following data matrix:

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1g} \\ x_{21} & x_{22} & \cdots & x_{2g} \\ \vdots & \vdots & \ddots & \vdots \\ x_{s1} & x_{s2} & \cdots & x_{sg} \end{pmatrix} \tag{4.3}$$

Each value represents the expression level of a particular gene $e = 1, ..., g$ of sample o is given by x_{oe}.

4.3 Proposed adaptive lasso with weights from various rank based feature selection techniques

In the first step, the training data X_k is applied to the RFS methods to lessen the dimension of the data by ignoring the irrelevant genes and obtain filtered significant gene set X_r. This step not only helps in improving the classification performance with the help of important genes but also reduces the computational time required in the next classification step. The RFS methods used are the FS, IG, and CS.

The equation for the proposed alasso method is given by:

$$\hat{\beta}_{proposed} = \underset{\beta}{\operatorname{argmin}} \left[-l(\beta, Y_k) + \lambda \sum_{e=1}^{f} w_e |\beta_e| \right], \tag{4.4}$$

where f is the filtered genes set after RFS step, λ is the tuning parameter, weight vector is given by w_e which is based on the proposed normalized weights of the RFS methods.

Calculating proposed weights

The RFS methods are used to assign scores to each of the feature based on their condition. Using these scores, the features are ranked through their importance. The top-ranked features are selected using the SIS condition and is defined as follows:

$$\left\lceil \frac{s}{log(s)} \right\rceil \tag{4.5}$$

where s is represents total count of samples. Selected features based on SIS are having the scores, the scores are normalized. Let ϕ_i be the scores for f number of features obtained after RFS step.

$$w_e = \frac{|\phi_i|}{|\frac{1}{s}\sum_{i=1}^{f} \phi_i|}, \tag{4.6}$$

where w_e is the normalized scores called as weights for all the f filtered significant genes, ϕ_i are the final scores from the RFS methods such as FS, IG, and CS.

Next, we divide the filtered training feature set X_r by the calculated proposed weights w_e to modify the filtered training feature set X_r

$$X_r^* = \frac{X_r}{w_e}, \quad e = 1, ..., f \tag{4.7}$$

Finally, use the above updated filtered feature set X_r for training purpose along with the weights w_e in the Equation (4.4)

Algorithm 2 The proposed alasso

Step 1: Obtain filtered training dataset X_r having true significant genes based on SIS approach using the RFS methods.

Step 2: Compute the proposed weights w_j for each of the RFS method using the Equation (4.6).

Step 3: Calculate new filtered training data set $X_r^* = X_r/w_e$.

Step 4: Obtain $\beta_{proposed}$ using the Equation (4.4).

Step 5: Apply test data X_q and evaluate various performance metrics.

4.4 Results

4.4.1 Synthetic data results

The RFS method with SIS for correlation data of 0.2, 0.5, and 0.8 are complied as boxplots described in Figure 5.1. As shown in boxplots, the true variable selection average of the RFS as FS method is better than the IG and CS methods, which have an overlapping performance.

Table 4.1: True variable selection average of filtering methods across synthetic data with different correlation scenarios. True positive rate (TPR) calculations are based on the mean and SD taken from 100 times of simulations.

Correlation	FS (SD)	IG (SD)	CS (SD)
0.2	5.29 (0.671)	4.44 (0.924)	4.45 (0.946)
0.5	5.26 (0.733)	4.41 (0.933)	4.43 (0.912)
0.8	5.34 (0.669)	4.49 (1.039)	4.50 (1.039)

SD: Standard Deviation; FS: Fisher score; IG: Information gain; CS: Chi-square

The FS method selected an average of 5.29, 5.26, and 5.34 true variables among the six true important variables across the three different correlation structures, as recorded in Table 4.1. Besides, when we observe the SD of FS, IG, and CS, we see that the FS method is more consistent with less variation in true variable selection average across 100 iterations.

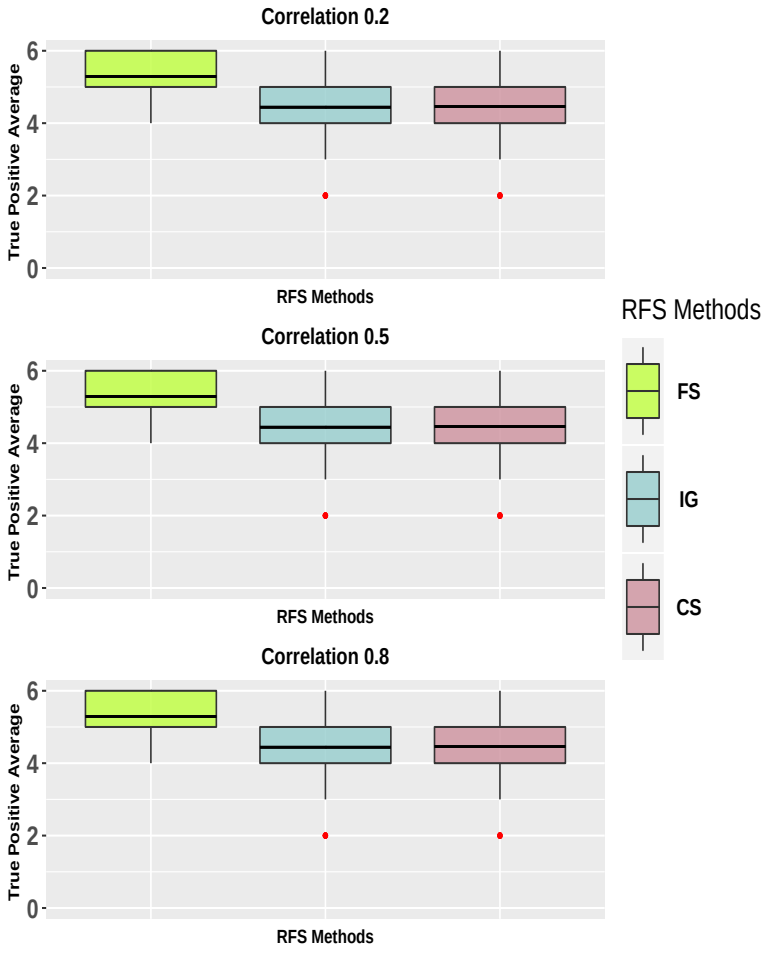

Figure 4.1: Boxplot showing the true positive rate (TPR) of the variable selection average of filtering methods in correlated data 0.2, 0.5, and 0.8. Calculations are based on the averages of true positives taken from 100 times of simulations. RFS: Rank-based feature selection methods; FS: Fisher score; IG: Information gain; CS: Chi-square

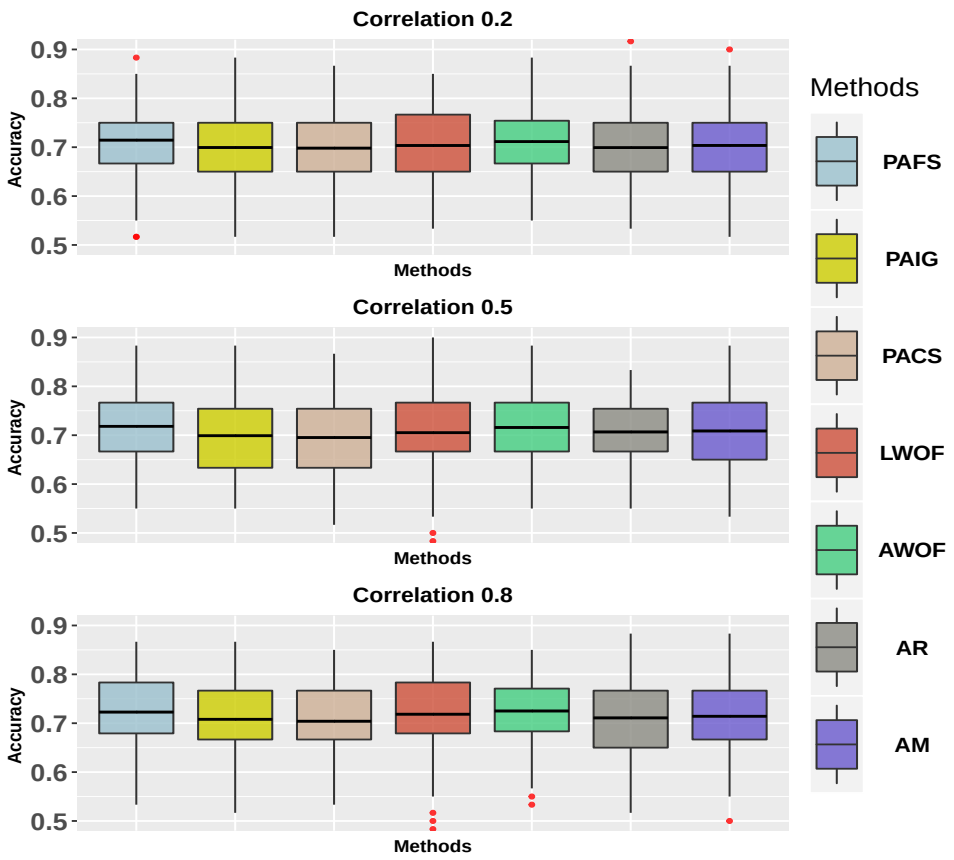

Figure 4.2: Average accuracy of the classification methods with RFS methods on the simulated data set with different correlations. PAFS (proposed alasso with fisher score), PAIG (proposed alasso with information gain), PACS (proposed alasso with chi-square), LWOF (lasso without filtering), AWOF (alasso without filtering), AR (alasso with ridge weights), AM (alasso with mmle weights).

Figure 4.2 shows the boxplots of the classification accuracies for the classifiers with filtering based on SIS and without filtering for three different correlation structures of 0.2, 0.5, and 0.8. We see that the accuracies of PAFS (proposed alasso with Fisher score) with SIS is better compared to others. The proposed alasso with weights from various filtering methods such as FS, IG, and CS, the alasso with weights from filtering methods such as ridge and mmle, lasso and alasso without filtering were run for 100 times with the

autoregressive correlation structure with 0.2, 0.5, and 0.8. The performances of classification accuracy, AUROC, GM-mean, and TP for these variable selection methods with and without filtering are reported in Table 4.2.

The proposed alasso with the FS as a filtering step with an accuracy of 0.7145 and 0.7183 for correlation of 0.2 and 0.5, respectively, is having better prediction accuracy compared to the classifiers with and without filtering approaches. From Table 4.1, we saw that the FS method had the highest count of true positives among the six important variables across all correlation structures. Similar results were seen on proposed alasso with FS, where the TP was higher than proposed alasso with IG or CS.

Table 4.2: Performance of filtering and classification methods in synthetic data with three types of correlation structures. The calculations are based on the mean and SD of the performances taken from 100 times of iterations in each of the correlated data.

Corr	Methods	Accuracy (SD)	AUROC (SD)	GM-mean (SD)	TP (SD)
0.2	PAFS	0.7145 (0.073)	0.7160 (0.071)	0.7126 (0.073)	5.20 (0.752)
	PAIG	0.6993 (0.072)	0.7013 (0.071)	0.6964 (0.074)	4.40 (0.953)
	PACS	0.6981 (0.072)	0.6999 (0.071)	0.6960 (0.072)	4.45 (0.946)
	AR	0.6990 (0.074)	0.7011 (0.072)	0.6975 (0.074)	5.29 (0.607)
	AM	0.7036 (0.068)	0.7052 (0.067)	0.7016 (0.070)	5.25 (0.672)
	LWOF	0.7035 (0.082)	0.7079 (0.076)	0.6955 (0.105)	4.70 (1.251)
	AWOF	0.7113 (0.071)	0.7127 (0.070)	0.7089 (0.073)	5.40 (0.752)
0.5	PAFS	0.7183 (0.073)	0.7180 (0.075)	0.7141 (0.077	5.11 (0.827)
	PAIG	0.6968 (0.078)	0.6987 (0.078)	0.6938 (0.081)	4.36 (0.948)
	PACS	0.6951 (0.084)	0.6951 (0.084)	0.6896 (0.086)	4.43 (0.912)
	AR	0.7066 (0.065)	0.7056 (0.067)	0.7012 (0.070)	5.26 (0.705)
	AM	0.7086 (0.076)	0.7077 (0.076)	0.7025 (0.079)	5.28 (0.725)
	LWOF	0.7050 (0.094)	0.7076 (0.087)	0.6768 (0.165)	4.68 (1.420)
	AWOF	0.7158 (0.069)	0.7148 (0.070)	0.7097 (0.073)	5.41 (0.779)
0.8	PAFS	0.7228 (0.073)	0.7222 (0.074)	0.7190 (0.075)	5.15 (0.770)
	PAIG	0.7081 (0.072)	0.7082 (0.073)	0.7051 (0.074)	4.41 (1.064)
	PACS	0.7041 (0.072)	0.7043 (0.073)	0.7011 (0.074)	4.47 (1.058)
	AR	0.7108 (0.074)	0.7109 (0.074)	0.7079 (0.075)	5.26 (0.733)
	AM	0.7143 (0.074)	0.7146 (0.074)	0.7109 (0.076)	5.25 (0.729)
	LWOF	0.7185 (0.075)	0.7192 (0.075)	0.6984 (0.131)	4.78 (1.259)
	AWOF	0.7250 (0.068)	0.7249 (0.068)	0.7214 (0.069)	5.36 (0.785)

4.4.2 Application to real data

The real microarray data of colon cancer is used in this study. This dataset had 62 samples and 2000 genes. To test the estimates of the proposed alasso method with various RFS methods, alasso with mmle and ridge weights, Lasso and alasso, without filtering, the colon cancer microarray data is analyzed. Figure 4.3 shows boxplot and density plot of the pairwise correlation within the genes in colon cancer data. The average correlation between genes is 0.4275, which is a high correlation and lies between the values evaluated in the synthetic datasets.

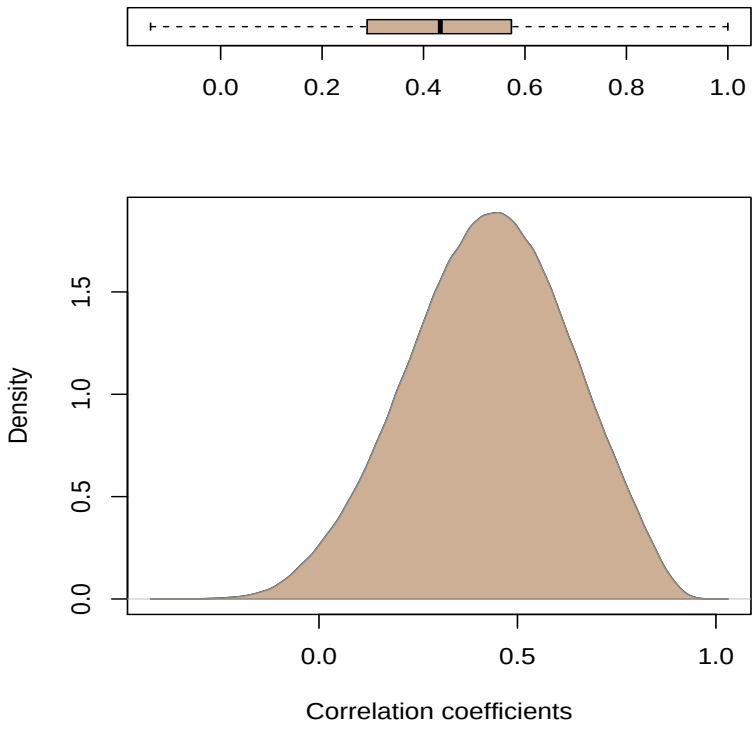

Figure 4.3: Pairwise correlation coefficients between 2000 genes in colon data. The mean pairwise correlation value is 0.4275 as shown in the boxplot.

Table 4.3: Evaluation of classifiers with RFS methods in colon cancer. All values are evaluated based on the average and SD of various performances measured from 100 iterations.

Methods	Accuracy (SD)	AUROC (SD)	GM-mean (SD)
PAFS	0.7936 (0.092)	0.7504 (0.115)	0.7094 (0.164)
PAIG	0.7852 (0.087)	0.7307 (0.114)	0.6760 (0.187)
PACS	0.7715 (0.101)	0.7175 (0.119)	0.6417 (0.223)
AR	0.7573 (0.095)	0.6976 (0.113)	0.6166 (0.221)
AM	0.7042 (0.100)	0.6599 (0.122)	0.5727 (0.242)
LWOF	0.7531 (0.093)	0.6815 (0.114)	0.5796 (0.234)
AWOF	0.7115 (0.098)	0.6600 (0.120)	0.5903 (0.209)

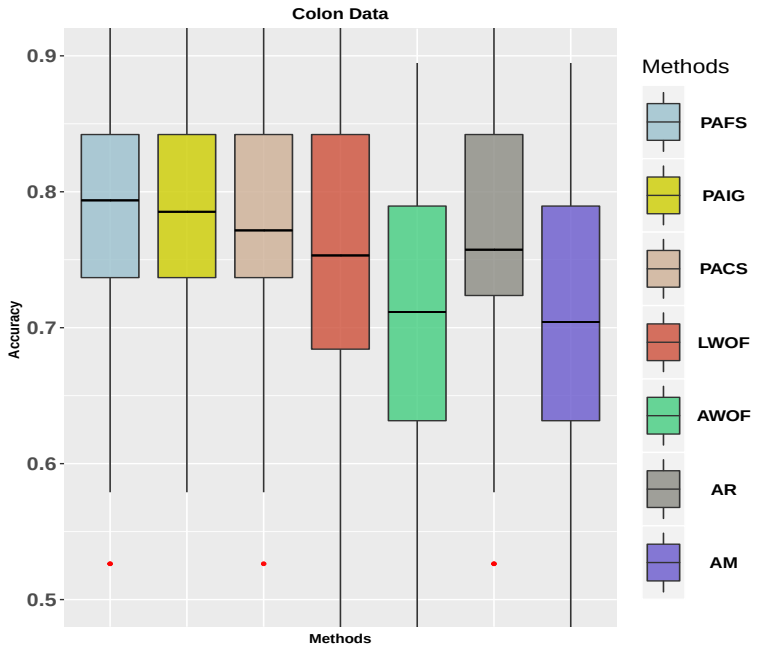

Figure 4.4: Average accuracy of the various classifiers with RFS methods on the Colon data.

Table 4.3 shows the averages of accuracy, AUROC, and GM-mean calculated for all the methods with and without filtering. The proposed alasso with the weights from FS, IG, and CS as filtering methods had better average accuracy compared to other combinations of methods measured. The proposed alasso with weights from FS based on SIS attained an accuracy of 0.7936 while the alasso without filtering and alasso with mmle weights attained accuracies of 0.7115 and 0.7042, respectively. This variation in the accuracies explains the importance of the proposed alasso algorithm with the filtering step in achieving better prediction accuracies in gene expression analysis. The variation of accuracies across 100 data splittings is shown as boxplots in Figure 4.4. There is a clear difference between the performance of the proposed alasso method with filtering methods such as FS, IG, and CS and other combinations of methods compared.

The final list of top 10 genes selected by each of the proposed alasso with RFS methods such as FS, IG, and CS, based on SIS condition, run over 100 times of data splittings are noted and displayed as Venn diagram in Figure 4.5. We see that there are six common genes across all three methods. M63391, M76378(3), T71025, X14958, Z50753, and M76378(1) are the common genes found across the three methods, as reported in Table 4.4.

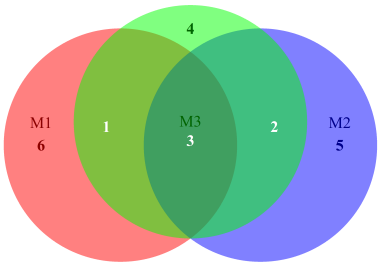

Figure 4.5: A schematic illustration of genes from various filtering methods across proposed variable selection method. M1: proposed alasso with FS, M2: proposed alasso with IG, and M3: proposed alasso with CS.

The genes M63391, M76378(3), and T71025 are common between proposed alasso with FS, proposed alasso with IG, and proposed alasso with CS. The M76378 was ranked among the top 3 genes in all three methods. This shows that it is a very significant gene related to colon cancer. The genes M63391, M76378(3), H08393, and H55916 selected by proposed alasso with FS were also recorded as genes significantly pertinent to colon cancer [Kim and Kim, 2019, Wang et al., 2019, Mcdermott et al., 2016].

Table 4.4: The top 10 significant genes selected across three methods of Proposed alasso with FS, IG, and CS on colon cancer microarray data.

Rank	Proposed alasso with FS	Proposed alasso with IG	Proposed alasso with CS
1	M63391***	J02854	M76378(1)**
2	M76378(3)***	M76378(1)	M63391
3	M26697	M76378(3)	M76378(3)
4	J05032	H64489	T71025
5	T71025***	Z50753**	H05803
6	X14958**	M63391	Z50753
7	X63629	T71025	X14958
8	R54097	M82919	R41873
9	H08393	L03840	H23544
10	H55916	T57468	H40137

4.5 Discussion

We proposed alasso with weights based on RFS methods. The RFS methods FS, IG, and CS, rank the genes, and each of the methods selects top SIS genes. The RFS methods assign scores to each of the chosen genes. These scores were normalized as proposed new weights and applied to the alasso to give the best performance of classification and gene

selection. The RFS methods with SIS helps in removing the unrelated genes which lead to a reduction in the computational time when applied to the proposed alasso. We compared the metrics of the proposed RFS with alasso framework with popular filtering approaches of ridge and mmle whose weights were applied to the alasso, and without filtering where the lasso and alasso were trained on the 70% of the dataset with all the genes. The performance of all these combinations of methods was assessed using accuracy, AUROC, and GM-mean. The proposed alasso with the filtering procedures showed the best performance in all of these metrics as shown in Table 4.3.

To illustrate the performance of the proposed alasso with RFS methods, we applied synthetic data with wide range of correlation structures and real microarray data. Firstly, we showed the superior performance of the RFS as FS method in selecting the true important variables in Table 4.1. This shows that the FS is the best RFS method in selecting more number of true important variables in highly correlated data. Also, the proposed alasso method with weights based on the FS was the best-performed combination on both synthetic and real colon gene expression datasets. The minor disadvantage in the proposed framework is that the best performing RFS method as FS retrieved a significantly higher number of true important features. However, it did not include all the important features. The performance of the proposed framework would have been still better if all the important features were added.

Along with the classification accuracy, the role of gene selection by the proposed alasso with various RFS methods was assessed. There were a total of six common genes found across three proposed Alssso with RFS methods combinations. Most of the genes selected by these combinations were significantly related to colon cancer [Kim and Kim, 2019, Wang et al., 2019, Mcdermott et al., 2016]. The three common genes found in all the three FS-proposed alasso combined methods were found within top-five ranked important genes of the proposed alasso with an FS. All three genes M63391, M76378, and T71025, were significantly related to colon cancer [Yap et al., 2004, Wang et al., 2019, Kim and Kim, 2019]. This also shows that the FS as RFS method, when applied with proposed alasso, is more

effective in selecting such significant genes.

4.6 Conclusion

In this research, the alasso method with weights from various RFS methods was proposed. Firstly, the RFS methods such as FS, IG, and CS were employed to remove the redundant features, and the scores assigned to each feature by these methods were normalized as proposed weights and applied to the alasso method. The proposed framework of alasso with RFS methods such as FS, IG, and CS was compared with other existing methods with filtering and without filtering on synthetic and real gene expression microarray data. The proposed framework displayed superior performance of accuracy, AUROC, GM-mean, and gene selection. The genes selected by the proposed alasso method on the colon data were proven to be the significant genes relating to the cancer. With more relevant genes in the model, attaining the highest classification performance in terms of various performance metrics was expected. As a future work, we would focus on developing a methodology of better feature selection in highly correlated biological data.

Chapter 5

Improving the classification performance with group lasso based ranking method in high dimensional correlated data

The content of this chapter has been published in [Patil et al., 2020b]

5.1 Introduction

The alteration of healthy cells to tumor ones results from the interactions of genetic agents with extraneous factors such as chemicals and viruses [Celli et al., 2018]. The process of transformation is complex, and the part of DNA Methylation (DNAmeth) in process of carcinogenesis is widely seen [Celli et al., 2018]. It is an essential epigenetic mechanism regulating the direct alteration of DNA, the aberrant DNA is responsible for the development of many illness [Patil et al., 2020a, Weber et al., 2007, Jones, 2012]. DNAmeth is further associated with cell differentiation, genetic reprogramming, and gene expression [Mikeska and Dobrovic, 2016, Holliday and Pugh, 1975, Choy et al., 2010]. DNAmeth includes a process where the enzyme DNA methyltransferase is used as an agent in catalyzing the conversion of cytosine to 5-methylcytosine that might alter the activity near DNA [Patil et al., 2020a, Celli et al., 2018, Liu et al., 2017]. The process is carried out by adding the methyl group to the fifth carbon in the DNA molecule [Patil et al., 2020a]. The transcription process of genes can be changed by the presence of methyl group [Stolzenburg et al., 2016] and may lead to the progression of a tumor [Liu et al., 2017, Kobayashi et al., 2011]. Different types of diseases such as cancer and cardiovascular are linked with DNAmeth [Kim et al., 2010, Urdinguio et al., 2009]. Hence, the studies on

DNAmeth may aid in determining biomarker identification and various disease classification [Patil et al., 2020a].

Large amounts of DNAmeth data are developed with IIHM technology and are deposited and made publicly available through online repository of the gene expression omnibus (GEO) [Teschendorff et al., 2010]. The data sets generated by IIHM 450K are ultra-high dimensional with over 450K CpGs. The high-dimensional IIHM 27K data sets are having more than 27000 CpGs called CpG sites [Toyota et al., 1999, Patil et al., 2020a]. The IIHM 27K DNAmeth data is utilized for validating the performance of the models in this paper. The methylation status [Bibikova et al., 2006] shows the DNA sequence that are methylated and not methylated and is estimated with β-values. For every CpG site, the β-value is computed based on ratio of fluorescent signals measured as levels of the alleles that are methylated denoted by Y and not methylated denoted by N [Sun and Wang, 2012]. The β-value lies within 0 and 1 which represent the no methylation and methylation status [Sun and Wang, 2012, Patil et al., 2020a].

$$\beta = \frac{\max(Y, 0)}{\max(N, 0) + \max(Y, 0) + 100} \tag{5.1}$$

Many studies have been conducted pertaining to the FS and classifiers in gene expression data [Patil and Kim, 2020]. However, comparatively fewer statistical analysis has been conducted for DNAmeth data produced from the IIHM 27K technology. Few of the present studies of DNAmeth data analysis includes from supervised [Patil et al., 2020a, Raweh et al., 2017, Sun and Wang, 2012, Milani et al., 2010] and unsupervised [Mitra et al., 2002] learning problems. The main difference between the DNAmeth data and microarray data is that the former is having methylated values that lie between 0 and 1 with the group correlation structure within genes, and the latter has continuous values based on the expression levels of each gene [Patil et al., 2020b].

Many different types of high-throughput data are available in bioinformatics, such as DNAmeth, microarray RNA-seq analysis [Kim and Kim, 2019, Bourgon et al., 2010a]. Dealing with this data type is very difficult problem and it becomes a necessity to re-

move the unimportant features and lower the dimension of the data by employing various FS methods [Patil and Kim, 2020, Patil et al., 2020a]. The FS method is applied to rank the CpGs depending upon their importance, the top-ranked significant CpGs can be selected, ignoring the noisy and unrelated features. There are several popular methods used for FS in the literature [Hira and Gillies, 2015, Jović et al., 2015], including fisher score (FIS) [UM, 2013], information gain (ING) [Quinlan, 1993], minimum redundancy maximum relevance (MRMR) [Peng et al., 2005], random forest variable importance (RFVI) [Menze et al., 2009, Guyon et al., 2002]. The FS methods are computationally faster and help to improve classification performance [Patil and Kim, 2020]. These well known FS methods also do not work accurately on highly correlated grouped data. This effects in selecting the considerably important features causing a disease. In the highly correlated grouped DNAmeth data, the FS methods rank the significant CpG sites related to the disease in higher-order and ignore the unrelated CpG sites. The important CpG sites help to boost classification performance. For selecting the most significant CpG sites related to the disease considering the high correlation among the groups of CpG sites in a gene, it is necessary to develop a novel filtering method applied to classifiers and compare the proposed RGLR method with the other FS methods with the different classifiers through simulated and experimental DNAmeth data.

In this chapter, we propose a filtering method called RGLR to obtain filtered DNAmeth data. The filtered DNAmeth data is applied to various classifiers. The resampling based method was better FS approach in high-throughput data [Kim and Kim, 2019, Patil and Kim, 2020]. The proposed RGLR method ranks the significantly important CpG sites, and these CpG sites are applied to the different classifiers to attain the best accuracies [Patil et al., 2020b]. Several popular classifiers such as random forests, support vector machines, and Naive Bayes were considered for comparison wih proposed models. These classifiers along with some of the widely used FS methods such as FIS, ING, MRMR, and RFVI have been discussed in Chapter 2.

5.2 Data

5.2.1 Simulation data

The simulation dataset is produced from a multivariate normal distribution using the auto-regressive structure. In order to generate simulation data that is similar to experimental DNAmeth data, we need to have correlated group variables that lie within 0 and 1. In the simulation setting, we have 600 genes that have 1-9 CpG sites in a manner that there are 100 genes in the first group with one CpG site each follwed by 150 genes in the second group, the third to ninth have 50 genes, respectively. Therefore, among the 9 groups, we have 2500 CpG sites in total [Patil et al., 2020a, Sun and Wang, 2012]. The DNAmeth β values of the j^{th} gene for i^{th} individual is evaluated as:

$$X_{i,j} = \frac{exp(m_{i,j})}{1 + exp(m_{i,j})} \tag{5.2}$$

where, $m_{i,j} \sim \sqrt{s}N_{f_j}(\mu, \sum)$ and f_j is the size of the j^{th} gene. $s = 4$ is a scale parameter and $\mu = (-0.1, -0.1, -0.1, -0.1, ... - 0.1, -0.1, -0.1)$, the methylated values have 0 for no methylation and 1 for methylated. We chose a total of nine genes from nine different gene groups [Sun and Wang, 2012]. The corresponding regression coefficients [Sun and Wang, 2012, Li and Li, 2010] were denoted as follows:

$$\theta_{l,j} = (-1)^{j+1}\frac{\delta}{\sqrt{l}}\ , \quad \text{for all } \ l = 1, ..., \lceil \frac{f_j}{2} \rceil; \ \ j = 1, ..., 9 \ \ and \ \ \delta = 1. \tag{5.3}$$

Half of the CpGs in the CpG feature set are assigned as true important features related to the disease [Sun and Wang, 2012]. We selected a total of 25 CpG sites as true important features among a total of 2500 CpG sites. The binary class variables are generated using,

$$y_i \sim Bernoulli(p(t_i)), \qquad p(tt_i) = \frac{\exp(t_i^T\theta)}{1 + \exp(t_i^T\theta)} \tag{5.4}$$

where $t_i = (t_{i,1}^T, t_{i,2}^T, ...t_{i,599}^T, t_{i,600}^T)^T$ is the true CpG sites, and $\theta = (\theta_1^T, \theta_2^T .....\theta_{599}^T, \theta_{600}^T)^T$.

The DNAmeth data is popular for their strong correlation structures. The correlation within genes is denoted through the covariance matrix $\sum_{uv} = \rho^{|u-v|}$. The autoregressive

AR(1) is used to generate the correlated variables. Three datasets comprising of low to high correlation structures set as $\rho = 0.1$, 0.4, and 0.7, respectively were generated through this synthetic data setting. In each of the synthetic data, equal number of cases and controls were obtained contributing to a total of 400 samples. Each of these samples had 2500 CpGs called as features.

Firstly, we divide the data into a training data and test data having 70% and 30% of samples, respectively. The training data is applied to the proposed RGLR method along with the other FS methods; the reduced significant training data is obtained with significant CpG sites and then is applied to various classifiers. Finally, the test data is applied to estimate the models performance. To obtain accurate estimates of the proposed RGLR method and existing FS methods, each of these procedures were carried out for 100 iterations.

5.2.2 Experimental data

To measure the performance of the FS and the classifiers, we obtained the DNAmeth data from the GEO database with accession GSE26126 [Kobayashi et al., 2011]. The dataset had 95 prostate tumor samples and 98 healthy prostate samples, making a total of 193 samples generated from IIHM 27K technology. This high-dimensional DNAmeth dataset with 27578 CpG sites needs to undergo filtering steps to remove the noisy and uninformative CpG sites. The given DNAmeth data is divided into 70% train and 30% test set respectively. As an initial pre-processing stage, to remove the uninformative CpG sites, we applied the marginal maximum likelihood estimation [Kim and Kim, 2019] to the training data, the CpG sites were ranked based on the mmle criteria. We selected the top 2500 highly correlated CpG sites based on the ranks assigned by the marginal maximum likelihood estimator. These top-ranked significant CpG sites are applied to the proposed RGLR method and the other popular FS methods used in this study. These filtering methods further evaluate the data and ignore the redundant CpG sites and rank the important CpG sites. The ranked CpG sites are applied to the various classifiers to fit the models. Finally,

the test data is applied to the fitted models, and the performances are estimated using the various metrics. We conducted this procedure for 100 times with random data splittings at each time.

5.3 Proposed resampling of group lasso based ranking method

The resampling-based filtering methods are proven to be a better approach than using the present feature selection methods in microarray data [Kim and Kim, 2019, Patil and Kim, 2020]. The DNAmeth data is having group correlation among the CpG sites and they are highly correlated. The proposed RGLR method ranks the CpG sites based on the selection frequency.

The group lasso method is derived from the groupwise-majorization-descent [Yang and Zou, 2015] (GMD) algorithm by satisfying the quadratic majorization (QM) condition [Yang and Zou, 2015, Yang and Zou, 2017]. Let $\Phi(Y_t, \beta^T X_t)$ be denoted as loss function that is used by model while training. In this paper, we use DNAmeth data with binary class labels $Y_t = (0, 1)$ coded as $Y_t = (-1, 1)$. The group-lasso penalized empirical loss [Yang and Zou, 2015] formulation is used to estimate β and is computed as follows:

$$\underset{\beta}{\operatorname{argmin}} \frac{1}{u} \sum_{t=1}^{u} \tau_t \Phi(Y_t, \beta^T X_t) + \lambda \sum_{b=1}^{B} w_b ||\beta^{(b)}||_2 \tag{5.5}$$

where $\tau_t \geq 0$ and $w_b \geq 0$ for all t, b.

Let the training data (X_t, Y_t) be defined as D_T and $L(\beta|D_T)$ be empirical loss i.e.,

$$L(\beta|D_T) = \frac{1}{u} \sum_{t=1}^{u} \tau_t \Phi(Y_t, \beta^T X_t) \tag{5.6}$$

The $L(\beta|D_T)$ is differentiable as a function of β and is denoted as $\nabla L(\beta|D_T)$. The loss for logistic regression is defined as follows:

$$\Phi(Y_t, \beta^T X_t) = \log(1 + \exp(-Y_t \beta^T X_t)) \tag{5.7}$$

After differentiating we obtain $\Phi'(Y_t, \beta^T X_t) = -Y_t \frac{1}{1+\exp(Y_t \beta^T X_t)}$

$$\nabla L(\beta | D_T) = \frac{1}{u} \sum_{t=1}^{u} \tau_t Y_t X_t \frac{1}{1 + \exp(Y_t X_t^T \beta)} \tag{5.8}$$

The group-lasso penalized logistic regression model is defined by using the λ values $\lambda = \frac{\max_{b=1,....B} || \left[\nabla L \hat{\beta} | D_T\right]^{(b)} ||_2}{w_b}$ and $w_b \neq 0$.

$$\hat{\beta}_{glasso} = \log \left(\frac{\sum_{Y_t=1} \tau_t}{\sum_{Y_t=-1} \tau_t} \right) \tag{5.9}$$

The selection probability $S(f_a)$ of the CpG sites with group lasso is as follows.

$$S(f_a) = \frac{1}{L} \sum_{j=1}^{L} \frac{1}{O} \sum_{k=1}^{O} \mathrm{I}(\beta_{jka} \neq 0), \text{ for } a = 1, 2, ..., d. \tag{5.10}$$

where resampling count is denoted by L, λ values is defined by O, f_a is the CpG indexed as c, and β_{ija} is regression coefficient of cth CpG site. For every resampling L, O number of λ are considered and a 10-fold cross-validation procedure is applied during the model building process. The true important features referred as CpG sites are ranked based on their significance level.

Algorithm 3 Proposed RGLR algorithm

Step 1: Obtain 70% of data samples from training data X_t by random sampling with replacement.

Step 2: The frequency of each CpG sites is counted in the 100 different models of λ values.

Step 3: Step 1 and Step 2 are repeated for 100 times.

Step 4: Rank the CpG sites based on the scores computed for every CpG site from Equation (5.10).

Step 5: Apply the top-ranked significant features referred as CpG site to various classifiers to build predictive models.

5.4 Results

The synthetic data were generated based on the correlation structures such as 0.1, 0.4, and 0.7. To illustrate the performance of the method we also used real DNAmeth data GSE26126.

5.4.1 Simulated data results

The CpGs are ranked with the proposed RGLR method and the other popular current FS methods. The true positive rate (TPR) rate is calculated based on averages taken over 100 iterations in all the FS methods. The TPR is calculated from 25 to 2500 ranked features with increment of 25 and is plotted as line plots, as shown in Figure 5.1. From the line plot, we see that the RGLR is performing better than the other FS methods such as ING, MRMR, FIS, and RFVI in all the correlation scenarios of 0.1, 0.4, and 0.7. The RGLR is even successful in selecting the 90% of the true important features within the top 200 ranked features. We also showed the boxplots in the form of percentiles, where we see that the RGLR method is showing the best performance compared to the other methods by a larger margin. The boxplots can also be seen in Figure 5.1.The RGLR method, with average percentiles of 0.06, 0.02, and 0.01 across the three different correlated data, was the best in comparison to the other FS methods, and it is shown in Table 5.1. Also, when we consider the SD of all the FS methods, we see that the RGLR is having less variation in selecting the significant variables on average across 100 times.

Table 5.1: True variable ranking percentile of FS methods on different synthetic data

Correlation	RGLR (SD)	FIS (SD)	ING (SD)	RFVI (SD)	MRMR (SD)
0.1	0.06 (0.11)	0.29 (0.30)	0.20 (0.25)	0.28 (0.27)	0.32 (0.31)
0.4	0.02 (0.05)	0.22 (0.29)	0.10 (0.17)	0.18 (0.22)	0.23 (0.29)
0.7	0.01 (0.02)	0.19 (0.28)	0.04 (0.10)	0.10 (0.16)	0.14 (0.23)

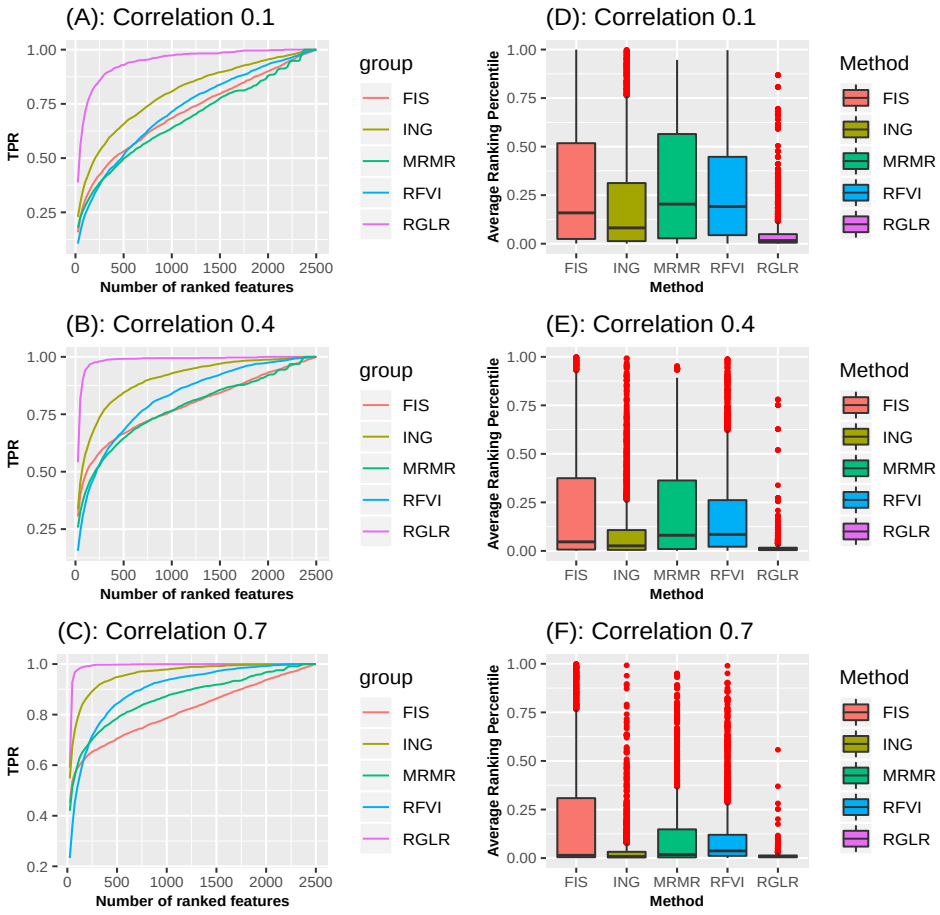

Figure 5.1: Line plot and boxplot showing the true positive rate (TPR) of the FS methods across different correlated synthetic data. The evaluations are made by taking averages over 100 times in each of the correlated data. FIS is Fisher's score, ING is Information Gain, MRMR is Minimum Redundancy Maximum Relevance, RGLR is proposed rasampling of group lasso ranking, and RFVI is Random Forest Variable Importance.

Figure 5.2 shows the line plots and boxplots of the prediction accuracies for various classifiers with the RGLR and other current methods on the data where the correlation was set to 0.1. We see that the accuracies of NB, SVM, and RF classifiers with RGLR are better compared to the other FS methods such as FIS, ING, MRMR, and RFVI. The line plots show the classification averages calculated at an interval of 25 features among

the top 200 ranked features. The RGLR on all the classifiers showed consistently superior performance along with all the top 200 ranked features. The boxplots show the peak accuracies obtained by the FS methods with different classifiers among the top 200 ranked features across 100 iterations.

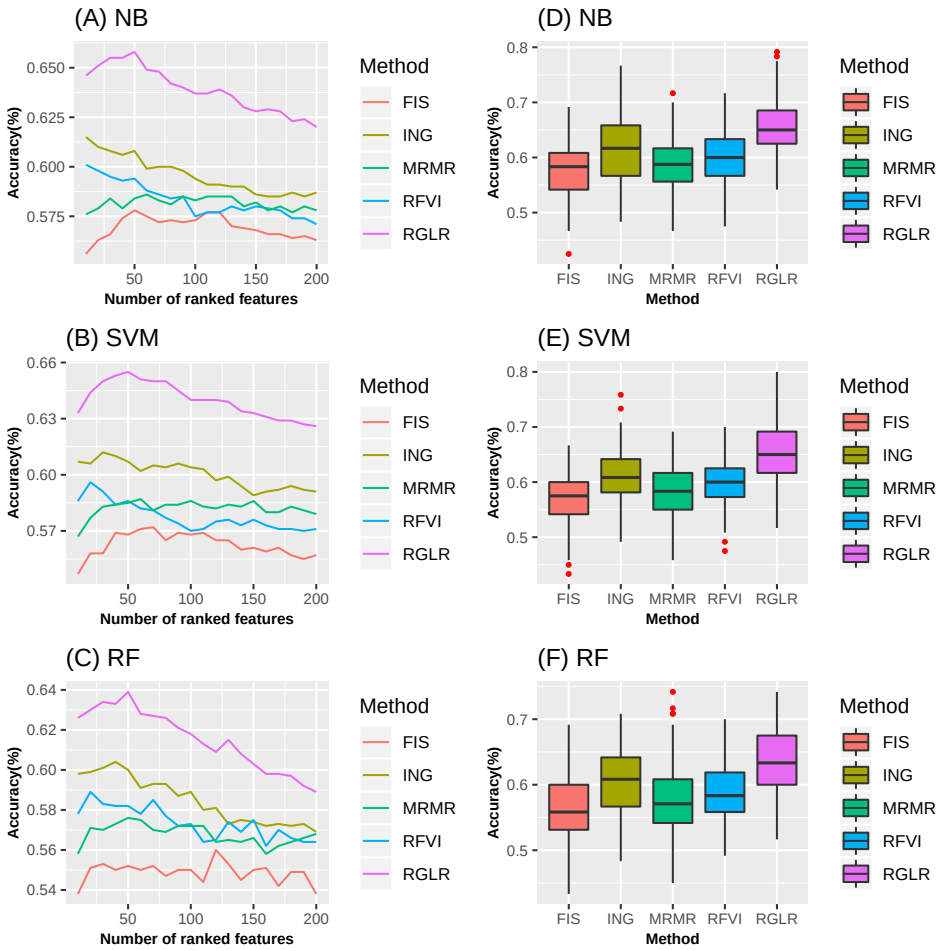

Figure 5.2: Average ACC of the FS methods with different classification methods on the simulated dataset when correlation = 0.1. NB is Naive Bayes, SVM is Support Vector Machines, RF is Random Forests. The evaluations are made by taking averages over 100 times of 0.1 simulated data.

Similar results were seen in simulated data with correlation of 0.4 and 0.7 and the plots

are shown in Figure 5.3 and Figure 5.4. From the line plots in both figures, we see that the ACC of various classifiers with RGLR is consistently better than other methods among the top 200 ranked features. The best performance obtained by each combination of FS and classification methods was recorded in every iteration and plotted as boxplots for both datasets and classifiers with RGLR method showed superior performance in both of these data.

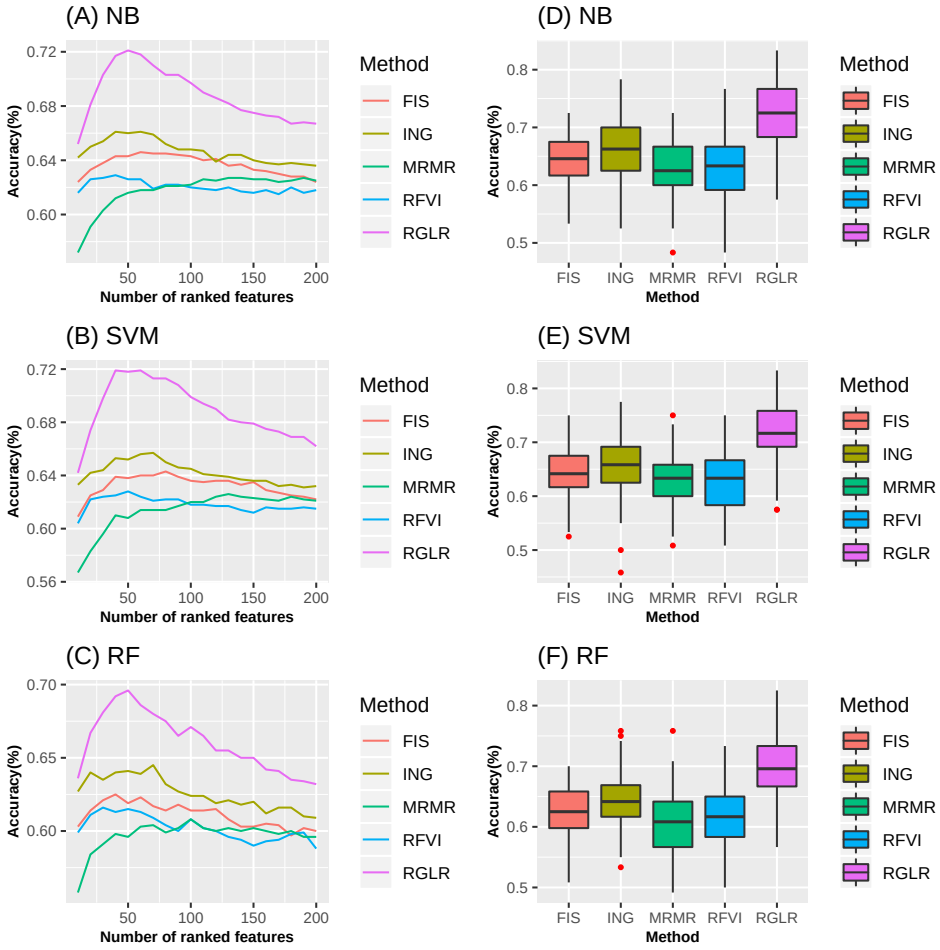

Figure 5.3: Average ACC of the FS methods with various classification methods on the simulated dataset when correlation = 0.4

To further estimate the metrics of various classifiers with the proposed RGLR and other

current FS methods, we made use of SENS, SPEC, and GMM along with the ACC and the results are reported in Table 5.2. The classifiers of NB, SVM, and RF with the proposed RGLR method achieved an accuracy of 65.80, 65.50, and 63.90, respectively, has better accuracy than that of the other FS methods in data with a correlation of 0.1. The SVM classifier with RGLR method achieved the highest accuracy in the synthetic dataset with a correlation of 0.7. The RF classifier with MRMR attained an accuracy of 0.671, which was the lowest among all the combinations of various classifiers with FS methods in the dataset when the correlation was set as 0.7. Under the data with a correlation of 0.4, all classifiers with RGLR method outperformed the combination of classifiers with MRMR with a margin of around 10%.

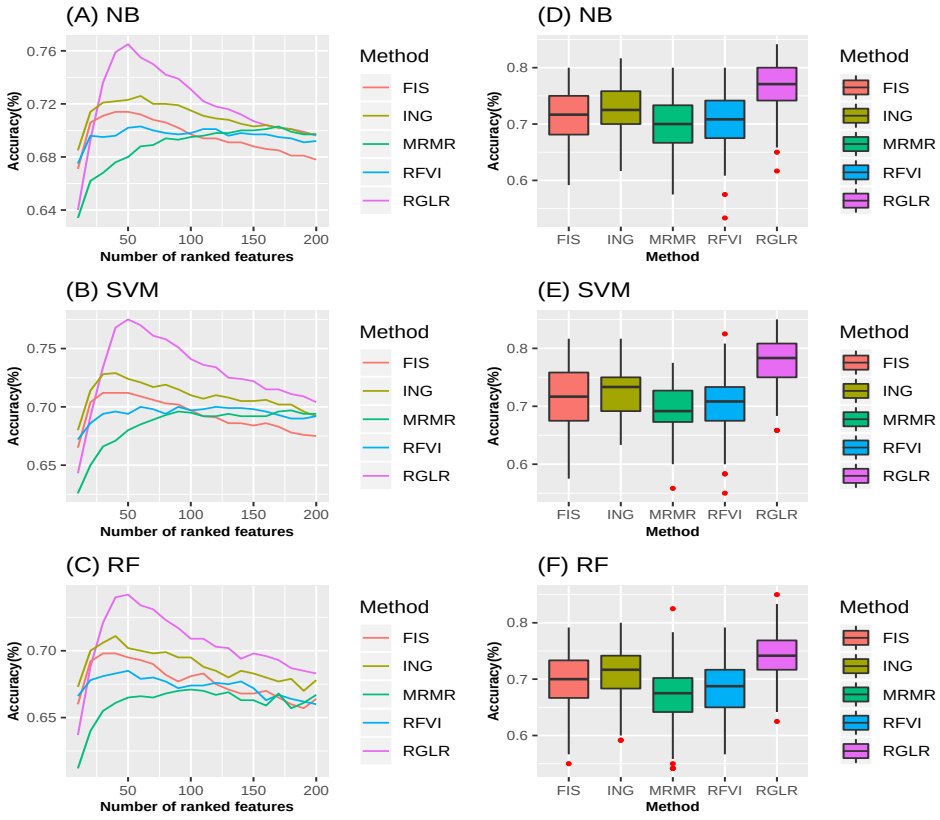

Figure 5.4: Average ACC of different classification methods with FS methods on the simulated data when correlation = 0.7

Table 5.2: Evaluation of classification methods with FS methods in simulated data with correlation structures of 0.1, 0.4, and 0.7. The calculations of mean and SD was taken from 100 times of repetitions.

Cors	CM	Metrics	**RGLR**	FIS	ING	RFVI	MRMR
0.1	NB	ACC(SD)	**0.658**(0.054)	0.578(0.053)	0.615(0.061)	0.601(0.049)	0.586(0.048)
		SENS(SD)	0.660(0.079)	0.577(0.081)	0.619(0.076)	0.609(0.081)	0.589(0.075)
		SPEC(SD)	0.659(0.076)	0.585(0.076)	0.614(0.090)	0.595(0.076)	0.586(0.077)
		GMM(SD)	0.657(0.062)	0.578(0.062)	0.614(0.053)	0.599(0.053)	0.585(0.054)
	SVM	ACC(SD)	**0.655**(0.054)	0.572(0.044)	0.612(0.048)	0.596(0.042)	0.587(0.049)
		SENS(SD)	0.657(0.082)	0.569(0.091)	0.616(0.076)	0.604(0.069)	0.592(0.072)
		SPEC(SD)	0.659(0.080)	0.583(0.088)	0.611(0.084)	0.591(0.069)	0.587(0.088)
		GMM(SD)	0.655(0.061)	0.571(0.057)	0.610(0.053)	0.595(0.050)	0.586(0.048)
	RF	ACC(SD)	**0.639**(0.048)	0.560(0.051)	0.604(0.049)	0.589(0.046)	0.576(0.055)
		SENS(SD)	0.635(0.088)	0.561(0.100)	0.604(0.081)	0.594(0.080)	0.580(0.096)
		SPEC(SD)	0.650(0.084)	0.570(0.106)	0.610(0.085)	0.589(0.079)	0.580(0.089)
		GMM(SD)	0.638(0.059)	0.558(0.046)	0.603(0.048)	0.588(0.052)	0.575(0.059)
0.4	NB	ACC(SD)	**0.721**(0.056)	0.646(0.041)	0.661(0.047)	0.629(0.057)	0.627(0.049)
		SENS(SD)	0.720(0.066)	0.644(0.066)	0.660(0.073)	0.634(0.072)	0.629(0.081)
		SPEC(SD)	0.724(0.072)	0.650(0.069)	0.666(0.067)	0.627(0.089)	0.630(0.076)
		GMM(SD)	0.721(0.062)	0.645(0.048)	0.661(0.062)	0.628(0.062)	0.627(0.051)
	SVM	ACC(SD)	**0.719**(0.054)	0.643(0.045)	0.657(0.052)	0.628(0.054)	0.626(0.047)
		SENS(SD)	0.718(0.073)	0.644(0.073)	0.662(0.074)	0.633(0.073)	0.630(0.091)
		SPEC(SD)	0.723(0.069)	0.647(0.082)	0.656(0.082)	0.628(0.087)	0.631(0.081)
		GMM(SD)	0.719(0.060)	0.642(0.054)	0.657(0.048)	0.628(0.065)	0.626(0.048)
	RF	ACC(SD)	**0.696**(0.051)	0.625(0.046)	0.645(0.044)	0.616(0.052)	0.608(0.051)
		SENS(SD)	0.693(0.077)	0.627(0.084)	0.647(0.082)	0.619(0.083)	0.603(0.102)
		SPEC(SD)	0.705(0.080)	0.629(0.081)	0.650(0.087)	0.619(0.083)	0.623(0.091)
		GMM(SD)	0.696(0.057)	0.624(0.061)	0.645(0.050)	0.615(0.052)	0.608(0.056)
0.7	NB	ACC(SD)	**0.765**(0.043)	0.714(0.047)	0.726(0.041)	0.703(0.050)	0.703(0.046)
		SENS(SD)	0.769(0.063)	0.721(0.063)	0.728(0.067)	0.705(0.068)	0.703(0.065)
		SPEC(SD)	0.763(0.064)	0.706(0.067)	0.727(0.059)	0.704(0.071)	0.706(0.067)
		GMM(SD)	0.764(0.050)	0.713(0.051)	0.726(0.050)	0.703(0.050)	0.703(0.052)
	SVM	ACC(SD)	**0.775**(0.044)	0.712(0.052)	0.729(0.042)	0.700(0.052)	0.697(0.043)
		SENS(SD)	0.778(0.070)	0.717(0.071)	0.726(0.073)	0.702(0.074)	0.697(0.077)
		SPEC(SD)	0.774(0.062)	0.710(0.073)	0.735(0.066)	0.701(0.073)	0.703(0.078)
		GMM(SD)	0.774(0.056)	0.712(0.050)	0.728(0.050)	0.699(0.050)	0.697(0.038)
	RF	ACC(SD)	**0.742**(0.044)	0.698(0.049)	0.711(0.046)	0.685(0.047)	0.671(0.054)
		SENS(SD)	0.743(0.068)	0.700(0.07)	0.711(0.085)	0.678(0.083)	0.674(0.092)
		SPEC(SD)	0.746(0.074)	0.701(0.078)	0.717(0.066)	0.699(0.076)	0.677(0.088)
		GMM(SD)	0.743(0.055)	0.698(0.046)	0.711(0.049)	0.685(0.055)	0.671(0.052)

5.4.2 Application to real data

To estimate the performance of different classifiers with RGLR and other FS methods the real DNAmeth data of prostate cancer with the accession number GSE26126 is used in this study. This dataset has 27578 CpG sites, which are reduced to 2500 significant CpG sites with MMLE as part of initial pre-processing. The absolute pairwise correlation among CpG sites in prostate cancer data is shown using the density plot and boxplot. The average absolute correlation between CpG sites is 0.61. This value lies within the range of correlated data of 0.1 to 0.7 generated in the simulation studies and is shown as a density plot and boxplot in Figure 5.5.

The variation of accuracies by several classifiers with the FS methods across 100 data splittings is recorded as boxplots in Figure 5.6. From the line plots, we see that the performance of the RGLR method, when applied to the classifiers such as NB, SVM, and RF, outperforms the other combination of classifiers and FS.

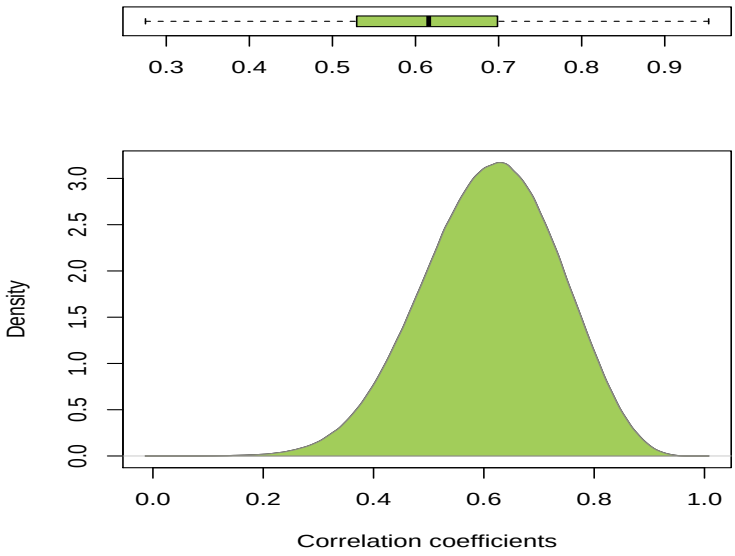

Figure 5.5: Absolute pairwise correlation coefficients between 2500 CpG sites in prostate cancer data the average is 0.61.

Figure 5.6 also shows the boxplots, where the highest performance attained by different

classifiers with all the FS methods in each of the iterations are recorded. From these plots, we see that the proposed RGLR method, when applied on all the classification methods, is comparatively better than other methods of combination. Table 5.3 shows the averages of ACC, SENS, SPEC, and GMM calculated for all the methods. The classifiers of NB, SVM, and RF with proposed RGLR method attained the accuracies of 0.898, 0.906, and 0.910, which were comparatively better compared to other methods of combination.

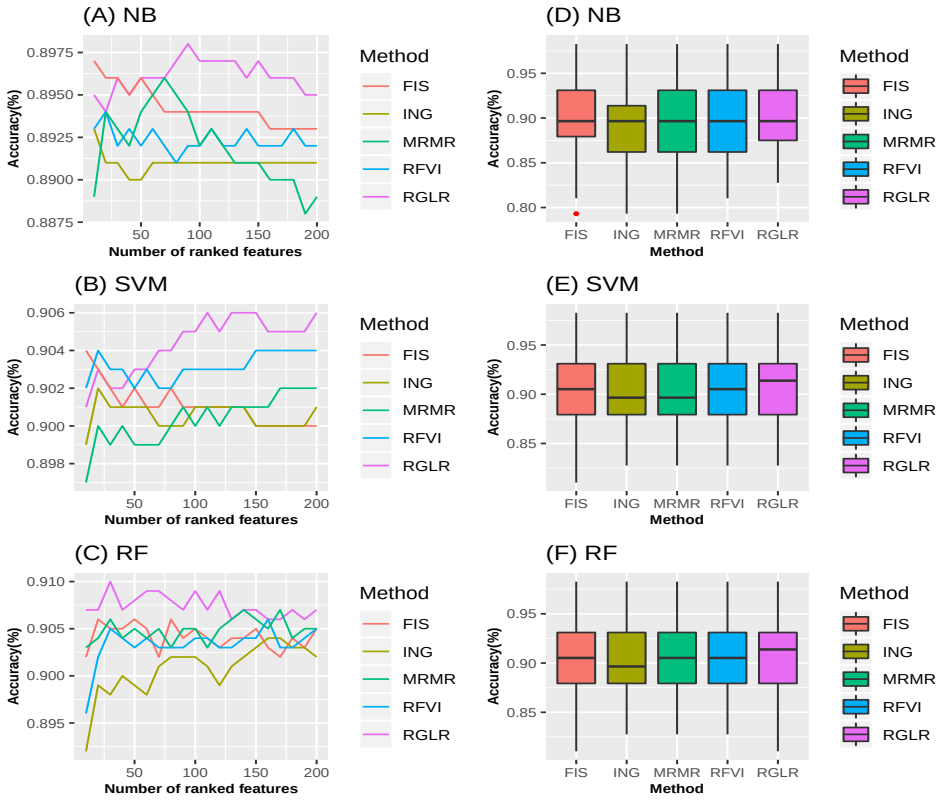

Figure 5.6: Average ACC of classification methods with FS methods on the real data GSE26126.

Table 5.3: Evaluation of classification and FS methods in real data GSE26126. The metrics are calculated by taking averages over 100 times of iterations.

CM	Metrics	**RGLR**	FIS	ING	RFVI	MRMR
NB	ACC(SD)	**0.898**(0.037)	0.897(0.040)	0.893(0.038)	0.894(0.040)	0.896(0.036)
	SENS(SD)	0.882(0.058)	0.874(0.065)	0.873(0.065)	0.871(0.061)	0.883(0.057)
	SPEC(SD)	0.912(0.050)	0.918(0.046)	0.912(0.051)	0.916(0.052)	0.907(0.051)
	GMM(SD)	0.896(0.041)	0.895(0.037)	0.891(0.037)	0.892(0.041)	0.894(0.039)
SVM	ACC(SD)	**0.906**(0.037)	0.904(0.039)	0.902(0.038)	0.904(0.038)	0.902(0.036)
	SENS(SD)	0.873(0.060)	0.877(0.063)	0.877(0.061)	0.875(0.059)	0.871(0.061)
	SPEC(SD)	0.936(0.041)	0.930(0.043)	0.926(0.046)	0.933(0.043)	0.931(0.043)
	GMM(SD)	0.903(0.040)	0.902(0.038)	0.900(0.041)	0.902(0.041)	0.899(0.039)
RF	ACC(SD)	**0.910**(0.039)	0.906(0.037)	0.904(0.038)	0.906(0.039)	0.907(0.037)
	SENS(SD)	0.885(0.064)	0.884(0.065)	0.881(0.063)	0.881(0.061)	0.884(0.059)
	SPEC(SD)	0.932(0.045)	0.927(0.048)	0.926(0.048)	0.929(0.047)	0.928(0.046)
	GMM(SD)	0.908(0.042)	0.904(0.039)	0.902(0.039)	0.904(0.039)	0.905(0.041)

5.5 Discussion

We examined the performance of classifiers such as NB, SVM, and RF with the proposed RGLR method and other present FS methods like FIS, ING, RFVI, and MRMR using simulated and real data. The proposed RGLR method ranks the features known as CpG sites in the context of DNAmeth through resampling of group lasso method. The ranked features are applied to the classifiers, and the performance is evaluated for top significantly ranked features. The RGLR method is having the higher performance of true important feature selection and eventually leading to attain better prediction accuracy with all the classifiers on DNAmeth data in comparison to other combination of FS with classifiers.

For training the models, 70% of the training data was used, and the performance of all the combinations of methods, including the RGLR, was evaluated using ACC, SENS, SPEC, and GMM. The RGLR showed the best performance in all of these metrics, as seen in Table 5.2 and Table 5.3.

To exemplify the performance of RGLR method, we used simulation data with three different correlation structures and real DNAmeth data. The simulation data was having low = 0.1, medium = 0.4, and high = 0.7 correlation structures. The real data had a correlation of 0.6, which falls within the range of the simulation setup. At first, we showed the superior performance of the RGLR in selecting the true important variables in Figure 5.1 where the TPR of the RGLR is higher than the popular FS methods such as FIS, ING, RFVI, and MRMR. This shows that the proposed RGLR method is the best FS method compared to other popular FS methods in selecting more number of true important features in highly correlated DNAmeth data with group structures. Also, different classifiers with the RGLR method achieved best performance compared to other methods in both simulation and real prostate DNAmeth datasets. The various classifiers with FS methods showed much instability. Therefore, the classifiers have lower performance when there are fewer true important variables and more amount of noise in the model. The SVM classifier was relatively consistent in both simulation and real data with all the FS methods.

The proposed RGLR method attained most of the true important features within the top 50 to 100 ranked features as seen in the line plots and box plots in Figure 5.1. The positive effect of having such significant CpGs within short-range of ranked CpGs in a high-dimensional setting is seen in Figures 5.2–5.4. The accuracies described in the form of plots show that the RGLR method with all the classifiers in all the simulation scenarios attained the peak accuracies within the top 50 ranked features. The other FS methods, such as FIS, ING, RFVI, and MRMR, had very low TPR and retrieved all the important variables when all the features in the data were added. Henceforth, the number of noisy features that were unrelated to the outcome was added in comparison to the true important features. This resulted in the downward estimate when higher number of unrelated features

were added. The RGLR method acquired more number of significant CpG sites compared to other FS methods. However, the RGLR is computationally intensive because of the resampling technique. The RGLR method by removing the irrelevant features helps in boosting classification accuracy and also to reduce the required computation time, as the number of CpGs is reduced.

5.6 Conclusion

The proposed RGLR method is developed for filtering out the irrelevant CpGs from the highly correlated DNAmeth data. We show better performance of the RGLR method in detecting the significant features in comparison to other competitive FS methods through extensive simulation studies. To illustrate the classification performance, the classifiers such as NB, SVM, and RF were applied on the top-ranked CpGs obtained from the proposed RGLR along with the other popular FS methods such as FIS, ING, RFVI, and MRMR. The various classifiers with the proposed RGLR method showed better performance of ACC, SENS, and SPEC, and GMM in comparison to other combination of FS with classifiers in both simulation and real data. However, the proposed RGLR method involves computationally intensive tasks because of the resampling approach. As future research, we intend to focus on improving the computational efficiency and significant feature detection power.

Chapter 6

Identification of Hub Genes and Role of Apoptosis and Oxidative Stress Related Pathways in Different Stages of Colorectal Cancer Through Integrated Bioinformatics Approach

6.1 Introduction

Colorectal cancer (CRC) is diagnosed as the third most common cancer in males and females in the United States. It is also the third leading cause of cancer-related mortality in the year 2020 [Siegel et al., 2020]. CRC has become a major health issue, with expected estimates showing 147,950 individuals to be diagnosed and 53,200 deaths to be expected from the disease in 2020 alone [Siegel et al., 2020]. Between 2008 to 2017, the CRC-related mortality in the younger than 55 age group has increased by 1.3% per year [Siegel et al., 2020]. The risk of developing CRC is contingent on certain conditions that can be categorized into environmental, lifestyle, and genetic factors [Aran et al., 2016], which concur with tumor initiation, progression and metastasis [Aran et al., 2016, Ding et al., 2020]. Most of the CRC starts as a polyp that grows on the colon or rectum part's inner lining. Depending on the type of polyps, some of them change into adenocarcinoma or carcinoma. Colorectal adenocarcinomas form in the glands and can be treated through surgical or therapeutic procedures such as radiation therapy or chemotherapy. Adenomas or adenomatous polyps are precancerous state polyps that may turn into carcinoma over a specific time. These adenomas can be removed surgically, and the prognosis is known to be favorable in the early stages [Brody, 2015]. The early detection of colorec-

tal polyps through endoscopic screening procedure and their removal before they develop into cancers is crucial [Thorsteinsson and Jess, 2011]. However, detection of CRC in early stages is often misdiagnosed as the symptoms are unusual [Xu et al., 2020] and lead to cancer metastasize to organs, which significantly reduces the survival rate of CRC patients [Brody, 2015, Yuan et al., 2020]. Therefore, it is of great importance to explore molecular mechanisms of CRC proliferation and apoptosis [Ding et al., 2020] to improve diagnosis and treatment by understanding CRC gene expression. Besides, the identification of multiple genes and pathways that may be involved in the occurrence and progression of CRC is also important to develop more optimal therapeutic techniques [Yuan et al., 2020, Chen et al., 2019c].

With the advancement of high-throughput sequencing technology, microarray and next-generation RNA sequencing (RNA-seq) have become popular in under-standing gene expression studies [Saito et al., 2018, Deshiere et al., 2019]. The high-throughput gene expression data is screened through bioinformatics approaches to identify hub genes related to the CRC. The hub genes were considered important candidates for biomarkers in the development and progression of CRC [Xu et al., 2020]. Some of the studies conducted microarray analysis to identify hub genes in CRC [Solé et al., 2014, Zhao et al., 2019, Chen et al., 2019b]. However, these studies have some limitations. First, most CRC studies considered a single microarray dataset for finding hub genes but data from a single microarray platform might not be accurate for identifying hub genes [Solé et al., 2014, Zhao et al., 2019, Chen et al., 2019b]. Second, some studies used multiple microarray datasets to find hub CRC genes [Xu et al., 2020, Yuan et al., 2020, Chen et al., 2019c, Guo et al., 2017]. However, none of these studies compared datasets with samples from different CRC types, such as adenoma and adenocarcinoma. Lastly, there is a lack of literature on the role of oxidative stress-induced cellular survival pathways in CRC. These limitations emphasize the necessity to utilize diverse datasets generated from different high-throughput technologies containing different tissue sample types. These datasets will be used to identify potential biomarkers of CRC in this study and discover the associated links

between hub genes and oxidative stress and apoptosis pathways.

In this study, we used ten microarray datasets obtained from gene expression omnibus (GEO) and two RNA-seq datasets from the cancer genome atlas (TCGA) to identify hub genes associated with CRC. The datasets were from different platforms such as GPL570, GPL16699, GPL4133, GPL3282, GPL15207, and Illumina HiSeq. These datasets were assigned to three groups, normal tissue vs. adenomas, normal vs. adenocarcinoma, and normal vs. carcinoma. The differentially expressed genes (DEGs) in each dataset were identified, and the robust rank aggregation (RRA) algorithm was used to integrate the gene lists across different groups. A candidate gene list for each group was generated that had the most significantly expressed genes. Functional enrichment of the candidate genes in each group was analyzed using the Gene Ontology (GO) and Kyoto Encyclopedia of Genes and Genomes (KEGG) pathways. Further, the protein-protein interactions were identified with Cytoscape software, and the hub genes for each group were determined through a clustering approach. The hub genes found in our study were associated with several important pathways related to CRC, such as the p53 signaling pathway and some hub genes having a link with oxidative stress and apoptosis pathways were shown. The potential biomarkers for each group identified in our study may help promote early diagnosis and treatment of CRC.

6.2 Materials and Methods

6.2.1 Acquisition of Microarray and RNA-seq data.

The gene expression profiles of colorectal cancer (CRC) were obtained from the gene expression omnibus (GEO) database (`http://www.ncbi.nlm.nih.gov`) and the cancer genome atlas (TCGA) database from the genomic data commons (GDC) data portal (`https://portal.gdc.cancer.gov/`). The gene expression profiles of 13 CRC datasets were divided into three groups. The first group (G1) has three datasets that consist of

colorectal adenoma (CA) samples, group two (G2) had six datasets related to colorectal adenocarcinoma (CAC) samples. Finally, four of the colorectal carcinoma (CC) datasets were assigned in group3 (G3). These datasets were chosen such that there were at least ten samples in cases or control groups. The detailed description of the datasets used in this study is shown in Table 6.1.

Table 6.1: Description of the datasets used in the study.

Groups	Dataset	#Samples (Cases/ Control)	#Genes	Source	DEGs (UR/DR)
G1	GSE8671	64 (32/32)	54675	[Sabates-Bellver et al., 2007]	5498 (1355/4143)
(CA)	GSE20916	69 (45/24)	27697	[Skrzypczak et al., 2010]	2761 (1290/ 1471)
	GSE89076	80 (41/39)	58717	[Satoh et al., 2017]	7538 (2844/ 4694)
G2	GSE20842	130 (65/65)	40645	[Gaedcke et al., 2010]	3122 (1432/1680)
(CAC)	GSE20916	60 (36/24)	27697	[Skrzypczak et al., 2010]	3556 (1820/1736)
	GSE39582	585 (566/19)	54675	[Marisa et al., 2013]	3389 (1582/1807)
	GSE110225	34 (17/17)	54675	[E.I. et al., 2019]	1109 (478/631)
	TCGAREAD	519 (478/41)	56499	[Muzny et al., 2012]	6703 (3180/3523)
	TCGACOAD	176 (166/10)	56493	[Chang et al., 2013]	6813 (3291/3522)
G3	GSE3964	30 (18/12)	23232	[Graudens et al., 2006]	483 (158/325)
(CC)	GSE113513	28 (14/14)	49395	Unpublished	2864 (1151/1713)
	GSE32323	585 (17/27)	54675	[Khamas et al., 2012]	4671 (4530/141)
	GSE21510	148 (123/25)	54675	[Dong et al., 2007]	7720 (4383/3137)

6.2.2 Identification of differentially expressed genes (DEGs) in Microarray GEO datasets.

The pre-processing step includes normalization and log2 conversion for each microarray GEO datasets' raw matrix data. All the tissue samples were based on the Affymetrix Array's different platforms, as shown in Table 1. The Biobase and GEOquery packages in Bioconductor R were used to load and pre-process the data. The difference in the expression levels between different cases and control samples for each of the groups (CA vs. normal, CAC vs. normal, CC vs. normal) were analyzed separately using the linear models for microarray data (LIMMA) R package [Ritchie et al., 2015] in Bioconductor to select the statistically significant DEGs. The DEGs were ranked based on the adjusted P-value with the Benjamini & Hochberg False discovery rate method. We defined the cut-off criteria as adjusted P-value < 0.05 and $|$ FC (fold change)$| > 2$ to filter statistically significant DEGs.

6.2.3 Identification of differentially expressed genes (DEGs) in RNA-seq TCGA datasets.

The RNA-seq samples of data type HTSeq-Counts were used for analysis. The CAC samples were collected from TCGA Colon adenocarcinoma (TCGACOAD) and TCGA rectal adenocarcinoma (TCGAREAD) datasets. The TCGAbiolinks package in Bioconductor was used to obtain the data. The edgeR and Limma packages were used for filtering and identification of the DEGs. The lowly expressed genes were filtered out using the edgeR package. Composition biases between the libraries were eliminated using the trimmed mean of m-values (TMM) normalization. The variance modeling at the observational level (VOOM) function from the Limma package transforms the read counts into logCPMs (log Counts per millions). The generalized linear model using weighted least squares method is applied to the VOOM transformed data to test for differentially expressed genes between CAC vs. normal samples in the RNA-seq datasets from group G2. The threshold of adjusted P-value < 0.05 and $|$ FC (fold change)$| > 2$ is set to filter the statistically significant DEGs in each

of the RNA-seq datasets.

6.2.4 Integration of ranked lists of DEGs in groups G1, G2, and G3.

Each dataset had a different number of DEGs (Table 1). We obtained three ranked list of DEGs for three datasets in G1, six ranked list of DEGs for G2, and four ranked DEGs list for G3. Next, to integrate the different lists of DEGs into a robust individual list of DEGs for each group, we applied the robust rank aggregation (RRA) algorithm [Kolde et al., 2012]. The ranked lists of DEGs in a group were considered, and for each gene, the RRA algorithm looked at how the gene is positioned in the ranked lists of DEGs and com-pared this to the baseline case where all the preference lists are randomly shuffled [Kolde et al., 2012]. Finally, a P-value is assigned for all the genes. This shows how much better the gene is positioned in the ranked lists than expected by chance. The P-value for each gene shows the significance in the final robust ranked list.

Overall, the RRA algorithm is based on a probabilistic model for aggregating the genes from different datasets into a final robust ranked list. This method is computationally efficient and statistically robust. The "RobustRankAggreg" package in R was used to conduct the gene integration in groups G1, G2, and G3.

6.2.5 Functional and pathway enrichment analysis.

The database for annotation, visualization, and integrated discovery (DAVID, `https://david.ncifcrf.gov/home.jsp`, version 6.8) is a publicly available bioinformatics database. This database helps to identify the various biological pathways of DEGs through a set of functional annotation tools. Gene ontology (GO, `http://geneontology.org/`) classifies the description of gene function into three categories: biological process (BP), cellular component (CC), and molecular function (MF). The Kyoto Encyclopedia of Genes and Genomes (KEGG, `https://www.genome.jp/kegg/`) database resource provides an

understanding of high-level gene functions and biological signaling pathways. The GO term and KEGG pathway enrichment analyses were performed with the help of the DAVID bioinformatics tool. The terms with P-value < 0.05 were considered to be statistically significant.

6.2.6 Protein-Protein Interaction.

The search tool for the retrieval of interacting proteins (STRING-db, `https://string-db.org/`, version 11.0) is used to identify the various protein-protein interaction (PPI) networks of DEGs based on the confidence score set to medium = 0.7. To analyze the PPI network from String-db, we used Cytoscape (`https://cytoscape.org/`), an open-source bioinformatics software for loading, visualizing, and integrating complex PPI network. The StringApp plugin in Cytoscape was used to load the PPI network from Stringdb, and the network analyzer plugin was applied to measure the degree of interaction be-tween nodes and the network with upregulated and downregulated genes was dis-played. The MCODE plugin was also applied in Cytoscape to analyze the network further. MCODE uses a clustering algorithm to generate the network clusters to find the densely connected regions. The key clusters in the network were filtered with degree cutoff = 2., node score cut off = 0.2, k-core = 2, and max.depth = 100.

6.2.7 Hub genes screening and analysis.

We used the network analyzer algorithm to compute the number of connected pairs of nodes, the average number of neighbors, and node degrees. The node degree represents the interaction score assigned for each gene. The genes were ranked based on the degree of interactions among them. The DEGs with highest degree of interaction are defined as hub genes [Nangraj et al., 2020, Hu et al., 2020]. We determined the hub genes (top forty DEGs) from each group ranked based on the highest degree of interaction, using the network analyzer results. Functional enrichment of hub genes was further analyzed

and noted. We also used MCODE algorithm to verify the hub gene results further. The MCODE algorithm grouped the genes from the original network into modules and ranked them based on degrees. The top two clusters were selected from each of the groups to display the interactions.

6.2.8 Identification of top-ranked significant genes (TSGs) in GEO datasets using the Resampling-based lasso feature selection (RLFS) approach.

The significantly expressed genes in high-throughput data can also be found using machine learning approaches [Patil and Kim, 2020, Patil et al., 2020c]. The RLFS method is based on the lasso penalized regression method and the resampling approach employed to obtain the ranked important features using the frequency [Patil and Kim, 2020]. The RLFS method was applied to select the TSGs from each of the GEO datasets. The procedure to select the TSGs from each of the GEO datasets from G1, G2, and G3 is described as follows: At first, the original data were randomly divided into 75% for the training and 25% for the testing set. As the first filtering step, 75% of the training data were given to the marginal maximum likelihood estimator (MMLE) to overcome the redundant noisy genes, and the genes were ranked based on their level of significance. The top 10000 ranked significant genes were filtered from the MMLE and applied to the RLFS method as the second filtering step. After filtering through RLFS, the TSGs were obtained. For standard comparison of the performance of the model and subsidizing the effects of the data splitting, the RLFS model was built using the 10-fold cross-validation procedure, and 100 times of resampling was carried out to rank the genes based on their level of significance. This procedure was repeated for 100 iterations. The ranked genes in each iteration were taken, and a majority voting procedure was applied to select the significant genes that appeared in most of the iterations. The genes that were selected 95 or more times out of 100 iterations were considered as the TSGs.

Table 6.2: Description of the Oncomine datasets used in the study.

Groups	Dataset	#Samples (Cases/ Control)	Oxidative stress DEGs (UR/DR)	Apoptosis DEGs (UR/DR)
Oncomine Group-1 (CA)	Sabates (GSE8671)	64 (32/32)	29 (21/08)	208 (142/66)
	Skrzypczak (GSE20916)	69 (45/24)	24 (15/09)	189 (105/ 84)
	Skrzypczak2 (GSE20916)	15 (5/10)	22 (12/10)	176 (90/ 86)
Oncomine Group-2 (CAC)	Dulak (GSE36458)	122 (62/60)	19 (10/09)	175 (93/82)
	Gaedcke (GSE20842)	130 (65/65)	39 (23/16)	199 (107/92)
	Kaiser (GSE5206)	54 (49/5)	37 (20/17)	225 (117/108)
	Kurashina (GSE11417)	184 (94/90)	29 (18/11)	173 (111/62)
	Skrzypczak (GSE20916)	60 (36/24)	27 (15/12)	187 (101/86)
	TCGA CRC	184 (162/22)	38 (21/17)	257 (120/137)
	TCGA CRC2	970 (389/581)	33 (17/16)	209 (108/101)
Oncomine Group-3 (CC)	Graudens (GSE3946)	30 (18/12)	16 (09/07)	64 (20/44)
	Hong (GSE9348)	82 (70/12)	28 (16/12)	180 (110/70)
	Skrzypczak2 (GSE20916)	15 (5/10)	30 (15/15)	188 (97/91)

6.2.9 To identify oxidative stress-response and apoptosis associated genes in CRC with Oncomine.

The Oncomine database (`https://www.oncomine.org/`) was used to analyze the genes related to oxidative stress and apoptosis survival pathways. The CRC datasets in Oncomine were downloaded and grouped into three groups as previously. As shown in Table 2, G1, G2, and G3 contain the CA, CAC, and CC samples respectively. The concept-based filters

of response to oxidative stress and apoptosis were applied separately on the datasets to find the DEGs related to each concept. The threshold of and adjusted P-value < 0.05 and $|$ FC (fold change)$| > 2$ is set to filter the statistically significant DEGs related to oxidative stress and apoptosis.

6.3 Results

6.3.1 Genes differentially expressed in Colorectal Adenoma (G1) datasets:

The details of the G1 datasets are described in Table 6.1. The DEGs were identified using the Limma procedure for all the datasets. From Table 6.1, we see that the datasets GSE8671 and GSE89076 are having more than 5000 DEGs. The Robust rank aggregation (RRA) method [Kolde et al., 2012] uses a probabilistic model for aggregation robust to noise and facilitates the calculation of significance probabilities for all the elements in the final ranking. We applied the RRA algorithm using the "RobustRankAggreg" package in R to conduct the gene integration across all the G1 datasets. The P-value cut-off was set to 0.05 to filter the ranked DEGs. The number of robust DEGs for the G1, G2, and G3 datasets are shown in Table 6.3.

Table 6.3: Robust DEGs (P-value < 0.05) in each group from Table 6.1.

Groups	Robust UR	Robust DR	Total Robust DEGs
G1 (CA)	186	449	635
G2 (CAC)	499	494	993
G3 (CC)	206	79	285

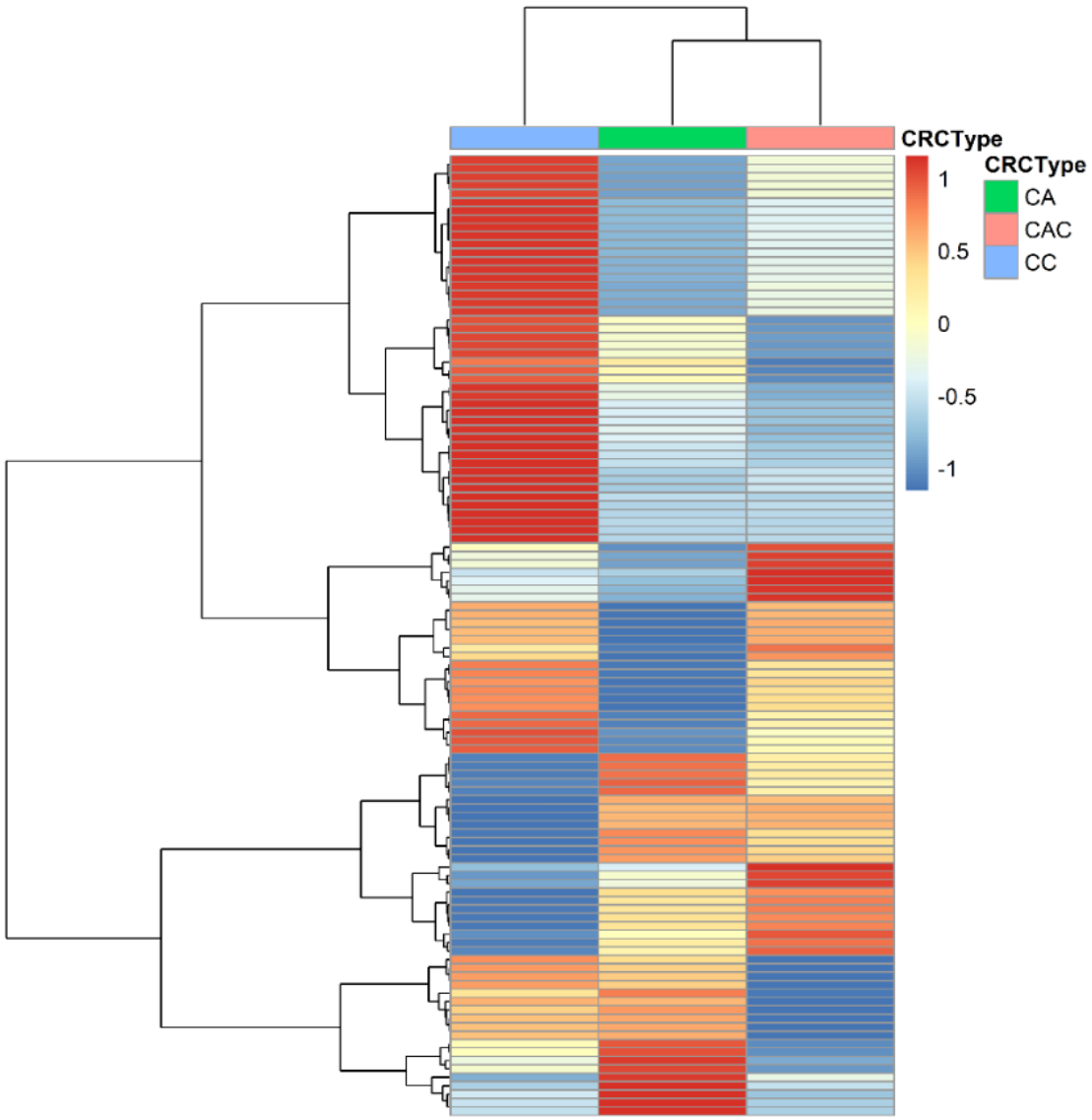

Figure 6.1: Heatmap of common genes in all groups.

We plotted heatmap to understand the difference in expression levels of common RRA genes in all groups. In Figure 6.1, we show the change in logFC among the common RRA genes in Colorectal adenoma, Colorectal adenocarcinoma, and Colorectal carcinoma.

Next, the common genes among the different datasets in upregulated (UR) and down-regulated (DR) were shown using the Venn diagram. There were 370 UR and 622 DR commonly found DEGs as displayed in Figure S1A and S1B, respectively. Further, the

intersection of these common UR and DR genes was taken with the RRA ranked genes to show the important genes. Finally, we see that the 154 and 403 RRA genes are shown in Figure 6.2 would be the most significant.

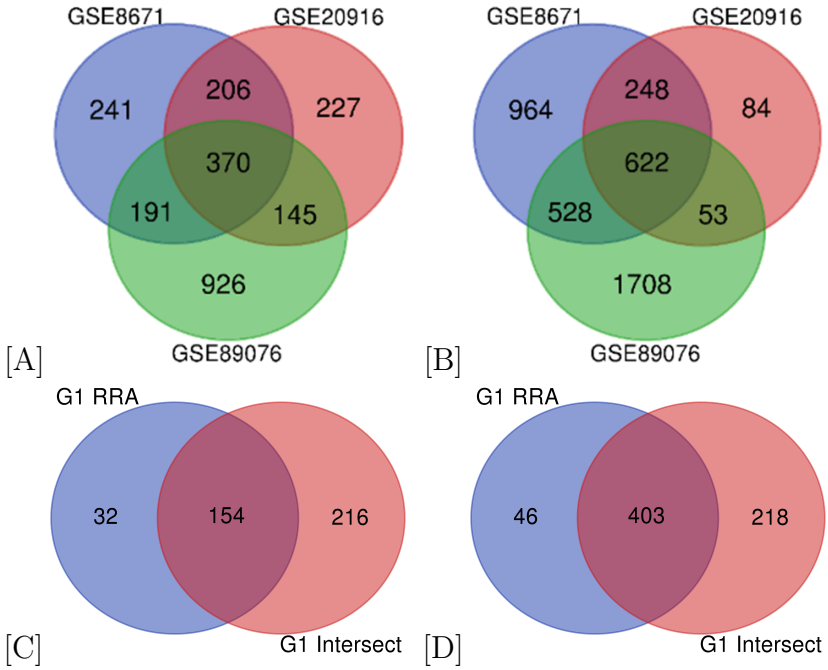

Figure 6.2: Results of the Venn diagram and RRA for G1. (A) UR common DEGs from G1 in the Venn diagram. (B) DR common DEGs from G1 in the Venn diagram. (C) The intersection of UR genes in Venn and RRA results. (D) The intersection of DR genes in Venn and RRA results. Abbreviations: RRA, Robust Rank Aggregation.

6.3.2 Genes differentially expressed in Colorectal Adenocarcinoma (G2) datasets:

The details of the G2 datasets are described in Table 6.1. The DEGs were identified using the Limma procedure for all the datasets. The TCGA COAD and READ datasets consist of around 3000 UR and DR genes. 153 UR genes are common to all the datasets in G2, as seen in Figure 6.3. Among the 153 UR genes, 136 of them were also found in the robust UR genes, as shown in Figure 6.5.

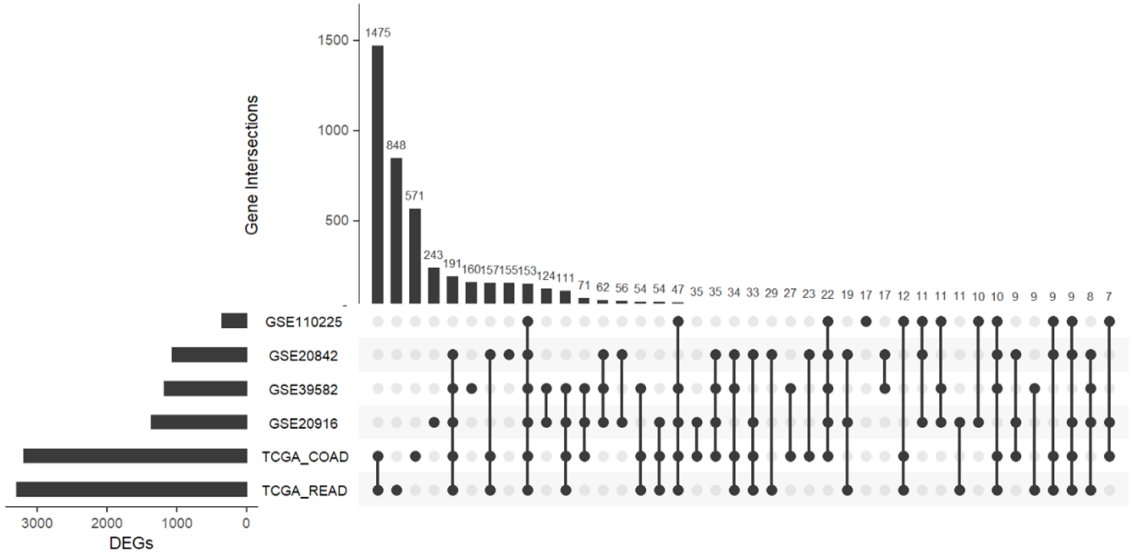

Figure 6.3: UpSetR plot showing the intersection of UR genes in G2.

In Figure 6.4, we see that 225 DR genes are unique to all the datasets in G2. Among these, 213 genes were also selected by the RRA method, as shown in Figure 6.5.

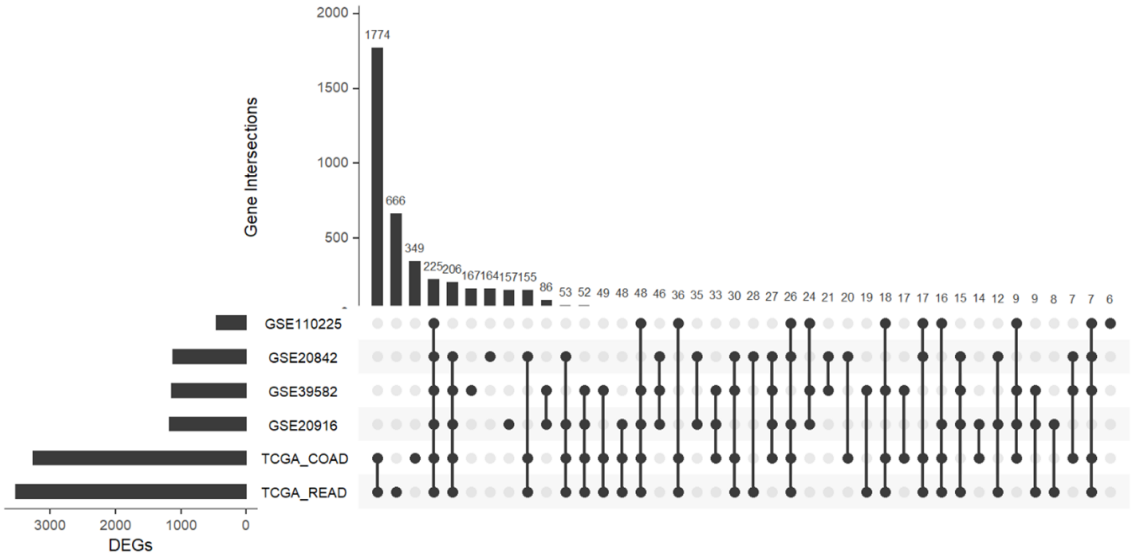

Figure 6.4: UpSetR plot showing the intersection of DR genes in G2.

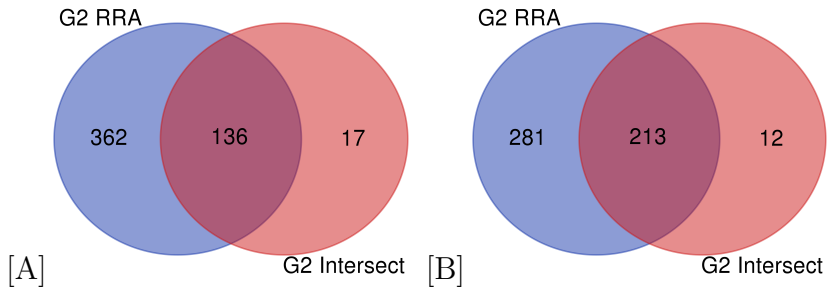

Figure 6.5: Venn diagram showing the intersection of UR and DR RRA genes in G2 datasets.

6.3.3 Genes differentially expressed in Colorectal Carcinoma (G3) datasets:

The description of the CC datasets can be found in Table 6.1. The count of DEGs for each of the G3 datasets is mentioned in Table 6.1. Here, we see that the dataset GSE21510 with 7720 DEGs is the largest in this group. The RRA procedure was applied to find the ranked integrated genes across all the G3 datasets. The P-value cut-off was set to 0.05 to filter the ranked DEGs. Venn diagram was plotted to show the overlapping genes among the datasets. There were three and one overlappings of up and downregulated DEGs found among the G3 datasets, as shown in Figure 6.6.

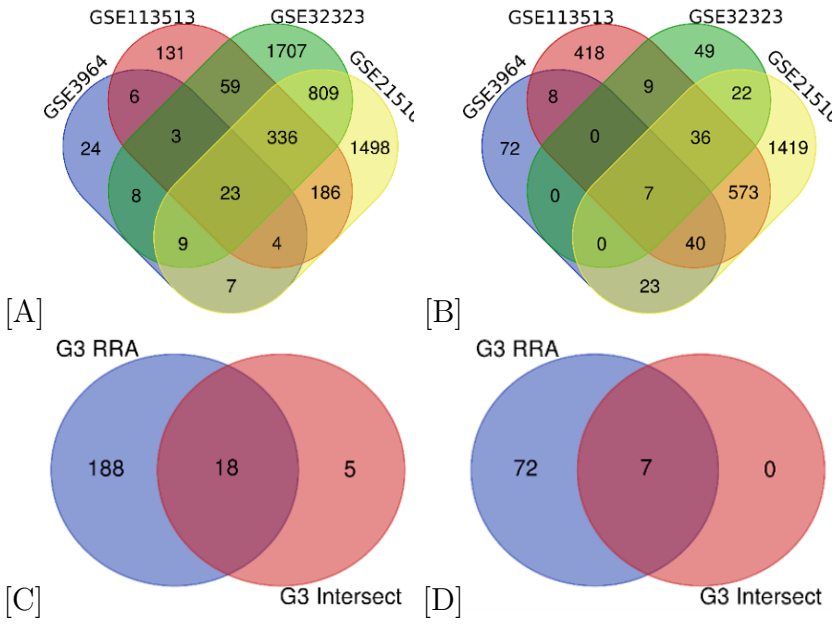

Figure 6.6: Results of the Venn diagram and RRA for G3. (A) UR common DEGs from G3 in the Venn diagram. (B) DR common DEGs from G3 in the Venn diagram. (C) The intersection of UR genes in Venn and RRA results. (D) The intersection of DR genes in Venn and RRA results.

6.3.4 Analyzing the performance of RLFS in all groups.

We applied the RLFS method to all the datasets in their corresponding groups to assess gene selection. The significant genes for each group were obtained using the RRA algorithm. We compared the list of RRA-RLFS genes with RRA-DEGs to interpret the results. We took an intersection of the DEGs found through standard procedures and significant genes found through the RLFS method to know the overlapping genes. We found all the RLFS genes in G1 and G2 DEGs, as shown in Figure 6.7. Similarly, we can see that about 22 genes from RLFS were also found in G3 DEGs, as shown in Figure 6.7.

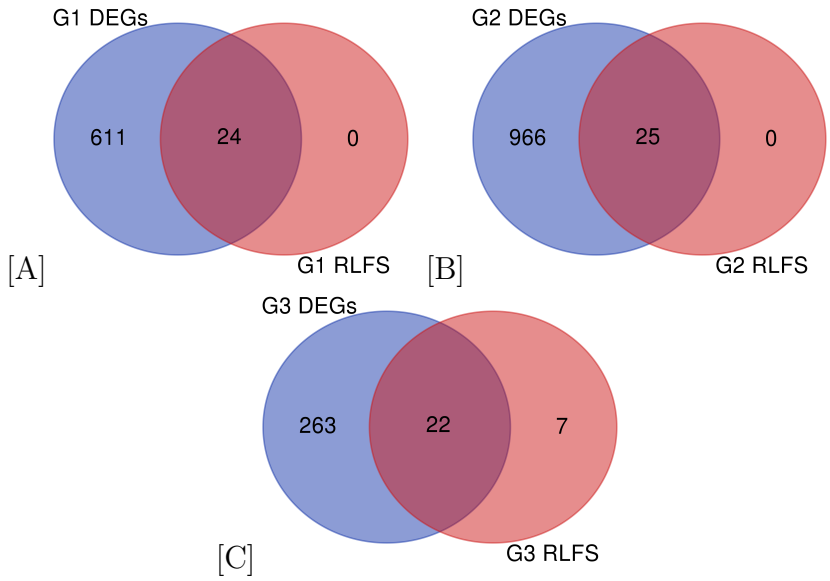

Figure 6.7: Results of the RLFS method, Intersect approach, and RRA algorithm. (A) The intersection of RRA DEGs and RRA RLFS genes in G1. (B) The intersection of RRA DEGs and RRA RLFS genes in G2. (C) The intersection of RRA DEGs and RRA RLFS genes in G3. Abbreviations: RLFS, Resampling based lasso feature selection.

6.3.5 GO enrichment analysis

The ranked RRA DEGs for each group G1, G2, and G3, including both UR and DR genes, were submitted separately to DAVID to retrieve the overrepresented GO categories and KEGG pathways. The summary of GO term enrichment analysis for the RRA UR and DR genes is shown in Figure 6.8 and Figure 6.9, respectively.

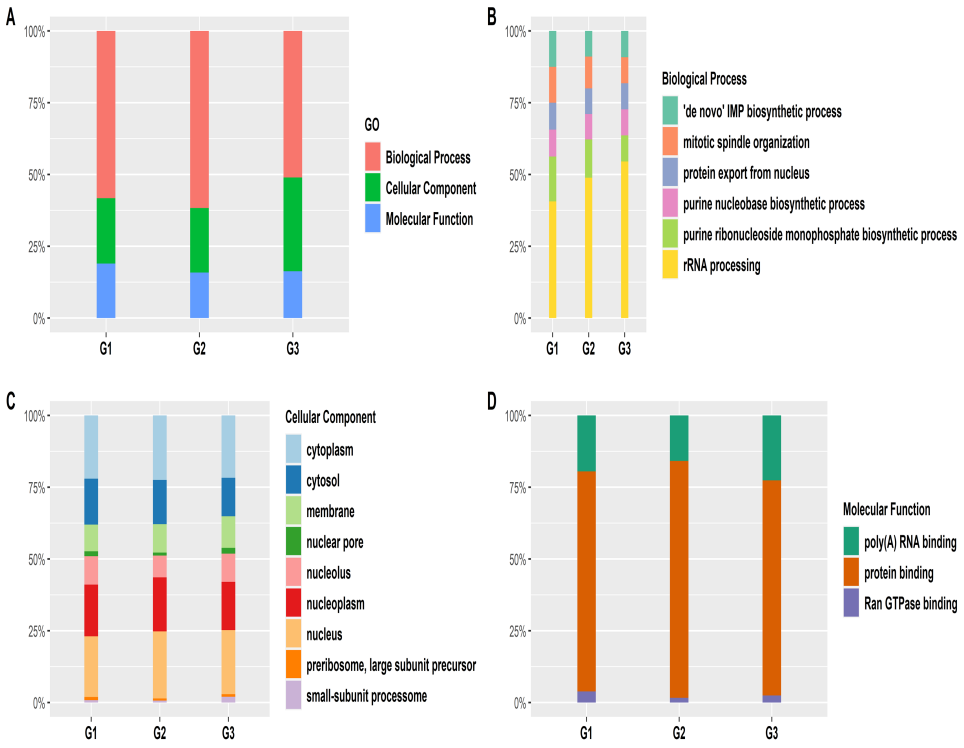

Figure 6.8: Functional group enrichment analysis of UR RRA genes in G1, G2, and G3. (A) Analyses of GO functional groups within the G1, G2, and G3. The Y-axis of these 100% stacked columns shows the percentage of genes that fall within each GO, functional group. (B-D) Common GO functionalities between all groups. (B) The GO biological process. (C) The molecular function. (D) The cellular component.

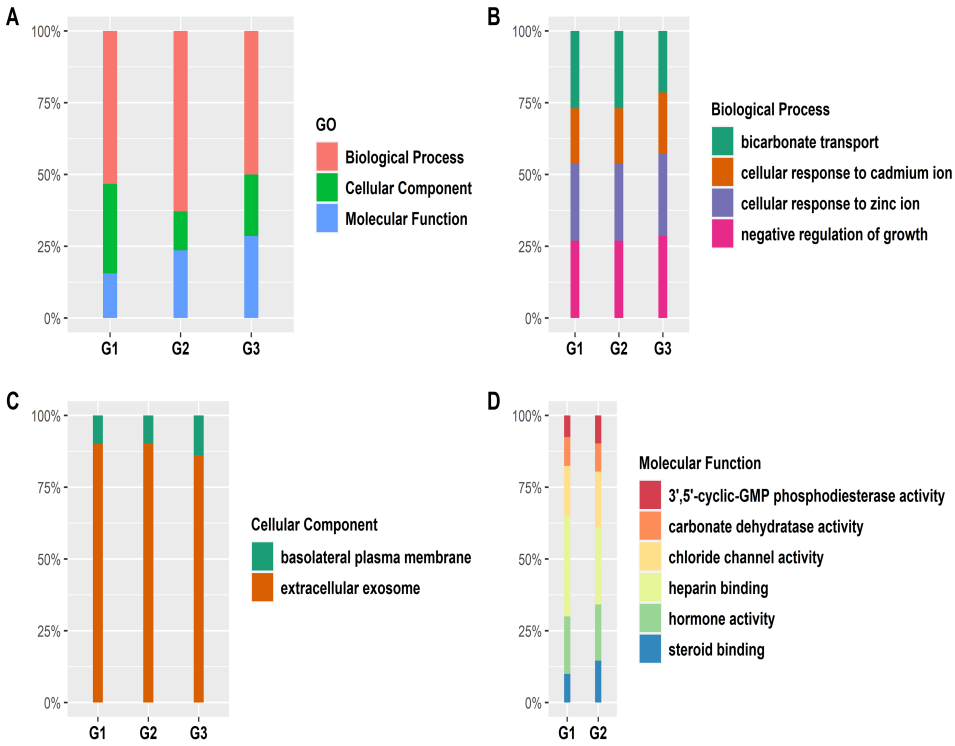

Figure 6.9: Functional group enrichment analysis of DR RRA genes in G1, G2, and G3. (A) Analyses of GO functional groups within G1, G2, and G3. The Y-axis of these 100% stacked columns shows the percentage of genes that fall within each GO functional group. (B-D) Common GO functionalities between all groups. (B) biological process. (C) molecular function. (D)cellular component.

GO enrichment analysis of UR genes

In the biological process of GO terms, the top 3 UR pathways (Figure 6.10) for G1 were rRNA processing, purine ribonucleoside monophosphate bio-synthetic process, and 'de novo' IMP biosynthetic process, the UR genes in G2 were mainly associated with cell division, mitotic nuclear division, and DNA replication, and finally in G3 the genes were associated with rRNA processing, maturation of SSU-rRNA from tricistronic rRNA transcript, and purine nucleobase biosynthetic process. Figure 6.10A shows that in G1-UR (CAC), cell division is significantly enriched, which is not seen in G2-UR and G3-UR. The

cell proliferation had a count of around 20 in G2-UR is seen to increase in G3-UR. The rRNA processing is seen in all three groups with the almost same count of genes. However, there is a change in expression. The G3-UR is highly expressed in comparison to the other two groups. In the cellular component ontology, the G1-UR genes were mainly enriched in the nucleolus, nucleoplasm, and cytosol. The G2-UR genes were enriched in the nucleoplasm, nucleolus, and nucleus. The G3-UR genes were enriched in the nucleolus, nucleoplasm, and small-subunit processome. We see that the nucleus and cytoplasm have a large number of genes involved in G2-UR compared to G1-UR and G3-UR (Figure 6.10B). Similarly, nucleoplasm is highly expressed in G2-UR compared to the other two groups. In the Molecular function analysis, the G1-UR genes were significantly enriched in poly(A) RNA binding, protein binding, Ran GTPase binding, and protein-arginine N-methyltransferase activity. The G2-UR were seen enriched in protein binding, ATP binding, and poly(A) RNA binding. The G3-UR were enriched in poly(A) RNA binding, snoRNA binding, and protein binding. There are more than 200 highly expressed genes (Figure 6.10C) in the protein binding of G2-UR, whereas, in G1-UR and G2-UR, there is relatively lower expression. Also, there is ATP binding interaction happening in G2-UR. However, there is no such activity reported in other groups.

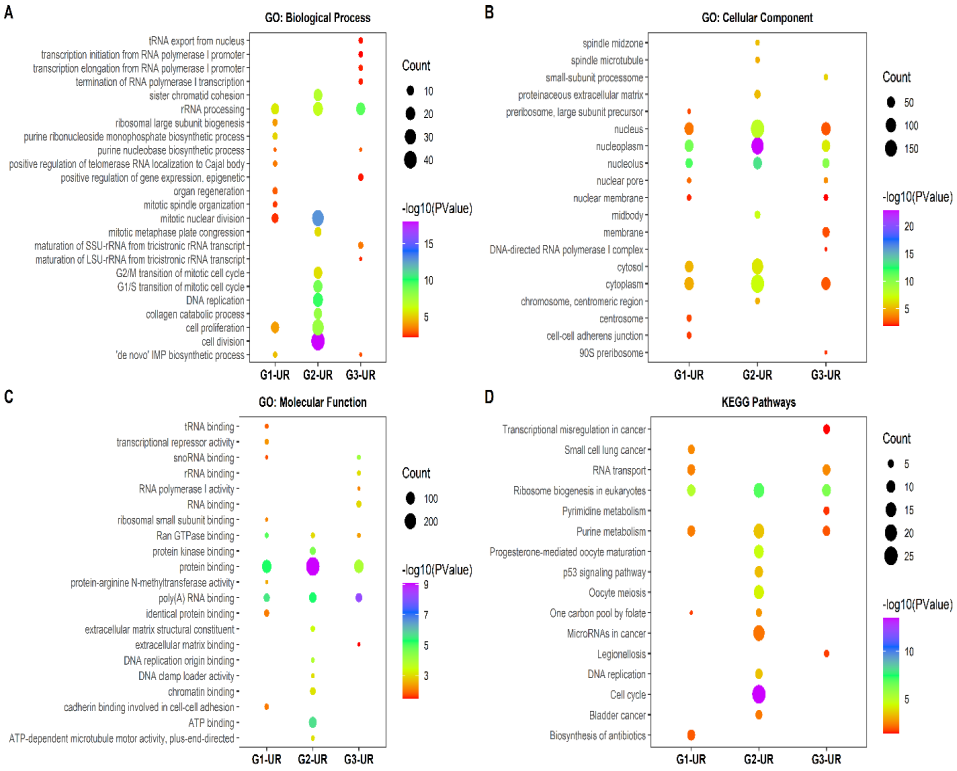

Figure 6.10: GO and KEGG pathway enrichment of RRA-UR genes. (A) GO enrichment of Biological process. (B) GO enrichment of Cellular components. (C) GO enrichment of Molecular function. (D) KEGG pathway enrichment. Abbreviations: UR, Upregulated; GO, gene ontology; KEGG, Kyoto encyclopedia of genes and genomes.

GO term enrichment analysis of DR genes.

In the biological process, the top G1-DR, G2-DR, and G3-DR genes (Figure 6.11A) were significantly enriched in the cellular response to zinc ion and negative regulation of growth, and these pathways are highly expressed in G3-DR compared to other groups. In G1-DR, many genes are involved in cell adhesion, which is not evident in G2-DR and G3-DR. In G1-DR, the extracellular exosome, brush border membrane, and apical plasma membrane was found to be the top significant pathways. The extracellular exosome, extracellular space, and integral component of the membrane were the top enriched pathways in G2-DR of the cellular component (Figure 6.11B). The extracellular exosome, chromaffin granule,

and basolateral plasma membrane were top pathways in G3-DR. The integral component of the membrane appears to be highly enriched in G2-DR compared to G1-DR. In the molecular function part (Figure 6.11C), the top pathways for G1-DR were heparin-binding, chloride channel activity, and carbonate dehydratase activity. The G2-DR were in chloride channel activity, carbonate dehydratase activity, and steroid-binding. The G3-DR were in xenobiotic-transporting ATPase activity, guanyl nucleotide-binding, and structural constituent of the myelin sheath.

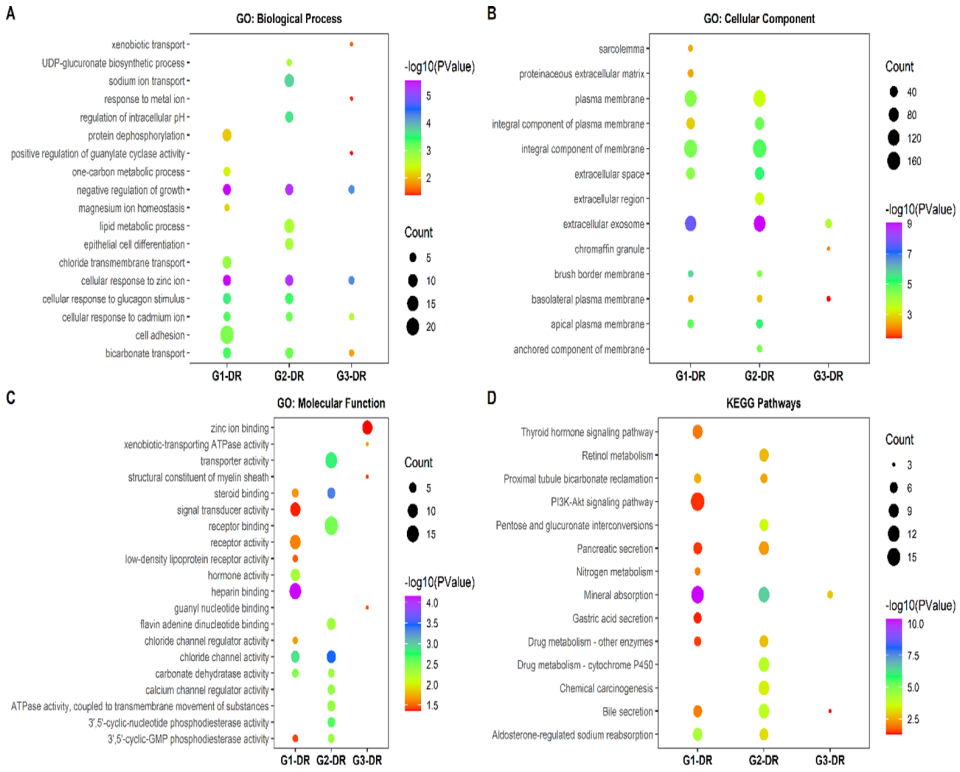

Figure 6.11: GO and KEGG pathway enrichment of RRA DR genes. (A) GO enrichment of Biological process. (B) GO enrichment of Cellular components. (C) GO enrichment of Molecular function. (D) KEGG pathway enrichment. Abbreviations: DR, Downregulated.

KEGG pathway analysis

The KEGG pathway analysis was performed to gain insights into the identified RRA-DEGs further. The target genes of the UR genes from G1 (Figure 6.10D) were significantly enriched in ribosome biogenesis in eukaryotes, small cell lung cancer, and RNA transport. The UR genes in G2 were found in the cell cycle, ribosome biogenesis, and oocyte meiosis in eukaryotes. Finally, the UR genes in G3 were found in ribosome biogenesis in eukaryotes, RNA transport, and purine metabolism. The ribosome biogenesis pathway was found in all the groups; however, it was highly expressed in G2-UR. The transcriptional misregulation in the cancer pathway was found only in G3-UR. The KEGG pathways for the DR genes (Figure 6.11D) in G1 were mainly enriched in mineral absorption, aldosterone-regulated sodium reabsorption, and proximal tubule bicarbonate reclamation. In G2, the DR genes were significantly enriched in mineral absorption, pentose and glucuronate interconversions, and drug metabolism - cytochrome P450. The target genes of the DR genes in G3 were enriched in mineral absorption and bile secretion. The mineral absorption is commonly found in all groups. However, it is highly enriched in G1-DR.

6.3.6 Protein-Protein Interaction (PPI) network construction and module selection for key genes screening.

PPI network construction and module selection in G1, G2, and G3.

PPI networks of DEGs in G1, G2, and G3 were constructed with the string app in Cytoscape. The original PPI network developed for each group had many nodes and edges among them. The nodes represent the proteins, and the edges are the interactions. The PPI network for G1 had 615 nodes and 846 edges (Figure 6.13A). The PPI net-work for G2 had 816 nodes and 3056 edges (Figure 6.14A). Finally, for G3, the PPI network had 280 nodes and 545 edges (Figure 6.15A). As seen in these figures, the networks are very dense, it is very difficult to interpret important information from these interaction networks. Therefore, we applied the network analyzer algorithm to filter the DEGs with a

high degree of interaction score. The top forty DEGs with high scores were de-termined as hub genes in each group, similar to that in studies done by Nangraj et al. (2020) and Hu et al. (2020). To further validate our results, we applied the MCODE algorithm to divide the original dense networks into modules. The top two clusters with high scores in each group were considered to be significant.

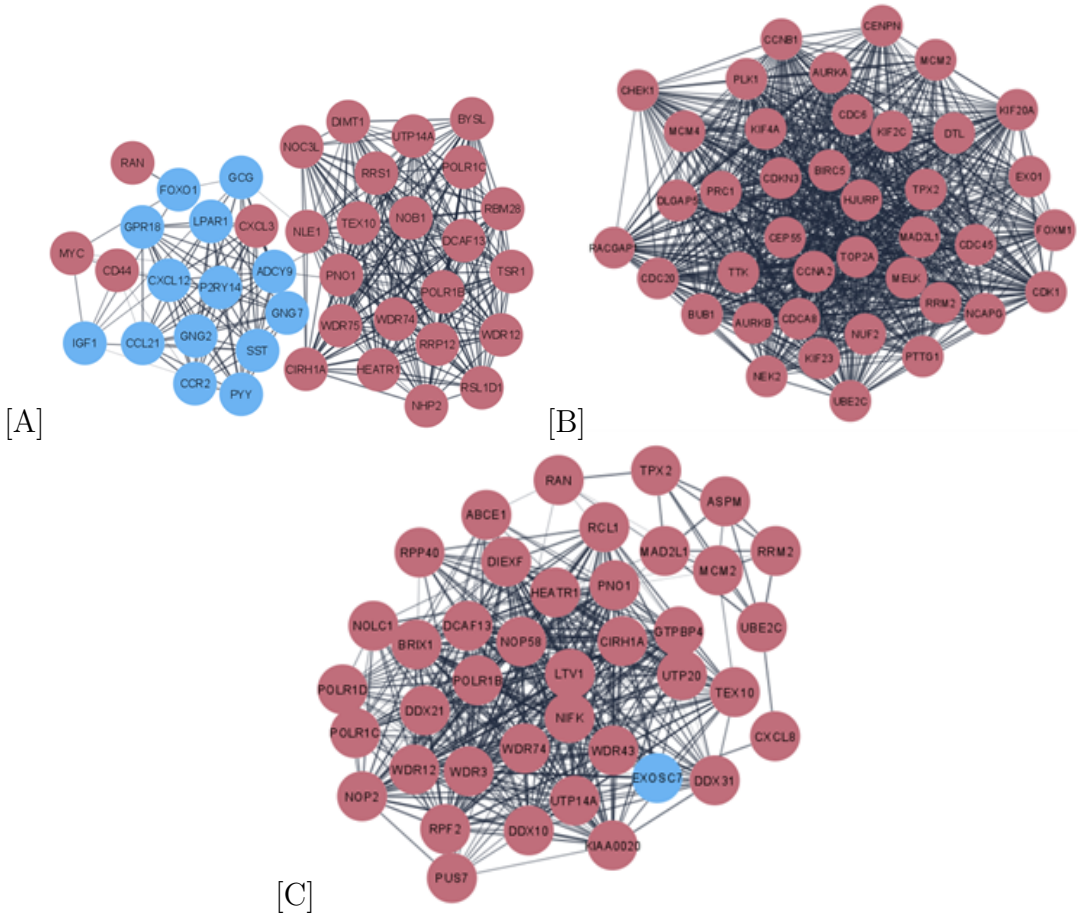

Figure 6.12: (A). Top forty hub genes in G1. (B) Hub genes in G2. (C) Hub genes in G3. Dark red color- UR; Blue color- DR;

The top forty genes with the highest degree of interaction from the original dense networks in each group were termed hub genes, the PPI network of hub genes in G1, G2, and G3 are shown in Figures Figure 6.12A, B, and C, respectively. In general, there were

more UR hub genes than DR hub genes in all three groups. However, in G1, which is the colorectal adenoma stage, we see that 35% of hub genes reported were DR (Figure 5A). In carcinoma stages of G2 and G3, we see that almost 99% of hub genes were UR.

Next, we used the MCODE algorithm to build the modular networks and compare the results with the hub genes obtained from the network analyzer method. The top two clusters created with MCODE from the DEGs network were selected based on their interaction scores. Cluster1 in G1 had 21 nodes and 195 edges (Figure 6.13B), whereas, cluster2 in G1 had 13 nodes with 77 edges (Figure 6.13C). These two clusters had all the hub genes determined earlier by the network analyzer methods Figure 6.12A.

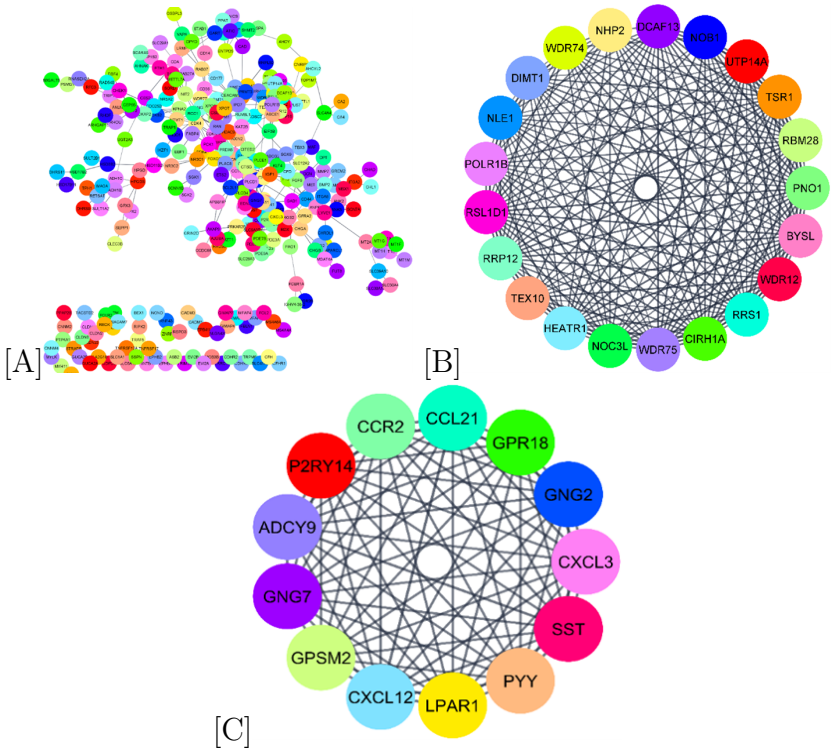

Figure 6.13: (A). PPI network of DEGs from G1 was constructed in Cytoscape with String-db. (B-C). The most significant module of DEGs in G1. Cluster1 and Cluster2 were obtained with MCODE plug-in in Cytoscape applied on the PPI network.

There were about 816 nodes and 3056 edges in G2 PPI network. The MCODE algo-

rithm generated the clusters, and the top two cluster networks were selected. The cluster1 (Figure 6.14B) in G2 had about 51 nodes and 1078 edges. The cluster2 (Figure 6.14C) had 23 nodes and 251 edges. Considering the high number of interaction scores in cluster1 and cluster2, the hub genes were determined Figure 6.12B.

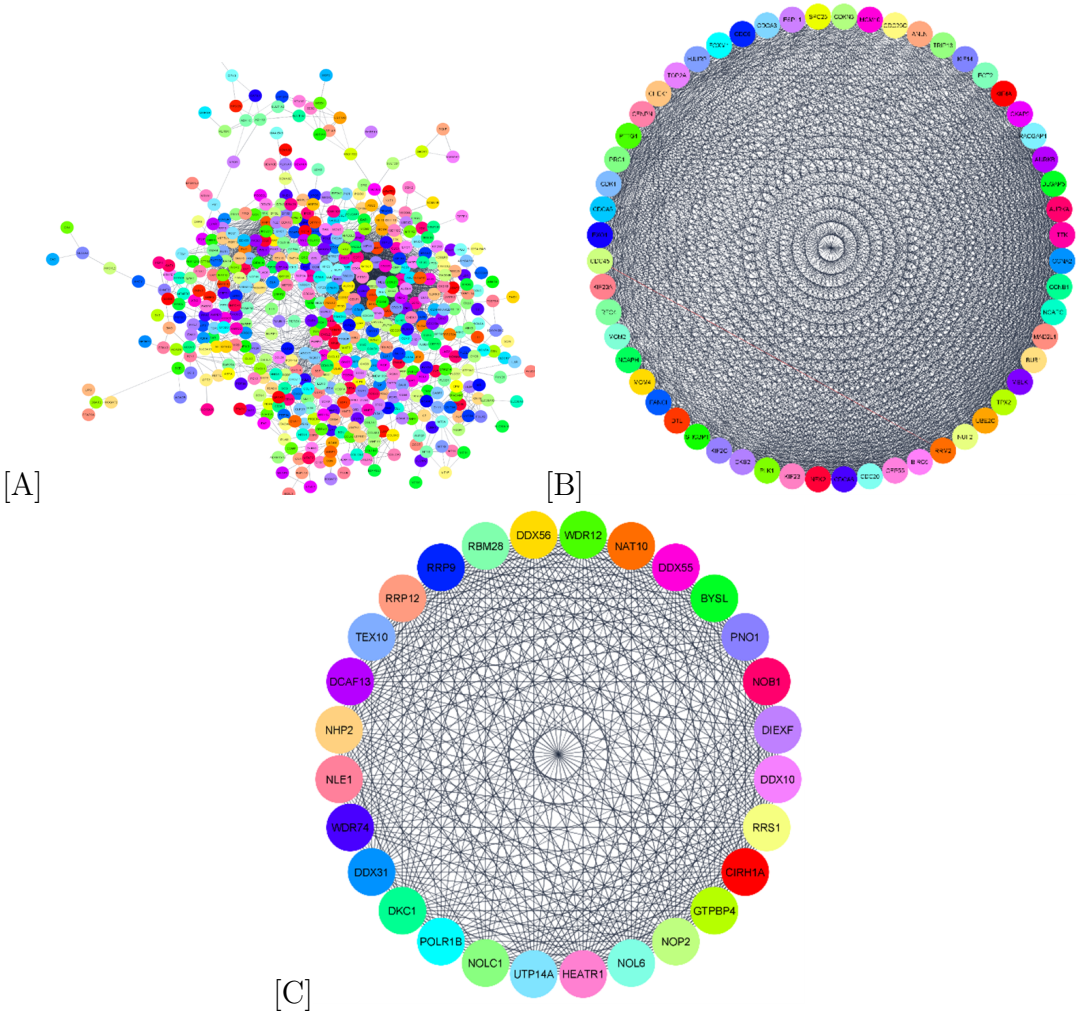

Figure 6.14: (A). PPI network of DEGs from G2 was constructed in Cytoscape with String-DB. DEGs- Differentially expressed genes. (B-C). The most significant module of DEGs in G2. Cluster1 and Cluster2 were obtained with MCODE plug-in in Cytoscape applied on the PPI network.

There was a total of 280 nodes and 545 edges in the original PPI network for G3. The

top two clusters created with MCODE from the PPI network were significant. Cluster1 (Figure 6.15B) had about 24 nodes and 252 edges. The cluster2 (Figure 6.15C) had 11 nodes with 43 edges. The nodes in cluster1 and 2 having a greater number of interactions were considered hub genes Figure 6.12C.

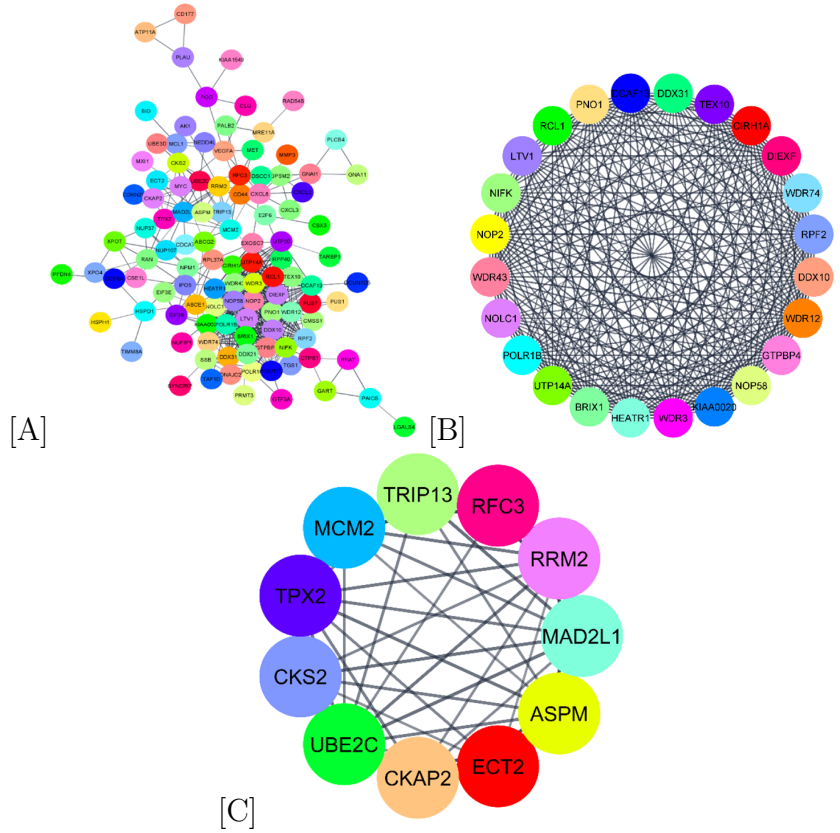

Figure 6.15: (A). PPI network of DEGs from G3 was constructed in Cytoscape with String-db. (B-C). The most significant module of DEGs in G3. Cluster1 and Cluster2 were obtained with MCODE plug-in in Cytoscape applied on the PPI network.

6.3.7 Functional enrichment of hub genes

The top 40 genes with the highest degree of interaction in each group were termed hub genes. The KEGG pathways of enrichment in the top 40 hub genes are shown in Table 6.4. The ribosome biogenesis in eukaryotes, chemokine signaling pathway, pathways in cancer,

and P13K-Akt signaling pathways are the top pathways for the hub genes involved in group G1.

In G2, the cell cycle, oocyte meiosis, and p53 signaling pathways are highly enriched. Some of the key hub genes involved in the cell division pathway were CDK1, CDC6, CDC20, CDC45, CHEK1, BUB1, and MAD2L1. The hub genes involved in the p53 signaling pathway were CCNB1, RRM2, CHEK1, CDK1. Some of the hub genes involved in these two pathways were also part of oxidative stress and apoptosis. The CDC25C, RRM2, and CDK4 genes were part of the oxidative stress pathway. The hub genes CHEK1, CDK1, CCNB1, and CDC20, were also enriched in the apoptosis pathway.

Finally, the ribosome biogenesis in eukaryotes with a count of nine genes was the most significantly pathway in G3 Table 6.4. Additionally, the hub genes RRM2, POLR1B, POLR1C, and POLR1D are enriched in pyrimidine and purine metabolism pathways in G3. RRM2 hub gene is also known to be linked to oxidative stress.

Table 6.4: KEGG pathways related to hub genes.

Groups	Term	KEGG Pathway	Count	PValue
G1	hsa03008	Ribosome biogenesis in eukaryotes	7	2.92E-07
	hsa04062	Chemokine signaling pathway	7	2.48E-05
	hsa05200	Pathways in cancer	8	0.000208
	hsa04151	PI3K-Akt signaling pathway	5	0.025838
	hsa04727	GABAergic synapse	3	0.032272
	hsa05032	Morphine addiction	3	0.036574
	hsa04713	Circadian entrainment	3	0.03956
	hsa04723	Retrograde endocannabinoid signaling	3	0.044207
	hsa04060	Cytokine-cytokine receptor interaction	4	0.045786
G2	hsa04110	Cell cycle	14	2.75E-19
	hsa04114	Oocyte meiosis	8	1.01E-08
	hsa04914	Progesterone-mediated oocyte maturation	6	2.91E-06
	hsa04115	p53 signaling pathway	4	0.000765
	hsa05203	Viral carcinogenesis	4	0.01777
	hsa05166	HTLV-I infection	4	0.031122
G3	hsa03008	Ribosome biogenesis in eukaryotes	9	1.05E-11
	hsa00240	Pyrimidine metabolism	4	0.001799
	hsa03020	RNA polymerase	3	0.00273
	hsa00230	Purine metabolism	4	0.008596

6.3.8 Analyzing DEGs in oxidative stress and apoptosis.

As mentioned in the previous section we found multiple genes in all three groups related to oxidative-stress and apopotosis. As a result we wanted to look into the genes from

the different CRC stages that are related to these pathways, in more detail. The dataset description for all the groups and the total count of DEGs found using the Limma procedure for all the datasets from Oncomine are described in Table 6.2. There are three Oncomine datasets in group-1, seven in group-2, and three in group-3. The total number of oxidative-stress related DEGs in Sabates, Skrzypczak, and Skrzypczak2 from group-1 were 29, 24, and 22, respectively. The apoptosis related DEGs were around 200 in these datasets. In group-2, the oxidative stress DEGs ranged from 19 to 39, and the apoptosis DEGs were around 200 in all seven Oncomine datasets (Table 6.2). The total oxidative stress DEGs in group-3 were 16, 28, and 30 for datasets Gradudens, Hong, and Skrzypczak2. The apoptosis-related DEGs in these datasets were 64, 180, and 188. Overall, more DEGs were found in the apoptosis pathway compared to the oxidative stress in all the groups. Among the DEGs in both of these pathways, there were more UR genes than DR in all three groups (Table 6.2). RRA algorithm was applied to integrate the list of DEGs from both oxidative stress and apoptosis separately in G1, G2, and G3. The robust integrated list of DEGs established for each group (G1, G2, and G3) related to oxidative stress and apoptosis is noted in Table 6.5. A P-value cut-off of 0.05 was assigned to filter the statistically significant robust DEGs. There is a relatively less number of robust Oxidative stress DEGs found when compared to apoptosis DEGs. More oxidative stress and apoptosis DEGs were found in colorectal adenocarcinoma (G2) than the other two groups.

Table 6.5: Robust DEGs related to oxidative stress and apoptosis genes in all groups from Oncomine.

Groups	RRA Oxidative-stress DEGs (UR/DR)	RRA Apoptosis DEGs (UR/DR)
G1	04 (02/02)	28 (19/09)
G2	09 (04/05)	47 (23/24)
G3	02 (00/02)	18 (09/09)

The details of Robust DEGs involved in the oxidative stress pathway are shown in Table 6.6. Some of the oxidative stress genes are common among these groups. PRDX6 is present in both G1 and G2. SEPP1 is common between G1 and G3. The G2 had the highest number of oxidative stress-related genes compared to the other two groups Table 6.6.

Table 6.6: Oxidative stress-related genes from Oncomine that are also found in RRA DEGs.

Groups	Common genes between RRA oxidative stress DEGs and RRA DEGs	Total
G1	GPX2 KIF9 PRDX6 SEPP1	04
G2	FOXM1 GSS NUDT1 PRDX2 ANGPTL7 MSRA PDLIM1 PRDX6 SCARA3	09
G3	CCL5 SEPP1	02

The number of Robust DEGs in the apoptosis pathway was more compared to oxidative stress. WDR74 gene is an apoptosis-related gene that was also identified among the top forty hub genes in G1. CHEK1, CDK1, CCNB1, MCM2, and CDC20 were the apoptosis-related genes in G2. These genes were also found within the forty hub genes of G2.

6.4 Discussion

Although the death rate of individuals from CRC has dropped over the past decade, CRC is still the third leading cause of mortality [Siegel et al., 2020]. CRC is the most common type of malignant tumor of the gastrointestinal tumors, with more than 1.5 million cases in the US [Siegel et al., 2020, Hong et al., 2014]. The total cases are expected to be increased by 2030, with 2.2 million new cases [Arnold et al., 2017]. The progression of CRC is a dynamic process with the expression levels of some molecules changing at different stages [Moroishi et al., 2015]. Because of its heterogeneity and complexity, early detection and diagnosis have become increasingly challenging. Effective biomarkers and

useful diagnostic approaches for early detection of CRC improve CRC patients' survival rate [Murphy et al., 2019, Lech et al., 2016]. Therefore, it is necessary to identify meaningful CRC biomarkers to better understand the molecular mechanism of CRC progression in different stages [Lech et al., 2016].

High-throughput technology, including microarray gene expression and next-generation sequencing, is widely used in cancer research [Lin and Tsai, 2016]. The microarray data can be used to identify hub genes and pathways related to CRC development [Xu et al., 2020, Solé et al., 2014, Zhao et al., 2019, Chen et al., 2019b]. However, these studies did not focus on different CRC subtypes, and most of the analysis was conducted on single CRC data [Solé et al., 2014, Zhao et al., 2019, Chen et al., 2019b]. These limitations highlight the requirement to conduct studies based on different CRC subtypes with multiple datasets to identify the hub genes and pathways in CRC.

In this study, we focused on expression profiles of normal vs. cases collected from several microarray studies and divided them into three groups. The G1 included colorectal adenoma vs. normal samples from GSE8671, GSE20916, and GSE89076. The G2 contained Colorectal adenocarcinoma vs. normal samples from GSE 20842, GSE20916, GSE39582, GSE110225, TCGA COAD, and TCGA READ. The G3 has Colorectal carcinoma vs. normal samples obtained from GSE3964, GSE113513, GSE32323, and GSE21510. An RRA approach was applied to integrate the DEGs from multiple datasets into a single list for each group. A total of 635 DEGs were identified in G1 with 186 UR and 449 DR. The G2 had 993 RRA DEGs with 499 UR and 494 DR. Finally, 285 RRA DEGs were present in G3, having 206 UR and 79 DR genes in them. To better understand these DEGs, we conducted GO enrichment analysis and KEGG pathway analysis for these RRA DEGs. We identified hub genes associated with CRC through computational techniques.

Studies have shown that the dysregulation of the cell cycle and mitotic nuclear cell division plays an important role in the occurrence and progression of CRC [Chen et al., 2019c, Wang et al., 2018, Cheng et al., 2018, Li et al., 2018b]. We observed that UR DEGs from G1 and G2 were enriched within the cell proliferation and mitotic nuclear division sub-

categories of the GO biological processes. It was also observed that the rRNA processing and cell division genes were highly enriched in G3 and G2, respectively. The UR DEGs in KEGG pathways showed that cell cycle, p53 signaling pathway, and microRNAs in cancer pathways are highly enriched in G2 compared to other groups. The enrichment of cellular response to zinc ion and bicarbonate transport pathways was found in all groups. The bicarbonate transport pathway plays a vital role in the diagnosis and treatment of many cancers, including CRC [Xu et al., 2020, Gorbatenko et al., 2014]. The KEGG analysis for DR DEGs showed that the mineral absorption pathway's enrichment was decreased from G1 to G3. The P13K-Akt signaling path-way, bile secretion, and gastric acid secretion were enriched only in G1. Chen et al. conducted in vitro experiments in CRC and showed that the cancer cell migration and invasion could be suppressed by epithelial mesenchymal transition (EMT) via P13K/Akt signaling [Chen et al., 2019a].

A PPI network provides a visual framework for a better insight into the functional organization of the proteins [Liu et al., 2009]. The RRA DEGs formed dense networks in each group. The network part that had the highest node interactions was considered and determined as hub genes using the net-work analyzer algorithm. The MCODE algorithm was used to obtain clusters that con-tained significant key genes. The results from the network analyzer when compared with MCODE results for each group, were found to be highly consistent. Overall, the top forty genes with the highest degree of interaction in each group were considered to be hub genes. These hub genes in each group were enriched in several critical path-ways related to cancer reported in Table 6.4.

The top KEGG pathways related to hub genes from G1 were ribosome biogenesis in eukaryotes, chemokine signaling receptor, pathways in cancer, P13K-Akt signaling pathway, cytokine-cytokine receptor interaction. The genes involved in the chemokine signaling pathway are CXCL12, CXCL3, and CCL21. CXCL3 was highly expressed in colorectal adenomas [Dai et al., 2020, Gong et al., 2020]. The FOXO1 gene was involved in cancer pathways. Agostini et al. compared the expression levels of FOXO1 in adenoma and carcinoma tissue and found that the expression levels were significantly higher in adenoma

compared to carcinoma [Agostini et al., 2008]. This result was consistent with our study, where FOXO1 is significantly expressed and is part of the top 40 hub genes in G1. However, it is not significantly expressed in G2 and G3, which are the cancerous stages.

In the adenocarcinoma (G2) datasets, the forty hub genes were mainly enriched in the cell cycle and p53 signaling pathway. Previous studies have shown that a few of the genes in cell-cycle-related pathways promote the proliferation of endothelial cells, contributing to tumor progression and metastasis in colorectal adenocarcinoma [Hong et al., 2009]. The genes involved in the cell cycle are CDK1, PLK1, TTK, CDC6, CCNA2, CDC20, CCNB1, CDC45, PTTG1, CHEK1, MCM4, BUB1, MCM2, and MAD2L1. The genes involved in the p53 pathway are CDK1, CCNB1, RRM2, and CHEK1. We searched the literature to find the association of these hub genes in colorectal adenocarcinoma. CDK1 gene promotes cell proliferation by inhibition of the FOXO1 transcription factor [Liu et al., 2008]. The effects of alteration of the CDK1 gene are seen in many cancer types, including esophageal adenocarcinoma, breast cancer, oral squamous cell carcinoma, hepatocellular, and pancreatic adenocarcinoma [Kim et al., 2008, Hansel et al., 2005, Wu et al., 2019, Piao et al., 2019, Chang et al., 2005]. Lu et al. argued that the RRM2 gene is overexpressed in colorectal adenocarcinoma [Lu et al., 2012]. The higher expression of RRM2 was correlated with the tumor node metastasis stage [Lu et al., 2012]. Gan et al. revealed that the expression of the CCNA2 gene is higher in colorectal adenocarcinoma tissues than in normal samples [Gan et al., 2018]. The study also revealed that knockdown of CCNA2 could suppress cell growth by disrupting the cell cycle and inducing cell apoptosis [Gan et al., 2018]. Takahashi et al. showed that PLK1 is overexpressed in CRC [Takahashi et al., 2003]. PLK1 is also involved in the proliferation, migration, and invasion of CRC cells [Han et al., 2012]. Gali-Muhtasib et al. proved overexpression of CHEK1 was correlated with advanced tumor stages and worse prognosis in CRC [Gali-Muhtasib et al., 2008]. We also searched the literature for other hub genes in G2 that were not part of the cell cycle and p53 pathways. The MAD2L1 and BUB1 are critical components of the spindle assembly [Xue et al., 2016] and are known important drivers of carcinogenesis in colorectal adeno-

carcinoma [Chen et al., 1998, Burum-Auensen et al., 2007]. Ding et al. showed cell growth suppression by knockdown of MAD2L1, which impaired cell cycle progression and induced cell apoptosis [Ding et al., 2020]. AURKA and TPX2 gene are also known for carcinogenesis in CRC by promoting the progression of colorectal adenoma to colorectal adenocarcinoma [Sillars-Hardebol et al., 2012]. Ren et al. revealed that the downregulation of PTTG1 suppressed cell proliferation and invasion in CRC [Ren and Jin, 2017]. DTL gene is known for its role in cell proliferation and cell cycle arrest in many cancers such as breast [Ueki et al., 2008], hepatocellular carcinoma [Pan et al., 2006], and gastric cancer [Missiaglia et al., 2009]. NUSAP1 plays an essential role in mitotic spindle assembly [Song and Rape, 2010]. It induced apoptosis and inhibited proliferation, migration, and invasion in CRC [Han et al., 2018]. KIF11 is another protein required for spindle formation [Zhu et al., 2005]. The knockdown of KIF11 prevents sphere formation indicating its importance in CRC [Imai et al., 2017]. TOP2A gene causes chromosomal instability in many different cancers [Simon et al., 2003, **?**, Bofin et al., 2003], and its protein expression level is mainly linked to advanced tumor stages and chemotherapeutic resistance via inhibition of apoptosis [Coss et al., 2009]. CEP55 enhances cell growth, and its knockdown inhibits cell growth in the breast and gastric [Tao et al., 2014, Wang et al., 2016].

The top 40 hub genes in G3 were involved in ribosome biogenesis in eukaryotes, rRNA processing, and pyrimidine and purine metabolism pathways. Previous studies have shown that the ribosome biogenesis pathway is altered in colorectal cancer. The alterations lead to increased production of ribosomes linked to the initiation and pro-gression of colorectal carcinogenesis [Slimane et al., 2020]. Stedman et al. showed that the dysfunction of ribosome biogenesis might also lead to p53-mediated apoptosis in some cancers [Stedman et al., 2015].

We further checked the presence of the oxidative stress and apoptosis pathway related genes in the Oncomine database. FOXM1, CDC25C, RRM2, CDK4, PRDX1, PRDX2, GPX2, GPX4, and FHL2 genes were the core genes found to be related to the oxidative stress pathway. The FOXM1 and RRM2 genes were also found in the cell cycle and p53

signaling pathway. Slattery et al. (2019) have suggested that the activation of p53 signaling may be related to cellular stress, which could influence apoptosis, cell cycle arrest and angiogenesis through the mRNA-miRNA interactions [Slattery et al., 2019]. The PRDX1 and PRDX2 regulate cellular signaling and differentiation. These genes were found to be UR in our analysis and may serve as a potential therapeutic target. Previous studies also showed that these genes were UR and promoted metastasis and angiogenesis in CRC [Li et al., 2018a, Lu et al., 2014]. Verset et al. showed that the higher expression of FHL2 was involved in the progression of CRC [Verset et al., 2013].

The genes linked to the apoptosis pathway were CHEK1, CDK1, CCNB1, GTSE1, MCM2, MCM10, CDC20, WDR74, and PMAIP1. As described in one of the previous sections, CHEK1, CDK1, and CCNB1 were involved in p53 signaling. Our analysis demonstrates that these are some of the major genes related to apoptosis pathways. The tumors lacking p53 have shown higher levels of expression of CHEK1, which was accompanied by inability to induce apoptosis [Gali-Muhtasib et al., 2008]. The higher expression of CDK1 and CDC20 in CRC patients is associated with a poorer prognosis [Li et al., 2020]. The GTSE1 promotes cell growth in breast cancer by activating the P13-Akt pathway and enhances metastasis. It could also regulate the p53 to alter the cell cycle [Lin et al., 2019]. PMAIP1 is also known as the NOXA gene. Its expression is regulated by the p53 signaling and has been involved in p53-mediated apoptosis [Shibue et al., 2003].

As future work, we will be validating the differential expression of the target hub genes in commercially available cell lines by qRT-PCR. We also plan to check the expression of these target transcripts during the different stages of CRC. The study of the genes in the key pathways related to tumorigenesis could lead to targets for novel therapeutic interventions.

6.5 Conclusion

In this study, we used multiple GEO datasets for different CRC types and found forty hub genes from each group through various bioinformatics approaches. The hub genes

were found to be enriched in the cell cycle, p53 signaling pathway, mineral absorption, and many other key pathways related to CRC progression. We also analyzed the genes involved in stress-survival and apoptosis-related pathways using Oncomine database. Our findings suggest that hub genes revealed in our study may be considered as biomarkers for the diagnosis of CRC. Additionally, the genes expressed in stress survival pathways could be tested as potential therapeutic targets. However, this study primarily being a in-silico analysis, has its limitations; in vivo and in vitro experiments would be needed to validate the biological functions of these genes in CRC.

Chapter 7

Conclusion and Future Research

7.1 Conclusion

Gene selection and classification in different high-dimensional biological data are challenging because of the large number of features present in the model. In this work, I have presented three algorithms for efficient gene selection and classification in biological data such as Microarray, DNA methylation, and RNA-seq. The algorithms were built such that they retrieve significantly expressed genes and achieve better classification accuracy. The performance of the algorithms was measured by comparing with the existing popular machine learning models on both synthetic and real data. The R codes for all the three algorithms are made publicly available [Patil, 2021]. The codes of interest can be easily incorporated by users in gene expression studies. These algorithms' development helps us identify significantly expressed genes in high dimensional data much efficiently. The significantly expressed genes can be further used for knowing their gene ontology and biological pathways information from various biological pathway databases. This work will also help in attaining other important information such as protein-protein interactions from given data that further helps to gain biological insights of the significant genes.

In the application part, the RLFS method developed earlier was also applied on RNA-seq datasets along with several microarray datasets related to CRC. The significant genes selected from the RLFS method were also present in the DEGs found through the traditional approach showing accurate gene selection accuracy of the RLFS method. The biological insights such as the expression of genes in different stages of cancer in several key pathways were gained through the functional enrichment analysis and protein- protein interaction networks developed using bioinformatics analysis. The genes revealed from the study would

be biomarkers and explain the CRC development and progression mechanisms.

7.2 Future Research

In this dissertation, three algorithms have been developed for accurate gene selection and improving classification performance in gene expression data. We also applied some of these algorithms on the colorectal cancer datasets to identify the hub genes. This section covers some of the future directions on application of machine learning tools on next-generation sequencing and third generation sequencing data.

Next-generation sequencing (NGS) technologies is growing rapidly producing a fast expanding collection of different NGS data types related to different diseases deposited in public databases. The Single-cell RNA sequencing (scRNA-seq), Assay for Transposase-Accessible Chromatin using sequencing (ATAC-seq), and Single-cell Assay for transposase-accessible chromatin using sequencing (scATAC-seq) are some of the NGS technologies widely used in biomedical research for whole-genome sequencing in cells and genome-wide chromatin accessibility, further increasing the size of the data generated. There are a limited amount of computational tools available to handle these data such as principal component analysis (PCA) and t-distributed stochastic neighbour embedding (tSNE) [Angarica and del Sol, 2017, Rostom et al., 2017, Tsompana and Buck, 2014, Qi et al., 2020, Yan et al., 2020]. Most of the existing tools need improvisation to attain good efficiency [Kiselev et al., 2019, Petegrosso et al., 2019]. There is an urgent need to develop machine learning models for prediction purposes in these types of data.

Further, the third-generation sequencing (TGS) technology is currently under active development. One of the commonly used TGS data is high-throughput chromosome conformation capture (Hi-C) data. The Hi-C data is used for genome-wide chromatin organization and is gaining popularity in the field of epigenetics. The big data challenges and size and complex structure interactions of these within the genomic data are very challenging to decipher for most of the existing computational tools [Pal et al., 2019]. There is a

need for newer computational approaches, including machine learning models, to handle these complex data. Future research in these directions will help address many complex biological problems.

In Chapter 6, we identified the hub genes related to different stages of colorectal cancer and also analyzed their enrichment in different biological pathways and networks. A major challenge for understanding cancer etiology is overcoming the complexity of gene-gene and gene-environment interactions contributing to carcinogenesis. As future research, the effects of variations in the regulatory and coding regions of selected genes can be considered. single nucleotide polymorphisms (SNPs) within the coding region can exert a direct effect on gene products and can alter protein expression levels, structure, and function. First, SNPs associated with key pathway genes can be identified using NCBI's SNP database (`https://www.ncbi.nlm.nih.gov/snp/`). A computational meta-analysis of genetic associations between key SNPs in the key genes and colorectal cancer risk can be performed based on available literature.

References

[Agapito, 2019] Agapito, G. (2019). Computer Tools to Analyze Microarray Data. In *Methods in Molecular Biology*.

[Agostini et al., 2008] Agostini, M., Enzo, M. V., Pucciarelli, S., Bedin, C., Pizzini, S., Urso, E. D. L., and Nitti, D. (2008). Forkhead box gene (FOXO1A) the emerging role in colon cancer. *Cancer Research*, 68(9 Supplement):4147 LP – 4147.

[Ahn et al., 2007] Ahn, H., Moon, H., Fazzari, M. J., Lim, N., Chen, J. J., and Kodell, R. L. (2007). Classification by ensembles from random partitions of high-dimensional data. *Computational Statistics and Data Analysis*.

[Algamal and Lee, 2015] Algamal, Z. Y. and Lee, M. H. (2015). Penalized logistic regression with the adaptive LASSO for gene selection in high-dimensional cancer classification. *Expert Systems with Applications*.

[Algamal and Lee, 2019] Algamal, Z. Y. and Lee, M. H. (2019). A two-stage sparse logistic regression for optimal gene selection in high-dimensional microarray data classification. *Advances in Data Analysis and Classification*.

[Alkuhlani et al., 2017] Alkuhlani, A., Nassef, M., and Farag, I. (2017). A comparative study of feature selection and classification techniques for high-throughput DNA methylation data. In *Advances in Intelligent Systems and Computing*.

[Almugren and Alshamlan, 2019] Almugren, N. and Alshamlan, H. (2019). A survey on hybrid feature selection methods in microarray gene expression data for cancer classification. *IEEE Access*, 7:78533–78548.

[Angarica and del Sol, 2017] Angarica, V. E. and del Sol, A. (2017). Bioinformatics tools for genome-wide epigenetic research.

[Aran et al., 2016] Aran, V., Victorino, A. P., Thuler, L. C., and Ferreira, C. G. (2016). Colorectal Cancer: Epidemiology, Disease Mechanisms and Interventions to Reduce Onset and Mortality.

[Arnold et al., 2017] Arnold, M., Sierra, M. S., Laversanne, M., Soerjomataram, I., Jemal, A., and Bray, F. (2017). Global patterns and trends in colorectal cancer incidence and mortality. *Gut*.

[Baker, 2010] Baker, M. (2010). Epigenome: mapping in motion. *Nature Methods*, 7(3):181–186.

[Battiti, 1994] Battiti, R. (1994). Using Mutual Information for Selecting Features in Supervised Neural Net Learning. *IEEE Transactions on Neural Networks*.

[Bibikova et al., 2009] Bibikova, M., Le, J., Barnes, B., Saedinia-Melnyk, S., Zhou, L., Shen, R., and Gunderson, K. L. (2009). Genome-wide DNA methylation profiling using Infinium® assay. *Epigenomics*.

[Bibikova et al., 2006] Bibikova, M., Lin, Z., Zhou, L., Chudin, E., Garcia, E. W., Wu, B., Doucet, D., Thomas, N. J., Wang, Y., Vollmer, E., Goldmann, T., Seifart, C., Jiang, W., Barker, D. L., Chee, M. S., Floros, J., and Fan, J. B. (2006). High-throughput DNA methylation profiling using universal bead arrays. *Genome Research*.

[Bielza et al., 2011] Bielza, C., Robles, V., and Larrañaga, P. (2011). Regularized logistic regression without a penalty term: An application to cancer classification with microarray data. *Expert Systems with Applications*.

[Bock, 2012] Bock, C. (2012). Analysing and interpreting DNA methylation data.

[Bofin et al., 2003] Bofin, A. M., Ytterhus, B., and Hagmar, B. M. (2003). TOP2A and HER-2 gene amplification in fine needle aspirates from breast carcinomas. *Cytopathology*.

[Bourgon et al., 2010a] Bourgon, R., Gentleman, R., and Huber, W. (2010a). Independent filtering increases detection power for high-throughput experiments. *Proceedings of the National Academy of Sciences of the United States of America*.

[Bourgon et al., 2010b] Bourgon, R., Gentleman, R., and Huber, W. (2010b). Reply to Talloen et al.: Independent filtering is a generic approach that needs domain specif c adaptation.

[Braun et al., 2009] Braun, P., Tasan, M., Dreze, M., Barrios-Rodiles, M., Lemmens, I., Yu, H., Sahalie, J. M., Murray, R. R., Roncari, L., de Smet, A. S., Venkatesan, K., Rual, J. F., Vandenhaute, J., Cusick, M. E., Pawson, T., Hill, D. E., Tavernier, J., Wrana, J. L., Roth, F. P., and Vidal, M. (2009). An experimentally derived confidence score for binary protein-protein interactions. *Nature Methods*, 6(1):91–97.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine Learning*.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*.

[Breuer et al., 2013] Breuer, K., Foroushani, A. K., Laird, M. R., Chen, C., Sribnaia, A., Lo, R., Winsor, G. L., Hancock, R. E. W., Brinkman, F. S. L., and Lynn, D. J. (2013). InnateDB: Systems biology of innate immunity and beyond - Recent updates and continuing curation. *Nucleic Acids Research*, 41(D1):D1228–D1233.

[Brody, 2015] Brody, H. (2015). Colorectal cancer.

[Burum-Auensen et al., 2007] Burum-Auensen, E., DeAngelis, P. M., Schjølberg, A. R., Røislien, J., Andersen, S. N., and Clausen, O. P. (2007). Spindle proteins Aurora A and BUB1B, but not Mad2, are aberrantly expressed in dysplastic mucosa of patients with longstanding ulcerative colitis. *Journal of Clinical Pathology*.

[Calderone et al., 2013] Calderone, A., Castagnoli, L., and Cesareni, G. (2013). Mentha: A resource for browsing integrated protein-interaction networks.

[Carbon et al., 2019] Carbon, S., Douglass, E., Dunn, N., Good, B., Harris, N. L., Lewis, S. E., Mungall, C. J., Basu, S., Chisholm, R. L., Dodson, R. J., Hartline, E., Fey, P., Thomas, P. D., Albou, L. P., Ebert, D., Kesling, M. J., Mi, H., Muruganujan, A., Huang, X., Poudel, S., Mushayahama, T., Hu, J. C., LaBonte, S. A., Siegele, D. A., Antonazzo, G., Attrill, H., Brown, N. H., Fexova, S., Garapati, P., Jones, T. E. M., Marygold, S. J., Millburn, G. H., Rey, A. J., Trovisco, V., Santos, G. D., Emmert, D. B., Falls, K., Zhou, P., Goodman, J. L., Strelets, V. B., Thurmond, J., Courtot, M., Osumi, D. S., Parkinson, H., Roncaglia, P., Acencio, M. L., Kuiper, M., Lreid, A., Logie, C., Lovering, R. C., Huntley, R. P., Denny, P., Campbell, N. H., Kramarz, B., Acquaah, V., Ahmad, S. H., Chen, H., Rawson, J. H., Chibucos, M. C., Giglio, M., Nadendla, S., Tauber, R., Duesbury, M. J., Del, N. T., Meldal, B. H. M., Perfetto, L., Porras, P., Orchard, S., Shrivastava, A., Xie, Z., Chang, H. Y., Finn, R. D., Mitchell, A. L., Rawlings, N. D., Richardson, L., Sangrador-Vegas, A., Blake, J. A., Christie, K. R., Dolan, M. E., Drabkin, H. J., Hill, D. P., Ni, L., Sitnikov, D., Harris, M. A., Oliver, S. G., Rutherford, K., Wood, V., Hayles, J., Bahler, J., Lock, A., Bolton, E. R., Pons, J. D., Dwinell, M., Hayman, G. T., Laulederkind, S. J. F., Shimoyama, M., Tutaj, M., Wang, S. J., D'Eustachio, P., Matthews, L., Balhoff, J. P., Aleksander, S. A., Binkley, G., Dunn, B. L., Cherry, J. M., Engel, S. R., Gondwe, F., Karra, K., MacPherson, K. A., Miyasato, S. R., Nash, R. S., Ng, P. C., Sheppard, T. K., Vp, A. S., Simison, M., Skrzypek, M. S., Weng, S., Wong, E. D., Feuermann, M., Gaudet, P., Bakker, E., Berardini, T. Z., Reiser, L., Subramaniam, S., Huala, E., Arighi, C., Auchincloss, A., Axelsen, K., Argoud, G. P., Bateman, A., Bely, B., Blatter, M. C., Boutet, E., Breuza, L., Bridge, A., Britto, R., Bye-A-Jee, H., Casals-Casas, C., Coudert, E., Estreicher, A., Famiglietti, L., Garmiri, P., Georghiou, G., Gos, A., Gruaz-Gumowski, N., Hatton-Ellis, E., Hinz, U., Hulo, C., Ignatchenko, A., Jungo, F., Keller, G., Laiho, K., Lemercier, P., Lieberherr, D., Lussi, Y., Mac-Dougall, A., Magrane, M., Martin, M. J., Masson, P., Natale, D. A., Hyka, N. N., Pedruzzi, I., Pichler, K., Poux, S., Rivoire, C., Rodriguez-Lopez, M., Sawford, T., Speretta, E., Shypitsyna, A., Stutz, A., Sundaram, S., Tognolli, M., Tyagi, N., Warner,

K., Zaru, R., Wu, C., Chan, J., Cho, J., Gao, S., Grove, C., Harrison, M. C., Howe, K., Lee, R., Mendel, J., Muller, H. M., Raciti, D., Auken, K. V., Berriman, M., Stein, L., Sternberg, P. W., Howe, D., Toro, S., and Westerfield, M. (2019). The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Research*, 47(D1):D330–D338.

[Celli et al., 2018] Celli, F., Cumbo, F., and Weitschek, E. (2018). Classification of Large DNA Methylation Datasets for Identifying Cancer Drivers. *Big Data Research*.

[Chandramohan et al., 2013] Chandramohan, R., Wu, P. Y., Phan, J. H., and Wang, M. D. (2013). Benchmarking RNA-Seq quantification tools. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*.

[Chandrashekar and Sahin, 2014] Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers and Electrical Engineering*.

[Chang et al., 2005] Chang, J. T., Wang, H. M., Chang, K. W., Chen, W. H., Wen, M. C., Hsu, Y. M., Yung, B. Y. M., Chen, I. H., Liao, C. T., Hsieh, L. L., and Cheng, A. J. (2005). Identification of differentially expressed genes in oral squamous cell carcinoma (OSCC): Overexpression of NPM, CDK1 and NDRG1 and underexpression of CHES1. *International Journal of Cancer*.

[Chang et al., 2013] Chang, K., Creighton, C. J., Davis, C., Donehower, L., Drummond, J., Wheeler, D., Ally, A., Balasundaram, M., Birol, I., Butterfield, Y. S. N., Chu, A., Chuah, E., Chun, H.-J. E., Dhalla, N., Guin, R., Hirst, M., Hirst, C., Holt, R. A., Jones, S. J. M., Lee, D., Li, H. I., Marra, M. A., Mayo, M., Moore, R. A., Mungall, A. J., Robertson, A. G., Schein, J. E., Sipahimalani, P., Tam, A., Thiessen, N., Varhol, R. J., Beroukhim, R., Bhatt, A. S., Brooks, A. N., Cherniack, A. D., Freeman, S. S., Gabriel, S. B., Helman, E., Jung, J., Meyerson, M., Ojesina, A. I., Pedamallu, C. S., Saksena, G., Schumacher, S. E., Tabak, B., Zack, T., Lander, E. S., Bristow, C. A., Hadjipanayis, A., Haseley, P., Kucherlapati, R., Lee, S., Lee, E., Luquette, L. J., Mahadeshwar, H. S., Pantazi, A., Parfenov, M., Park, P. J., Protopopov, A., Ren, X., Santoso, N., Seidman,

J., Seth, S., Song, X., Tang, J., Xi, R., Xu, A. W., Yang, L., Zeng, D., Auman, J. T., Balu, S., Buda, E., Fan, C., Hoadley, K. A., Jones, C. D., Meng, S., Mieczkowski, P. A., Parker, J. S., Perou, C. M., Roach, J., Shi, Y., Silva, G. O., Tan, D., Veluvolu, U., Waring, S., Wilkerson, M. D., Wu, J., Zhao, W., Bodenheimer, T., Hayes, D. N., Hoyle, A. P., Jeffreys, S. R., Mose, L. E., Simons, J. V., Soloway, M. G., Baylin, S. B., Berman, B. P., Bootwalla, M. S., Danilova, L., Herman, J. G., Hinoue, T., Laird, P. W., Rhie, S. K., Shen, H., Triche, T., Weisenberger, D. J., Carter, S. L., Cibulskis, K., Chin, L., Zhang, J., Getz, G., Sougnez, C., Wang, M., Saksena, G., Carter, S. L., Cibulskis, K., Chin, L., Zhang, J., Getz, G., Dinh, H., Doddapaneni, H. V., Gibbs, R., Gunaratne, P., Han, Y., Kalra, D., Kovar, C., Lewis, L., Morgan, M., Morton, D., Muzny, D., Reid, J., Xi, L., Cho, J., DiCara, D., Frazer, S., Gehlenborg, N., Heiman, D. I., Kim, J., Lawrence, M. S., Lin, P., Liu, Y., Noble, M. S., Stojanov, P., Voet, D., Zhang, H., Zou, L., Stewart, C., Bernard, B., Bressler, R., Eakin, A., Iype, L., Knijnenburg, T., Kramer, R., Kreisberg, R., Leinonen, K., Lin, J., Liu, Y., Miller, M., Reynolds, S. M., Rovira, H., Shmulevich, I., Thorsson, V., Yang, D., Zhang, W., Amin, S., Wu, C.-J., Wu, C.-C., Akbani, R., Aldape, K., Baggerly, K. A., Broom, B., Casasent, T. D., Cleland, J., Creighton, C., Dodda, D., Edgerton, M., Han, L., Herbrich, S. M., Ju, Z., Kim, H., Lerner, S., Li, J., Liang, H., Liu, W., Lorenzi, P. L., Lu, Y., Melott, J., Mills, G. B., Nguyen, L., Su, X., Verhaak, R., Wang, W., Weinstein, J. N., Wong, A., Yang, Y., Yao, J., Yao, R., Yoshihara, K., Yuan, Y., Yung, A. K., Zhang, N., Zheng, S., Ryan, M., Kane, D. W., Aksoy, B. A., Ciriello, G., Dresdner, G., Gao, J., Gross, B., Jacobsen, A., Kahles, A., Ladanyi, M., Lee, W., Lehmann, K.-V., Miller, M. L., Ramirez, R., Rätsch, G., Reva, B., Sander, C., Schultz, N., Senbabaoglu, Y., Shen, R., Sinha, R., Sumer, S. O., Sun, Y., Taylor, B. S., Weinhold, N., Fei, S., Spellman, P., Benz, C., Carlin, D., Cline, M., Craft, B., Ellrott, K., Goldman, M., Haussler, D., Ma, S., Ng, S., Paull, E., Radenbaugh, A., Salama, S., Sokolov, A., Stuart, J. M., Swatloski, T., Uzunangelov, V., Waltman, P., Yau, C., Zhu, J., Hamilton, S. R., Getz, G., Sougnez, C., Abbott, S., Abbott, R., Dees, N. D., Delehaunty, K., Ding, L., Dooling, D. J., Eldred, J. M., Fronick,

C. C., Fulton, R., Fulton, L. L., Kalicki-Veizer, J., Kanchi, K.-L., Kandoth, C., Koboldt, D. C., Larson, D. E., Ley, T. J., Lin, L., Lu, C., Magrini, V. J., Mardis, E. R., McLellan, M. D., McMichael, J. F., Miller, C. A., O'Laughlin, M., Pohl, C., Schmidt, H., Smith, S. M., Walker, J., Wallis, J. W., Wendl, M. C., Wilson, R. K., Wylie, T., Zhang, Q., Burton, R., Jensen, M. A., Kahn, A., Pihl, T., Pot, D., Wan, Y., Levine, D. A., Black, A. D., Bowen, J., Network, T. C. G. A. R., Center, G. C., Center, G. D. A., Center, S., Center, D. C., Site, T. S., and Center, B. C. R. (2013). The Cancer Genome Atlas Pan-Cancer analysis project. *Nature Genetics*, 45(10):1113–1120.

[Chatterjee et al., 2018] Chatterjee, A., Ahn, A., Rodger, E. J., Stockwell, P. A., and Eccles, M. R. (2018). A guide for designing and analyzing RNA-seq data. In *Methods in Molecular Biology*.

[Chen et al., 2019a] Chen, H., Liu, Y., Jiang, C. J., Chen, Y. M., Li, H., and Liu, Q. A. (2019a). Calcium-Activated Chloride Channel A4 (CLCA4) Plays Inhibitory Roles in Invasion and Migration Through Suppressing Epithelial-Mesenchymal Transition via PI3K/AKT Signaling in Colorectal Cancer. *Medical Science Monitor*.

[Chen et al., 2019b] Chen, J., Wang, Z., Shen, X., Cui, X., and Guo, Y. (2019b). Identification of novel biomarkers and small molecule drugs in human colorectal cancer by microarray and bioinformatics analysis. *Molecular Genetics and Genomic Medicine*.

[Chen et al., 1998] Chen, R. H., Shevchenko, A., Mann, M., and Murray, A. W. (1998). Spindle checkpoint protein Xmad1 recruits Xmad2 to unattached kinetochores. *Journal of Cell Biology*.

[Chen and Ishwaran, 2012] Chen, X. and Ishwaran, H. (2012). Random forests for genomic data analysis.

[Chen et al., 2019c] Chen, Z., Lin, Y., Gao, J., Lin, S., Zheng, Y., Liu, Y., and Chen, S. Q. (2019c). Identification of key candidate genes for colorectal cancer by bioinformatics analysis. *Oncology Letters*.

[Cheng et al., 2018] Cheng, J., Dwyer, M., Okolotowicz, K. J., Mercola, M., and Cashman, J. R. (2018). A novel inhibitor targets both WNT signaling and ATM/P53 in colorectal cancer. *Cancer Research*.

[Choy et al., 2010] Choy, M. K., Movassagh, M., Goh, H. G., Bennett, M. R., Down, T. A., and Foo, R. S. (2010). Genome-wide conserved consensus transcription factor binding motifs are hyper-methylated. *BMC Genomics*.

[Coss et al., 2009] Coss, A., Tosetto, M., Fox, E. J., Sapetto-Rebow, B., Gorman, S., Kennedy, B. N., Lloyd, A. T., Hyland, J. M., O'Donoghue, D. P., Sheahan, K., Leahy, D. T., Mulcahy, H. E., and O'Sullivan, J. N. (2009). Increased topoisomerase IIα expression in colorectal cancer is associated with advanced disease and chemotherapeutic resistance via inhibition of apoptosis. *Cancer Letters*.

[Cusick et al., 2009] Cusick, M. E., Yu, H., Smolyar, A., Venkatesan, K., Carvunis, A. R., Simonis, N., Rual, J. F., Borick, H., Braun, P., Dreze, M., Vandenhaute, J., Galli, M., Yazaki, J., Hill, D. E., Ecker, J. R., Roth, F. P., and Vidal, M. (2009). Literature-curated protein interaction datasets. *Nature Methods*, 6(1):39–46.

[Dai et al., 2020] Dai, G. P., Wang, L. P., Wen, Y. Q., Ren, X. Q., and Zuo, S. G. (2020). Identification of key genes for predicting colorectal cancer prognosis by integrated bioinformatics analysis. *Oncology Letters*.

[Dash, 2020] Dash, R. (2020). A two stage grading approach for feature selection and classification of microarray data using Pareto based feature ranking techniques: A case study. *Journal of King Saud University - Computer and Information Sciences*.

[Datta et al., 2007] Datta, S., Datta, S., Parrish, R. S., and Thompson, C. M. (2007). Microarray data analysis. In *Computational Methods in Biomedical Research*.

[Datta et al., 2010] Datta, S., Pihur, V., and Datta, S. (2010). An adaptive optimal ensemble classifier via bagging and rank aggregation with applications to high dimensional data. *BMC Bioinformatics*.

[Dayeh et al., 2014] Dayeh, T., Volkov, P., Salö, S., Hall, E., Nilsson, E., Olsson, A. H., Kirkpatrick, C. L., Wollheim, C. B., Eliasson, L., Rönn, T., Bacos, K., and Ling, C. (2014). Genome-Wide DNA Methylation Analysis of Human Pancreatic Islets from Type 2 Diabetic and Non-Diabetic Donors Identifies Candidate Genes That Influence Insulin Secretion. *PLoS Genetics*.

[Deshiere et al., 2019] Deshiere, A., Berthet, N., Lecouturier, F., Gaudaire, D., and Hans, A. (2019). Molecular characterization of Equine Infectious Anemia Viruses using targeted sequence enrichment and next generation sequencing. *Virology*.

[Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

[Ding et al., 2020] Ding, X., Duan, H., and Luo, H. (2020). Identification of Core Gene Expression Signature and Key Pathways in Colorectal Cancer. *Frontiers in Genetics*.

[Ditzler et al., 2015] Ditzler, G., Morrison, J. C., Lan, Y., and Rosen, G. L. (2015). Fizzy: Feature subset selection for metagenomics. *BMC Bioinformatics*.

[Dong et al., 2007] Dong, H. K., Jeung, H. C., Chan, H. P., Seung, H. K., Gui, Y. L., Won, S. L., Nam, K. K., Hyun, C. C., and Sun, Y. R. (2007). Whole genome analysis for liver metastasis gene signatures in colorectal cancer. *International Journal of Cancer*.

[Dong and Provart, 2018] Dong, S. and Provart, N. J. (2018). *Analyses of Protein Interaction Networks Using Computational Tools*, pages 97–117. Springer New York, New York, NY.

[Duda et al., 1998] Duda, R. O., Hart, P. E., and Stork, D. G. (1998). Pattern Classification (2nd ed .). *Computational Complexity*.

[E.I. et al., 2019] E.I., V., E., P., O., P., D., K., S., W., S., K., C., C., L., P., C., S., A., P., V., G., A., D.-S., and A., C. (2019). Radiogenomic Analysis of F-18-Fluorodeoxyglucose Positron Emission Tomography and Gene Expression Data Elucidates the Epidemiological Complexity of Colorectal Cancer Landscape. *Computational and Structural Biotechnology Journal*.

[Elyasigomari et al., 2017] Elyasigomari, V., Lee, D. A., Screen, H. R., and Shaheed, M. H. (2017). Development of a two-stage gene selection method that incorporates a novel hybrid approach using the cuckoo optimization algorithm and harmony search for cancer classification. *Journal of Biomedical Informatics*.

[Esteller, 2011] Esteller, M. (2011). Non-coding RNAs in human disease.

[Fabregat et al., 2016] Fabregat, A., Sidiropoulos, K., Garapati, P., Gillespie, M., Hausmann, K., Haw, R., Jassal, B., Jupe, S., Korninger, F., McKay, S., Matthews, L., May, B., Milacic, M., Rothfels, K., Shamovsky, V., Webber, M., Weiser, J., Williams, M., Wu, G., Stein, L., Hermjakob, H., and D'Eustachio, P. (2016). The reactome pathway knowledgebase. *Nucleic Acids Research*, 44(D1):D481–D487.

[Fan and Li, 2001] Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*.

[Fan and Lv, 2008] Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*.

[Ferreira and Figueiredo, 2012] Ferreira, A. J. and Figueiredo, M. A. (2012). Efficient feature selection filters for high-dimensional data. *Pattern Recognition Letters*.

[Fielden and Zacharewski, 2001] Fielden, M. R. and Zacharewski, T. R. (2001). Challenges and limitations of gene expression profiling in mechanistic and predictive toxicology. *Toxicological Sciences*.

[Fleuret, 2004] Fleuret, F. (2004). Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*.

[Freund, 2001] Freund, Y. (2001). An adaptive version of the boost by majority algorithm. *Machine Learning*.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*.

[Friedman et al., 1997] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning*.

[Gaedcke et al., 2010] Gaedcke, J., Grade, M., Jung, K., Camps, J., Jo, P., Emons, G., Gehoff, A., Sax, U., Schirmer, M., Becker, H., Beissbarth, T., Ried, T., and Ghadimi, B. M. (2010). Mutated KRAS results in overexpression of DUSP4, a MAP-kinase phosphatase, and SMYD3, a histone methyltransferase, in rectal carcinomas. *Genes Chromosomes and Cancer*.

[Gali-Muhtasib et al., 2008] Gali-Muhtasib, H., Kuester, D., Mawrin, C., Bajbouj, K., Diestel, A., Ocker, M., Habold, C., Foltzer-Jourdainne, C., Schoenfeld, P., Peters, B., Diab-Assaf, M., Pommrich, U., Itani, W., Lippert, H., Roessner, A., and Schneider-Stock, R. (2008). Thymoquinone triggers inactivation of the stress response pathway sensor CHEK1 and contributes to apoptosis in colorectal cancer cells. *Cancer Research*.

[Gan et al., 2018] Gan, Y., Li, Y., Li, T., Shu, G., and Yin, G. (2018). CCNA2 acts as a novel biomarker in regulating the growth and apoptosis of colorectal cancer. *Cancer Management and Research*.

[Giot et al., 2003] Giot, L., Bader, J. S., Brouwer, C., Chaudhuri, A., Kuang, B., Li, Y., Hao, Y. L., Ooi, C. E., Godwin, B., Vitols, E., Vijayadamodar, G., Pochart, P., Machineni, H., Welsh, M., Kong, Y., Zerhusen, B., Malcolm, R., Varrone, Z., Collis, A., Minto, M., Burgess, S., McDaniel, L., Stimpson, E., Spriggs, F., Williams, J., Neurath, K., Ioime, N., Agee, M., Voss, E., Furtak, K., Renzulli, R., Aanensen, N., Carrolla, S., Bickelhaupt, E., Lazovatsky, Y., DaSilva, A., Zhong, J., Stanyon, C. A., Finley, R. L., White, K. P., Braverman, M., Jarvie, T., Gold, S., Leach, M., Knight, J., Shimkets, R. A., McKenna, M. P., Chant, J., and Rothberg, J. M. (2003). A protein interaction map of drosophila melanogaster. *Science*, 302:1727–1736.

[Gong et al., 2020] Gong, B., Kao, Y., Zhang, C., Sun, F., Gong, Z., and Chen, J. (2020). Identification of Hub Genes Related to Carcinogenesis and Prognosis in Colorectal Cancer Based on Integrated Bioinformatics. *Mediators of Inflammation*.

[Gorbatenko et al., 2014] Gorbatenko, A., Olesen, C. W., Boedtkjer, E., and Pedersen, S. F. (2014). Regulation and roles of bicarbonate transporters in cancer. *Frontiers in Physiology*.

[Graudens et al., 2006] Graudens, E., Boulanger, V., Mollard, C., Mariage-Samson, R., Barlet, X., Grémy, G., Couillault, C., Lajémi, M., Piatier-Tonneau, D., Zaborski, P., Eveno, E., Auffray, C., and Imbeaud, S. (2006). Deciphering cellular states of innate tumor drug responses.

[Guo et al., 2017] Guo, Y., Bao, Y., Ma, M., and Yang, W. (2017). Identification of key candidate genes and pathways in colorectal cancer by integrated bioinformatical analysis. *International Journal of Molecular Sciences*.

[Guyon et al., 2002] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*.

[Han et al., 2012] Han, D. P., Zhu, Q. L., Cui, J. T., Wang, P. X., Qu, S., Cao, Q. F., Zong, Y. P., Feng, B., Zheng, M. H., and Lu, A. G. (2012). Polo-like kinase 1 is overexpressed

in colorectal cancer and participates in the migration and invasion of colorectal cancer cells. *Medical Science Monitor*.

[Han et al., 2018] Han, G., Wei, Z., Cui, H., Zhang, W., Wei, X., Lu, Z., and Bai, X. (2018). NUSAP1 gene silencing inhibits cell proliferation, migration and invasion through inhibiting DNMT1 gene expression in human colorectal cancer. *Experimental Cell Research*.

[Hansel et al., 2005] Hansel, D. E., Dhara, S., Huang, R. C. C., Ashfaq, R., Deasel, M., Shimada, Y., Bernstein, H. S., Harmon, J., Brock, M., Forastiere, A., Washington, M. K., Maitra, A., and Montgomery, E. (2005). CDC2/CDK1 expression in esophageal adenocarcinoma and precursor lesions serves as a diagnostic and cancer progression marker and potential novel drug target. *American Journal of Surgical Pathology*.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

[Hayer et al., 2015] Hayer, K. E., Pizarro, A., Lahens, N. F., Hogenesch, J. B., and Grant, G. R. (2015). Benchmark analysis of algorithms for determining and quantifying full-length mRNA splice forms from RNA-seq data. *Bioinformatics*.

[Hearst et al., 1998] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28.

[Hira and Gillies, 2015] Hira, Z. M. and Gillies, D. F. (2015). A review of feature selection and feature extraction methods applied on microarray data. *Advances in Bioinformatics*.

[Holliday and Pugh, 1975] Holliday, R. and Pugh, J. E. (1975). DNA modification mechanisms and gene activity during development. *Science*.

[Hong et al., 2009] Hong, B. S., Cho, J. H., Kim, H., Choi, E. J., Rho, S., Kim, J., Kim, J. H., Choi, D. S., Kim, Y. K., Hwang, D., and Gho, Y. S. (2009). Colorectal cancer cell-

derived microvesicles are enriched in cell cycle-related mRNAs that promote proliferation of endothelial cells. *BMC Genomics*.

[Hong et al., 2014] Hong, Y., Won, J., Lee, Y., Lee, S., Park, K., Chang, K. T., and Hong, Y. (2014). Melatonin treatment induces interplay of apoptosis, autophagy, and senescence in human colorectal cancer cells. *Journal of Pineal Research*.

[Honrado et al., 2006] Honrado, E., Osorio, A., Palacios, J., and Benitez, J. (2006). Pathology and gene expression of hereditary breast tumors associated with BRCA1, BRCA2 and CHEK2 gene mutations.

[Hrdlickova et al., 2017] Hrdlickova, R., Toloue, M., and Tian, B. (2017). RNA-Seq methods for transcriptome analysis.

[Hu et al., 2020] Hu, X., Bao, M., Huang, J., Zhou, L., and Zheng, S. (2020). Identification and Validation of Novel Biomarkers for Diagnosis and Prognosis of Hepatocellular Carcinoma. *Frontiers in Oncology*, 10.

[Huang et al., 2009a] Huang, D. W., Sherman, B. T., and Lempicki, R. A. (2009a). Bioinformatics enrichment tools: Paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Research*, 37(1):1–13.

[Huang et al., 2009b] Huang, D. W., Sherman, B. T., and Lempicki, R. A. (2009b). Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nature Protocols*, 4(1):44–57.

[Huerta and Burke, 2016] Huerta, L. and Burke, M. (2016). Functional genomics (II): Common technologies and data analysis methods.

[Hung and Weng, 2017] Hung, J. H. and Weng, Z. (2017). Analysis of microarray and RNA-seq expression profiling data. *Cold Spring Harbor Protocols*.

[Huttlin et al., 2015] Huttlin, E. L., Ting, L., Bruckner, R. J., Gebreab, F., Gygi, M. P., Szpyt, J., Tam, S., Zarraga, G., Colby, G., Baltier, K., Dong, R., Guarani, V., Vaites, L. P., Ordureau, A., Rad, R., Erickson, B. K., Wühr, M., Chick, J., Zhai, B., Kolippakkam, D., Mintseris, J., Obar, R. A., Harris, T., Artavanis-Tsakonas, S., Sowa, M. E., Camilli, P. D., Paulo, J. A., Harper, J. W., and Gygi, S. P. (2015). The BioPlex Network: A Systematic Exploration of the Human Interactome. *Cell*, 162(2):425–440.

[Iguyon and Elisseeff, 2003] Iguyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection.

[Imai et al., 2017] Imai, T., Oue, N., Sentani, K., Sakamoto, N., Uraoka, N., Egi, H., Hinoi, T., Ohdan, H., Yoshida, K., and Yasui, W. (2017). KIF11 is required for spheroid formation by oesophageal and colorectal cancer cells. *Anticancer Research*.

[Ito et al., 2002] Ito, T., Ota, K., Kubota, H., Yamaguchi, Y., Chiba, T., Sakuraba, K., and Yoshida, M. (2002). Roles for the two-hybrid system in exploration of the yeast protein interactome.

[Jones, 2012] Jones, P. A. (2012). Functions of DNA methylation: Islands, start sites, gene bodies and beyond.

[Jović et al., 2015] Jović, A., Brkić, K., and Bogunović, N. (2015). A review of feature selection methods with applications. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 - Proceedings*.

[Kanehisa et al., 2012] Kanehisa, M., Goto, S., Sato, Y., Furumichi, M., and Tanabe, M. (2012). KEGG for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Research*, 40(D1):D109–D114.

[Khamas et al., 2012] Khamas, A., Ishikawa, T., Shimokawa, K., Mogushi, K., Iida, S., Ishiguro, M., Mizushima, H., Tanaka, H., Uetake, H., and Sugihara, K. (2012). Screen-

ing for epigenetically masked genes in colorectal cancer using 5-aza-2-deoxycytidine, microarray and gene expression profile. *Cancer Genomics and Proteomics*.

[Khatri et al., 2012] Khatri, P., Sirota, M., and Butte, A. J. (2012). Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges. *PLoS Computational Biology*, 8(2):e1002375.

[Kim et al., 2010] Kim, M., Long, T. I., Arakawa, K., Wang, R., Yu, M. C., and Laird, P. W. (2010). DNA methylation as a biomarker for cardiovascular disease risk. *PLoS ONE*.

[Kim and Halabi, 2016] Kim, S. and Halabi, S. (2016). High Dimensional Variable Selection with Error Control. *BioMed Research International*.

[Kim and Kim, 2019] Kim, S. and Kim, J. M. (2019). Two-stage classification with SIS using a new filter ranking method in high throughput data. *Mathematics*.

[Kim et al., 2008] Kim, S. J., Nakayama, S., Miyoshi, Y., Taguchi, T., Tamaki, Y., Matsushima, T., Torikoshi, Y., Tanaka, S., Yoshida, T., Ishihara, H., and Noguchi, S. (2008). Determination of the specific activity of CDK1 and CDK2 as a novel prognostic indicator for early breast cancer. *Annals of oncology : official journal of the European Society for Medical Oncology / ESMO*.

[Kiselev et al., 2019] Kiselev, V. Y., Andrews, T. S., and Hemberg, M. (2019). Challenges in unsupervised clustering of single-cell RNA-seq data.

[Ko et al., 2013] Ko, Y. A., Mohtat, D., Suzuki, M., Park, A. S., Izquierdo, M. C., Han, S. Y., Kang, H. M., Si, H., Hostetter, T., Pullman, J. M., Fazzari, M., Verma, A., Zheng, D., Greally, J. M., and Susztak, K. (2013). Cytosine methylation changes in enhancer regions of core pro-fibrotic genes characterize kidney fibrosis development. *Genome Biology*.

[Kobayashi et al., 2011] Kobayashi, Y., Absher, D. M., Gulzar, Z. G., Young, S. R., McKenney, J. K., Peehl, D. M., Brooks, J. D., Myers, R. M., and Sherlock, G. (2011). DNA methylation profiling reveals novel biomarkers and important roles for DNA methyltransferases in prostate cancer. *Genome Research*.

[Kolde et al., 2012] Kolde, R., Laur, S., Adler, P., and Vilo, J. (2012). Robust rank aggregation for gene list integration and meta-analysis. *Bioinformatics*.

[Kononenko, 1994] Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

[Korostin et al., 2020] Korostin, D., Kulemin, N., Naumov, V., Belova, V., Kwon, D., and Gorbachev, A. (2020). Comparative analysis of novel MGISEQ-2000 sequencing platform vs Illumina HiSeq 2500 for whole-genome sequencing. *PLoS ONE*.

[Kukurba and Montgomery, 2015] Kukurba, K. R. and Montgomery, S. B. (2015). RNA sequencing and analysis. *Cold Spring Harbor Protocols*.

[Kumar et al., 2015] Kumar, M., Rath, N. K., Swain, A., and Rath, S. K. (2015). Feature Selection and Classification of Microarray Data using MapReduce based ANOVA and K-Nearest Neighbor. In *Procedia Computer Science*.

[Kursa and Rudnicki, 2010] Kursa, M. B. and Rudnicki, W. R. (2010). Feature selection with the boruta package. *Journal of Statistical Software*.

[Köcher and Superti-Furga, 2007] Köcher, T. and Superti-Furga, G. (2007). Mass spectrometry-based functional proteomics: From molecular machines to protein networks.

[Lech et al., 2016] Lech, G., Słotwiński, R., Słodkowski, M., and Krasnodębski, I. W. (2016). Colorectal cancer tumour markers and biomarkers: Recent therapeutic advances. *World Journal of Gastroenterology*.

[Li and Li, 2010] Li, C. and Li, H. (2010). Variable selection and regression analysis for graph-structured covariates with an application to genomics. *Annals of Applied Statistics*.

[Li et al., 2018a] Li, H. X., Sun, X. Y., Yang, S. M., Wang, Q., and Wang, Z. Y. (2018a). Peroxiredoxin 1 promoted tumor metastasis and angiogenesis in colorectal cancer. *Pathology Research and Practice*.

[Li et al., 2017a] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2017a). Feature selection: A data perspective.

[Li et al., 2017b] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2017b). Feature selection: A data perspective.

[Li et al., 2018b] Li, J., Liu, Y. Y., Yang, X. F., Shen, D. F., Sun, H. Z., Huang, K. Q., and Zheng, H. C. (2018b). Effects and mechanism of STAT3 silencing on the growth and apoptosis of colorectal cancer cells. *Oncology Letters*.

[Li et al., 2020] Li, J., Wang, Y., Wang, X., and Yang, Q. (2020). CDK1 and CDC20 overexpression in patients with colorectal cancer are associated with poor prognosis: Evidence from integrated bioinformatics analysis. *World Journal of Surgical Oncology*.

[Li and Ilie, 2017] Li, Y. and Ilie, L. (2017). SPRINT: Ultrafast protein-protein interaction prediction of the entire human interactome. *BMC Bioinformatics*, 18(1):485.

[Li et al., 2010] Li, Y., Zhang, Y., and Zhao, S. (2010). Gender classification with support vector machines based on non-tensor pre-wavelets. In *2nd International Conference on Computer Research and Development, ICCRD 2010*.

[Liao and Chin, 2007] Liao, J. G. and Chin, K. V. (2007). Logistic regression for disease classification using microarray data: Model selection in a large p and small n case. *Bioinformatics*.

[Lin and Tsai, 2016] Lin, E. and Tsai, S. J. (2016). Genome-wide microarray analysis of gene expression profiling in major depression and antidepressant therapy.

[Lin et al., 2019] Lin, F., Xie, Y. J., Zhang, X. K., Huang, T. J., Xu, H. F., Mei, Y., Liang, H., Hu, H., Lin, S. T., Luo, F. F., Lang, Y. H., Peng, L. X., Qian, C. N., and Huang, B. J. (2019). GTSE1 is involved in breast cancer progression in p53 mutation-dependent manner. *Journal of Experimental and Clinical Cancer Research*.

[Liu et al., 2009] Liu, G., Wong, L., and Chua, H. N. (2009). Complex discovery from weighted PPI networks. *Bioinformatics*.

[Liu et al., 2017] Liu, G., Zhang, F., Hu, Y., Jiang, Y., Gong, Z., Liu, S., Chen, X., Jiang, Q., and Hao, J. (2017). Genetic Variants and Multiple Sclerosis Risk Gene SLC9A9 Expression in Distinct Human Brain Regions. *Molecular Neurobiology*.

[Liu et al., 2008] Liu, P., Kao, T. P., and Huang, H. (2008). CDK1 promotes cell proliferation and survival via phosphorylation and inhibition of FOXO1 transcription factor. *Oncogene*.

[Lu et al., 2012] Lu, A. G., Feng, H., Wang, P. X. Z., Han, D. P., Chen, X. H., and Zheng, M. H. (2012). Emerging roles of the ribonucleotide reductase M2 in colorectal cancer and ultraviolet-induced DNA damage repair. *World Journal of Gastroenterology*.

[Lu et al., 2011a] Lu, J., Kerns, R. T., Peddada, S. D., and Bushel, P. R. (2011a). Principal component analysis-based filtering improves detection for Affymetrix gene expression arrays. *Nucleic Acids Research*.

[Lu et al., 2011b] Lu, J., Kerns, R. T., Peddada, S. D., and Bushel, P. R. (2011b). Principal component analysis-based filtering improves detection for Affymetrix gene expression arrays. *Nucleic Acids Research*.

[Lu et al., 2014] Lu, W., Fu, Z., Wang, H., Feng, J., Wei, J., and Guo, J. (2014). Peroxiredoxin 2 is upregulated in colorectal cancer and contributes to colorectal cancer cells' survival by protecting cells from oxidative stress. *Molecular and Cellular Biochemistry*.

[Mantione et al., 2014] Mantione, K. J., Kream, R. M., Kuzelova, H., Ptacek, R., Raboch, J., Samuel, J. M., and Stefano, G. B. (2014). Comparing bioinformatic gene expression profiling methods: microarray and RNA-Seq. *Medical science monitor basic research*.

[Marisa et al., 2013] Marisa, L., de Reyniès, A., Duval, A., Selves, J., Gaub, M. P., Vescovo, L., Etienne-Grimaldi, M. C., Schiappa, R., Guenot, D., Ayadi, M., Kirzin, S., Chazal, M., Fléjou, J. F., Benchimol, D., Berger, A., Lagarde, A., Pencreach, E., Piard, F., Elias, D., Parc, Y., Olschwang, S., Milano, G., Laurent-Puig, P., and Boige, V. (2013). Gene Expression Classification of Colon Cancer into Molecular Subtypes: Characterization, Validation, and Prognostic Value. *PLoS Medicine*.

[Marquardt and Snee, 1975] Marquardt, D. W. and Snee, R. D. (1975). Ridge regression in practice. *American Statistician*.

[Mcdermott et al., 2016] Mcdermott, P., Snyder, J., and Willison, R. (2016). Methods for bayesian variable selection with binary response data using the em algorithm. *arXiv: Computation*.

[Menze et al., 2009] Menze, B. H., Kelm, B. M., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W., and Hamprecht, F. A. (2009). A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC Bioinformatics*.

[Merrick et al., 2013] Merrick, B. A., Phadke, D. P., Auerbach, S. S., Mav, D., Stiegelmeyer, S. M., Shah, R. R., and Tice, R. R. (2013). RNA-Seq Profiling Reveals Novel Hepatic Gene Expression Pattern in Aflatoxin B1 Treated Rats. *PLoS ONE*.

[Metsis et al., 2004] Metsis, A., Andersson, U., Baurén, G., Ernfors, P., Lönnerberg, P., Montelius, A., Oldin, M., Pihlak, A., and Linnarsson, S. (2004). Whole-genome expression profiling through fragment display and combinatorial gene identification. *Nucleic acids research*.

[Michiels et al., 2011] Michiels, S., Kramar, A., and Koscielny, S. (2011). Multidimensionality of microarrays: Statistical challenges and (im)possible solutions.

[Mikeska and Dobrovic, 2016] Mikeska, T. and Dobrovic, A. (2016). Epigenetic basis of human cancer. In *The Molecular Basis of Human Cancer*.

[Milani et al., 2010] Milani, L., Lundmark, A., Kiialainen, A., Nordlund, J., Flaegstad, T., Forestier, E., Heyman, M., Jonmundsson, G., Kanerva, J., Schmiegelow, K., Söderhäll, S., Gustafsson, M. G., Lönnerholm, G., and Syvänen, A. C. (2010). DNA methylation for subtype classification and prediction of treatment outcome in patients with childhood acute lymphoblastic leukemia. *Blood*.

[Missiaglia et al., 2009] Missiaglia, E., Selfe, J., Hamdi, M., Williamson, D., Schaaf, G., Fang, C., Koster, J., Summersgill, B., Messahel, B., Versteeg, R., Pritchard-Jones, K., Kool, M., and Shipley, J. (2009). Genomic imbalances in rhabdomyosarcoma cell lines affect expression of genes frequently altered in primary tumors: An approach to identify candidate genes involved in tumor development. *Genes Chromosomes and Cancer*.

[Mitra et al., 2002] Mitra, P., Murthy, C. A., and Pal, S. K. (2002). Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Moore et al., 2013] Moore, L. D., Le, T., and Fan, G. (2013). DNA methylation and its basic function.

[Moroishi et al., 2015] Moroishi, T., Hansen, C. G., and Guan, K. L. (2015). The emerging roles of YAP and TAZ in cancer. *Nature Reviews Cancer*.

[Müller et al., 2019] Müller, A. C., Giambruno, R., Weißer, J., Májek, P., Hofer, A., Bigenzahn, J. W., Superti-Furga, G., Jessen, H. J., Bennett, K. L., Matsushima, Y., Kaguni, L. S., Yuzefovych, L. V., Musiyenko, S. I., Wilson, G. L., Rachek, L. I., Kooij, B. V. D., Creixell, P., Vlimmeren, A. V., Joughin, B. A., Miller, C. J., Haider, N., Linding, R., Stambolic, V., Park, J. H., Zhuang, J., Li, J., Hwang, P. M., Just, W., Reimand, J., Isserlin, R., Voisin, V., Kucera, M., Tannus-Lopes, C., Rostamianfar, A., Wadi, L., Meyer, M., Wong, J., Xu, C., Merico, D., Bader, G. D., cheh Shen, C., Wertelecki, W., Driggers, W. J., LeDoux, S. P., Wilson, G. L., Reis-Filho, J. S., Morrow, M., Carey, L. A., King, T. A., and Newman, L. A. (2019). Pathway enrichment analysis and visualization of omics data using g:Profiler, GSEA, Cytoscape and EnrichmentMap. *Nature Protocols*, 22(1):924–934.

[Murphy et al., 2019] Murphy, C. C., Wallace, K., Sandler, R. S., and Baron, J. A. (2019). Racial Disparities in Incidence of Young-Onset Colorectal Cancer and Patient Survival. *Gastroenterology*.

[Muzny et al., 2012] Muzny, D. M., Bainbridge, M. N., Chang, K., Dinh, H. H., Drummond, J. A., Fowler, G., Kovar, C. L., Lewis, L. R., Morgan, M. B., Newsham, I. F., Reid, J. G., Santibanez, J., Shinbrot, E., Trevino, L. R., Wu, Y.-Q., Wang, M., Gunaratne, P., Donehower, L. A., Creighton, C. J., Wheeler, D. A., Gibbs, R. A., Lawrence, M. S., Voet, D., Jing, R., Cibulskis, K., Sivachenko, A., Stojanov, P., McKenna, A., Lander, E. S., Gabriel, S., Getz, G., Ding, L., Fulton, R. S., Koboldt, D. C., Wylie, T., Walker, J., Dooling, D. J., Fulton, L., Delehaunty, K. D., Fronick, C. C., Demeter, R., Mardis, E. R., Wilson, R. K., Chu, A., Chun, H.-J. E., Mungall, A. J., Pleasance, E., Gordon Robertson, A., Stoll, D., Balasundaram, M., Birol, I., Butterfield, Y. S. N., Chuah, E., Coope, R. J. N., Dhalla, N., Guin, R., Hirst, C., Hirst, M., Holt, R. A., Lee, D., Li, H. I., Mayo, M., Moore, R. A., Schein, J. E., Slobodan, J. R., Tam, A., Thiessen, N., Varhol, R., Zeng, T., Zhao, Y., Jones, S. J. M., Marra, M. A., Bass, A. J., Ramos, A. H., Saksena, G., Cherniack, A. D., Schumacher, S. E., Tabak, B., Carter, S. L., Pho, N. H.,

Nguyen, H., Onofrio, R. C., Crenshaw, A., Ardlie, K., Beroukhim, R., Winckler, W., Getz, G., Meyerson, M., Protopopov, A., Zhang, J., Hadjipanayis, A., Lee, E., Xi, R., Yang, L., Ren, X., Zhang, H., Sathiamoorthy, N., Shukla, S., Chen, P.-C., Haseley, P., Xiao, Y., Lee, S., Seidman, J., Chin, L., Park, P. J., Kucherlapati, R., Todd Auman, J., Hoadley, K. A., Du, Y., Wilkerson, M. D., Shi, Y., Liquori, C., Meng, S., Li, L., Turman, Y. J., Topal, M. D., Tan, D., Waring, S., Buda, E., Walsh, J., Jones, C. D., Mieczkowski, P. A., Singh, D., Wu, J., Gulabani, A., Dolina, P., Bodenheimer, T., Hoyle, A. P., Simons, J. V., Soloway, M., Mose, L. E., Jefferys, S. R., Balu, S., O'Connor, B. D., Prins, J. F., Chiang, D. Y., Neil Hayes, D., Perou, C. M., Hinoue, T., Weisenberger, D. J., Maglinte, D. T., Pan, F., Berman, B. P., Van Den Berg, D. J., Shen, H., Triche Jr, T., Baylin, S. B., Laird, P. W., Getz, G., Noble, M., Voet, D., Saksena, G., Gehlenborg, N., DiCara, D., Zhang, J., Zhang, H., Wu, C.-J., Yingchun Liu, S., Shukla, S., Lawrence, M. S., Zhou, L., Sivachenko, A., Lin, P., Stojanov, P., Jing, R., Park, R. W., Nazaire, M.-D., Robinson, J., Thorvaldsdottir, H., Mesirov, J., Park, P. J., Chin, L., Thorsson, V., Reynolds, S. M., Bernard, B., Kreisberg, R., Lin, J., Iype, L., Bressler, R., Erkkilä, T., Gundapuneni, M., Liu, Y., Norberg, A., Robinson, T., Yang, D., Zhang, W., Shmulevich, I., de Ronde, J. J., Schultz, N., Cerami, E., Ciriello, G., Goldberg, A. P., Gross, B., Jacobsen, A., Gao, J., Kaczkowski, B., Sinha, R., Arman Aksoy, B., Antipin, Y., Reva, B., Shen, R., Taylor, B. S., Chan, T. A., Ladanyi, M., Sander, C., Akbani, R., Zhang, N., Broom, B. M., Casasent, T., Unruh, A., Wakefield, C., Hamilton, S. R., Craig Cason, R., Baggerly, K. A., Weinstein, J. N., Haussler, D., Benz, C. C., Stuart, J. M., Benz, S. C., Zachary Sanborn, J., Vaske, C. J., Zhu, J., Szeto, C., Scott, G. K., Yau, C., Ng, S., Goldstein, T., Ellrott, K., Collisson, E., Cozen, A. E., Zerbino, D., Wilks, C., Craft, B., Spellman, P., Penny, R., Shelton, T., Hatfield, M., Morris, S., Yena, P., Shelton, C., Sherman, M., Paulauskis, J., Gastier-Foster, J. M., Bowen, J., Ramirez, N. C., Black, A., Pyatt, R., Wise, L., White, P., Bertagnolli, M., Brown, J., Chan, T. A., Chu, G. C., Czerwinski, C., Denstman, F., Dhir, R., Dörner, A., Fuchs, C. S., Guillem, J. G., Iacocca, M., Juhl, H., Kaufman, A., Kohl III, B., Van Le, X., Mariano, M. C., Medina,

E. N., Meyers, M., Nash, G. M., Paty, P. B., Petrelli, N., Rabeno, B., Richards, W. G., Solit, D., Swanson, P., Temple, L., Tepper, J. E., Thorp, R., Vakiani, E., Weiser, M. R., Willis, J. E., Witkin, G., Zeng, Z., Zinner, M. J., Network, T. C. G. A., of Medicine, G. S. C. B. C., Institute, G. S. C. B., Louis, G. S. C. W. U. i. S., Agency, G. C. C. B. C. C., Institute, G.-C. C. B., School, G.-C. C. B., Hospital, W., Medical, H., Genome-Characterization Center University of North Carolina, C. H., University, G.-C. C. U. o. S. C., Hopkins, J., Institute, G. D. A. C. B., Biology, G. D. A. C. I. f. S., Center, G. D. A. C. M. S.-K. C., Center, G. D. A. C. U. o. T. M. D. A. C., Genome Data Analysis Centers Santa Cruz and the Buck Institute, U. o. C., Consortium, B. C. R. I. G., Resource, N. C. H. B. C., source sites, and disease working Group, T. (2012). Comprehensive molecular characterization of human colon and rectal cancer. *Nature*, 487(7407):330–337.

[Nangraj et al., 2020] Nangraj, A. S., Selvaraj, G., Kaliamurthi, S., Kaushik, A. C., Cho, W. C., and Wei, D. Q. (2020). Integrated PPI- and WGCNA-Retrieval of Hub Gene Signatures Shared Between Barrett's Esophagus and Esophageal Adenocarcinoma. *Frontiers in Pharmacology*, 11:1.

[Opitz and Maclin, 1999] Opitz, D. and Maclin, R. (1999). Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*.

[Page et al., 2007] Page, G. P., Zakharkin, S. O., Kim, K., Mehta, T., Chen, L., and Zhang, K. (2007). Microarray analysis.

[Pal et al., 2019] Pal, K., Forcato, M., and Ferrari, F. (2019). Hi-C analysis: from data generation to integration. *Biophysical Reviews*, 11(1):67–78.

[Pan et al., 2006] Pan, H. W., Chou, H. Y. E., Liu, S. H., Peng, S. Y., Liu, C. L., and Hsu, H. C. (2006). Role of L2DTL, cell cycle-regulated nuclear and centrosome protein, in aggressive hepatocellular carcinoma. *Cell Cycle*.

[Patil, 2021] Patil, A. R. (2021). R codes for algorithms. `https://sites.google.com/site/abhijeetrpatil01/file-cabinet`. Accessed: 22-Mar-2021.

[Patil et al., 2020a] Patil, A. R., Chang, J., Leung, M.-Y., and Kim, S. (2020a). Analyzing high dimensional correlated data using feature ranking and classifiers. *Computational and Mathematical Biophysics*.

[Patil et al., 2020b] Patil, A. R., Choi, B. J., and Kim, S. (2020b). Improving the classification performance with group lasso-based ranking method in high dimensional correlated data. *Journal of Theoretical and Computational Chemistry*.

[Patil and Kim, 2020] Patil, A. R. and Kim, S. (2020). Combination of ensembles of regularized regression models with resampling-based lasso feature selection in high dimensional data. *Mathematics*.

[Patil et al., 2020c] Patil, A. R., Park, B. K., and Kim, S. (2020c). Adaptive lasso with weights based on normalized filtering scores in molecular big data. *Journal of Theoretical and Computational Chemistry*, 19(4):1–18.

[Peng et al., 2005] Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Peng et al., 2017] Peng, X., Wang, J., Peng, W., Wu, F. X., and Pan, Y. (2017). Protein-protein interactions: detection, reliability assessment and applications.

[Petegrosso et al., 2019] Petegrosso, R., Li, Z., and Kuang, R. (2019). Machine learning and statistical methods for clustering single-cell RNA-sequencing data. *Briefings in Bioinformatics*, 21(4):1209–1223.

[Pi and Halabi, 2018] Pi, L. and Halabi, S. (2018). Combined performance of screening and variable selection methods in ultra-high dimensional data in predicting time-to-event outcomes. *Diagnostic and Prognostic Research*.

[Piao et al., 2019] Piao, J., Zhu, L., Sun, J., Li, N., Dong, B., Yang, Y., and Chen, L. (2019). High expression of CDK1 and BUB1 predicts poor prognosis of pancreatic ductal adenocarcinoma. *Gene*.

[Pidsley et al., 2016] Pidsley, R., Zotenko, E., Peters, T. J., Lawrence, M. G., Risbridger, G. P., Molloy, P., Van Djik, S., Muhlhausler, B., Stirzaker, C., and Clark, S. J. (2016). Critical evaluation of the Illumina MethylationEPIC BeadChip microarray for whole-genome DNA methylation profiling. *Genome Biology*.

[Pirooznia et al., 2008] Pirooznia, M., Yang, J. Y., Qu, M. Q., and Deng, Y. (2008). A comparative study of different machine learning methods on microarray gene expression data. *BMC Genomics*.

[Pletscher-Frankild et al., 2015] Pletscher-Frankild, S., Pallejà, A., Tsafou, K., Binder, J. X., and Jensen, L. J. (2015). DISEASES: Text mining and data integration of disease-gene associations. *Methods*, 74:83–89.

[Portela and Esteller, 2010] Portela, A. and Esteller, M. (2010). Epigenetic modifications and human disease.

[Qi et al., 2020] Qi, R., Ma, A., Ma, Q., and Zou, Q. (2020). Clustering and classification methods for single-cell RNA-sequencing data. *Briefings in Bioinformatics*, 21(4):1196–1208.

[Qian et al., 2014] Qian, X., Ba, Y., Zhuang, Q., and Zhong, G. (2014). RNA-seq technology and its application in fish transcriptomics.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[Quinlan, 1993] Quinlan, J. R. (1993). J. Ross Quinlan_C4.5_ Programs for Machine Learning.pdf.

[Rao et al., 2019] Rao, M. S., Van Vleet, T. R., Ciurlionis, R., Buck, W. R., Mittelstadt, S. W., Blomme, E. A., and Liguori, M. J. (2019). Comparison of RNA-Seq and microarray gene expression platforms for the toxicogenomic evaluation of liver from short-term rat toxicity studies. *Frontiers in Genetics*.

[Raweh et al., 2017] Raweh, A. A., Nassef, M., and Badr, A. (2017). Feature selection and extraction framework for dna methylation in cancer. *International Journal of Advanced Computer Science and Applications*, 8(7).

[Ren and Jin, 2017] Ren, Q. and Jin, B. (2017). The clinical value and biological function of PTTG1 in colorectal cancer. *Biomedicine and Pharmacotherapy*.

[Rish, 2001] Rish, I. (2001). An empirical study of the naive Bayes classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence*.

[Ritchie et al., 2015] Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., and Smyth, G. K. (2015). Limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research*.

[Robertson, 2005] Robertson, K. D. (2005). DNA methylation and human disease.

[Robinson and Oshlack, 2010] Robinson, M. D. and Oshlack, A. (2010). A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology*.

[Rolland et al., 2014] Rolland, T., Taşan, M., Charloteaux, B., Pevzner, S. J., Zhong, Q., Sahni, N., Yi, S., Lemmens, I., Fontanillo, C., Mosca, R., Kamburov, A., Ghiassian, S. D., Yang, X., Ghamsari, L., Balcha, D., Begg, B. E., Braun, P., Brehme, M., Broly, M. P., Carvunis, A. R., Convery-Zupan, D., Corominas, R., Coulombe-Huntington, J., Dann, E., Dreze, M., Dricot, A., Fan, C., Franzosa, E., Gebreab, F., Gutierrez, B. J., Hardy, M. F., Jin, M., Kang, S., Kiros, R., Lin, G. N., Luck, K., Macwilliams, A., Menche, J., Murray, R. R., Palagi, A., Poulin, M. M., Rambout, X., Rasla, J., Reichert, P., Romero, V., Ruyssinck, E., Sahalie, J. M., Scholz, A., Shah, A. A., Sharma, A., Shen,

Y., Spirohn, K., Tam, S., Tejeda, A. O., Trigg, S. A., Twizere, J. C., Vega, K., Walsh, J., Cusick, M. E., Xia, Y., Barabási, A. L., Iakoucheva, L. M., Aloy, P., Rivas, J. D. L., Tavernier, J., Calderwood, M. A., Hill, D. E., Hao, T., Roth, F. P., and Vidal, M. (2014). A proteome-scale map of the human interactome network. *Cell*, 159:1212–1226.

[Rostom et al., 2017] Rostom, R., Svensson, V., Teichmann, S. A., and Kar, G. (2017). Computational approaches for interpreting scRNA-seq data. *FEBS Letters*, 591(15):2213–2225.

[Sabates-Bellver et al., 2007] Sabates-Bellver, J., Van Der Flier, L. G., De Palo, M., Cattaneo, E., Maake, C., Rehrauer, H., Laczko, E., Kurowski, M. A., Bujnicki, J. M., Menigatti, M., Luz, J., Ranalli, T. V., Gomes, V., Pastorelli, A., Faggiani, R., Anti, M., Jiricny, J., Clevers, H., and Marra, G. (2007). Transcriptome profile of human colorectal adenomas. *Molecular Cancer Research*.

[Saito et al., 2018] Saito, M., Momma, T., and Kono, K. (2018). Targeted therapy according to next generation sequencing-based panel sequencing. *Fukushima journal of medical science*.

[Sarkar and Saha,] Sarkar, D. and Saha, S. Machine-learning techniques for the prediction of protein-protein interactions.

[Satoh et al., 2017] Satoh, K., Yachida, S., Sugimoto, M., Oshima, M., Nakagawa, T., Akamoto, S., Tabata, S., Saitoh, K., Kato, K., Sato, S., Igarashi, K., Aizawa, Y., Kajino-Sakamoto, R., Kojima, Y., Fujishita, T., Enomoto, A., Hirayama, A., Ishikawa, T., Taketo, M. M., Kushida, Y., Haba, R., Okano, K., Tomita, M., Suzuki, Y., Fukuda, S., Aoki, M., and Soga, T. (2017). Global metabolic reprogramming of colorectal cancer occurs at adenoma stage and is induced by MYC. *Proceedings of the National Academy of Sciences of the United States of America*.

[Shannon et al., 2003] Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). Cytoscape: A soft-

ware Environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504.

[Shi Jing et al., 2015] Shi Jing, L., Fathiah Muzaffar Shah, F., Saberi Mohamad, M., Moorthy, K., Deris, S., Zakaria, Z., and Napis, S. (2015). A Review on Bioinformatics Enrichment Analysis Tools Towards Functional Analysis of High Throughput Gene Set Data. *Current Proteomics*, 12(1):14–27.

[Shibue et al., 2003] Shibue, T., Takeda, K., Oda, E., Tanaka, H., Murasawa, H., Takaoka, A., Morishita, Y., Akira, S., Taniguchi, T., and Tanaka, N. (2003). Integral role of Noxa in p53-mediated apoptotic response. *Genes and Development*.

[Siegel et al., 2020] Siegel, R. L., Miller, K. D., Goding Sauer, A., Fedewa, S. A., Butterly, L. F., Anderson, J. C., Cercek, A., Smith, R. A., and Jemal, A. (2020). Colorectal cancer statistics, 2020. *CA: A Cancer Journal for Clinicians*.

[Sillars-Hardebol et al., 2012] Sillars-Hardebol, A. H., Carvalho, B., Tijssen, M., Beliën, J. A., De Wit, M., Delis-van Diemen, P. M., Pontén, F., Van De Wiel, M. A., Fijneman, R. J., and Meijer, G. A. (2012). TPX2 and AURKA promote 20q amplicon-driven colorectal adenoma to carcinoma progression. *Gut*.

[Simon et al., 2003] Simon, R., Atefy, R., Wagner, U., Forster, T., Fijan, A., Bruderer, J., Wilber, K., Mihatsch, M. J., Gasser, T., and Sauter, G. (2003). HER-2 and TOP2A coamplification in urinary bladder cancer. *International Journal of Cancer*.

[Skrzypczak et al., 2010] Skrzypczak, M., Goryca, K., Rubel, T., Paziewska, A., Mikula, M., Jarosz, D., Pachlewski, J., Oledzki, J., and Ostrowsk, J. (2010). Modeling oncogenic signaling in colon tumors by multidirectional analyses of microarray data directed for maximization of analytical reliability. *PLoS ONE*.

[Slattery et al., 2019] Slattery, M. L., Mullany, L. E., Wolff, R. K., Sakoda, L. C., Samowitz, W. S., and Herrick, J. S. (2019). The p53-signaling pathway and colorectal cancer: Interactions between downstream p53 target genes and miRNAs. *Genomics*.

[Slimane et al., 2020] Slimane, S. N., Marcel, V., Fenouil, T., Catez, F., Saurin, J. C., Bouvet, P., Diaz, J. J., and Mertani, H. C. (2020). Ribosome Biogenesis Alterations in Colorectal Cancer.

[Sobek et al., 2006] Sobek, J., Bartscherer, K., Jacob, A., Hoheisel, J., and Angenendt, P. (2006). Microarray Technology as a Universal Tool for High-Throughput Analysis of Biological Systems. *Combinatorial Chemistry & High Throughput Screening*.

[Solé et al., 2014] Solé, X., Crous-Bou, M., Cordero, D., Olivares, D., Guinó, E., Sanz-Pamplona, R., Rodriguez-Moranta, F., Sanjuan, X., De Oca, J., Salazar, R., and Moreno, V. (2014). Discovery and validation of new potential biomarkers for early detection of colon cancer. *PLoS ONE*.

[Song and Rape, 2010] Song, L. and Rape, M. (2010). Regulated Degradation of Spindle Assembly Factors by the Anaphase-Promoting Complex. *Molecular Cell*.

[Soria et al., 2011] Soria, D., Garibaldi, J. M., Ambrogi, F., Biganzoli, E. M., and Ellis, I. O. (2011). A 'non-parametric' version of the naive Bayes classifier. *Knowledge-Based Systems*.

[Stark et al., 2006] Stark, C., Breitkreutz, B. J., Reguly, T., Boucher, L., Breitkreutz, A., and Tyers, M. (2006). BioGRID: a general repository for interaction datasets. *Nucleic acids research*, 34(Database issue):D535–D539.

[Stark et al., 2019] Stark, R., Grzelak, M., and Hadfield, J. (2019). RNA sequencing: the teenage years.

[Stedman et al., 2015] Stedman, A., Beck-Cormier, S., Le Bouteiller, M., Raveux, A., Vandormael-Pournin, S., Coqueran, S., Lejour, V., Jarzebowski, L., Toledo, F., Robine,

S., and Cohen-Tannoudji, M. (2015). Ribosome biogenesis dysfunction leads to p53-mediated apoptosis and goblet cell differentiation of mouse intestinal stem/progenitor cells. *Cell Death and Differentiation*.

[Stolzenburg et al., 2016] Stolzenburg, S., Goubert, D., and Rots, M. G. (2016). DNA Methyltransferases - Role and Function. *Springer International*.

[Strobl et al., 2007] Strobl, C., Boulesteix, A. L., Zeileis, A., and Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*.

[Su et al., 2020] Su, C., Tong, J., and Wang, F. (2020). Mining genetic and transcriptomic data using machine learning approaches in Parkinson's disease.

[Su and Yang, 2008] Su, C. T. and Yang, C. H. (2008). Feature selection for the SVM: An application to hypertension diagnosis. *Expert Systems with Applications*.

[Sun and Wang, 2012] Sun, H. and Wang, S. (2012). Penalized logistic regression for high-dimensional DNA methylation data with case-control studies. *Bioinformatics*.

[Sun et al., 2019] Sun, J., Wu, Q., Shen, D., Wen, Y., Liu, F., Gao, Y., Ding, J., and Zhang, J. (2019). TSLRF: Two-Stage Algorithm Based on Least Angle Regression and Random Forest in genome-wide association studies. *Scientific Reports*.

[Szklarczyk et al., 2017] Szklarczyk, D., Morris, J. H., Cook, H., Kuhn, M., Wyder, S., Simonovic, M., Santos, A., Doncheva, N. T., Roth, A., Bork, P., Jensen, L. J., and Mering, C. V. (2017). The STRING database in 2017: Quality-controlled protein-protein association networks, made broadly accessible. *Nucleic Acids Research*, 45(D1):D362–D368.

[Takahashi et al., 2003] Takahashi, T., Sano, B., Nagata, T., Kato, H., Sugiyama, Y., Kunieda, K., Kimura, M., Okano, Y., and Saji, S. (2003). Polo-like kinase 1 (PLK1) is overexpressed in primary colorectal cancers. *Cancer Science*.

[Tao et al., 2014] Tao, J., Zhi, X., Tian, Y., Li, Z., Zhu, Y., Wang, W., Xie, K., Tang, J., Zhang, X., Wang, L., and Xu, Z. (2014). CEP55 contributes to human gastric carcinoma by regulating cell proliferation. *Tumor Biology*.

[Teschendorff et al., 2010] Teschendorff, A. E., Menon, U., Gentry-Maharaj, A., Ramus, S. J., Weisenberger, D. J., Shen, H., Campan, M., Noushmehr, H., Bell, C. G., Maxwell, A. P., Savage, D. A., Mueller-Holzner, E., Marth, C., Kocjan, G., Gayther, S. A., Jones, A., Beck, S., Wagner, W., Laird, P. W., Jacobs, I. J., and Widschwendter, M. (2010). Age-dependent DNA methylation of genes that are suppressed in stem cells is a hallmark of cancer. *Genome Research*.

[Thorsteinsson and Jess, 2011] Thorsteinsson, M. and Jess, P. (2011). The clinical significance of circulating tumor cells in non-metastatic colorectal cancer - A review.

[Tibshirani, 1996] Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*.

[Toufighi et al., 2005] Toufighi, K., Brady, S. M., Austin, R., Ly, E., and Provart, N. J. (2005). The Botany Array Resource: e-Northerns, Expression Angling, and promoter analyses. *The Plant Journal*, 43(1):153–163.

[Toyota et al., 1999] Toyota, M., Ahuja, N., Ohe-Toyota, M., Herman, J. G., Baylin, S. B., and Issa, J. P. J. (1999). CpG island methylator phenotype in colorectal cancer. *Proceedings of the National Academy of Sciences of the United States of America*.

[Tsompana and Buck, 2014] Tsompana, M. and Buck, M. J. (2014). Chromatin accessibility: a window into the genome. *Epigenetics Chromatin*, 7(1):33.

[Ueki et al., 2008] Ueki, T., Nishidate, T., Park, J. H., Lin, M. L., Shimo, A., Hirata, K., Nakamura, Y., and Katagiri, T. (2008). Involvement of elevated expression of multiple cell-cycle regulator, DTL/RAMP (denticleless/RA-regulated nuclear matrix associated protein), in the growth of breast cancer cells. *Oncogene*.

[UM, 2013] UM, O. (2013). Estimating the Fisher's Scoring Matrix Formula from Logistic Model. *American Journal of Theoretical and Applied Statistics*.

[Urbanowicz et al., 2018] Urbanowicz, R. J., Meeker, M., La Cava, W., Olson, R. S., and Moore, J. H. (2018). Relief-based feature selection: Introduction and review.

[Urdinguio et al., 2009] Urdinguio, R. G., Sanchez-Mut, J. V., and Esteller, M. (2009). Epigenetic mechanisms in neurological diseases: genes, syndromes, and therapies.

[Verset et al., 2013] Verset, L., Tommelein, J., Moles Lopez, X., Decaestecker, C., Mareel, M., Bracke, M., Salmon, I., De Wever, O., and Demetter, P. (2013). Epithelial expression of FHL2 is negatively associated with metastasis-free and overall survival in colorectal cancer. *British Journal of Cancer*.

[Wang et al., 2016] Wang, Y., Jin, T., Dai, X., and Xu, J. (2016). Lentivirus-mediated knockdown of CEP55 suppresses cell proliferation of breast cancer cells. *BioScience Trends*.

[Wang et al., 2008] Wang, Y., Miller, D. J., and Clarke, R. (2008). Approaches to working in high-dimensional data spaces: Gene expression microarrays.

[Wang et al., 2019] Wang, Y., Yang, X. G., and Lu, Y. (2019). Informative gene selection for microarray classification via adaptive elastic net with conditional mutual information. *Applied Mathematical Modelling*.

[Wang et al., 2018] Wang, Z. Z., Yang, J., Jiang, B. H., Di, J. B., Gao, P., Peng, L., and Su, X. Q. (2018). KIF14 promotes cell proliferation via activation of Akt and is directly targeted by miR-200c in colorectal cancer. *International Journal of Oncology*.

[Weber et al., 2007] Weber, M., Hellmann, I., Stadler, M. B., Ramos, L., Pääbo, S., Rebhan, M., and Schübeler, D. (2007). Distribution, silencing potential and evolutionary impact of promoter DNA methylation in the human genome. *Nature Genetics*.

[Wu et al., 2019] Wu, M., Liu, Z., Li, X., Zhang, A., Lin, D., and Li, N. (2019). Analysis of potential key genes in very early hepatocellular carcinoma. *World Journal of Surgical Oncology*.

[Xu et al., 2020] Xu, H., Ma, Y., Zhang, J., Gu, J., Jing, X., Lu, S., Fu, S., and Huo, J. (2020). Identification and Verification of Core Genes in Colorectal Cancer. *BioMed Research International*.

[Xu et al., 2013] Xu, X., Zhang, Y., Williams, J., Antoniou, E., McCombie, W. R., Wu, S., Zhu, W., Davidson, N. O., Denoya, P., and Li, E. (2013). Parallel comparison of Illumina RNA-Seq and Affymetrix microarray platforms on transcriptomic profiles generated from 5-aza-deoxy-cytidine treated HT-29 colon cancer cells and simulated datasets. *BMC Bioinformatics*.

[Xue et al., 2016] Xue, X., Ramakrishnan, S. K., Weisz, K., Triner, D., Xie, L., Attili, D., Pant, A., Győrffy, B., Zhan, M., Carter-Su, C., Hardiman, K. M., Wang, T. D., Dame, M. K., Varani, J., Brenner, D., Fearon, E. R., and Shah, Y. M. (2016). Iron Uptake via DMT1 Integrates Cell Cycle with JAK-STAT3 Signaling to Promote Colorectal Tumorigenesis. *Cell Metabolism*.

[Yan et al., 2020] Yan, F., Powell, D. R., Curtis, D. J., and Wong, N. C. (2020). From reads to insight: A hitchhiker's guide to ATAC-seq data analysis.

[Yang and Zou, 2015] Yang, Y. and Zou, H. (2015). A fast unified algorithm for solving group-lasso penalize learning problems. *Statistics and Computing*, 25:1129–1141.

[Yang and Zou, 2017] Yang, Y. and Zou, H. (2017). *gglasso: Group Lasso Penalized Learning Using a Unified BMD Algorithm*. R package version 1.4.

[Yap et al., 2004] Yap, Y. L., Zhang, X. W., Ling, M. T., Wang, X. H., Wong, Y. C., and Danchin, A. (2004). Classification between normal and tumor tissues based on the pair-wise gene expression ratio. *BMC Cancer*.

[Yuan et al., 2020] Yuan, Y., Chen, J., Wang, J., Xu, M., Zhang, Y., Sun, P., and Liang, L. (2020). Identification Hub Genes in Colorectal Cancer by Integrating Weighted Gene Co-Expression Network Analysis and Clinical Validation in vivo and vitro. *Frontiers in Oncology*.

[Zhang, 2010] Zhang, C. H. (2010). Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*.

[Zhang et al., 2020] Zhang, D., Hu, Q., Liu, X., Zou, K., Sarkodie, E. K., Liu, X., and Gao, F. (2020). AllEnricher: A comprehensive gene set function enrichment tool for both model and non-model species. *BMC Bioinformatics*, 21(1):1–6.

[Zhao et al., 2019] Zhao, B., Baloch, Z., Ma, Y., Wan, Z., Huo, Y., Li, F., and Zhao, Y. (2019). Identification of Potential Key Genes and Pathways in Early-Onset Colorectal Cancer Through Bioinformatics Analysis. *Cancer Control*.

[Zhao et al., 2014] Zhao, S., Fung-Leung, W. P., Bittner, A., Ngo, K., and Liu, X. (2014). Comparison of RNA-Seq and microarray in transcriptome profiling of activated T cells. *PLoS ONE*.

[Zhu et al., 2005] Zhu, C., Zhao, J., Bibikova, M., Leverson, J. D., Bossy-Wetzel, E., Fan, J. B., Abraham, R. T., and Jiang, W. (2005). Functional analysis of human microtubule-based motor proteins, the kinesins and dyneins, in mitosis/cytokinesis using RNA interference. *Molecular Biology of the Cell*.

[Zhuang et al., 2012] Zhuang, J., Widschwendter, M., and Teschendorff, A. E. (2012). A comparison of feature selection and classification methods in DNA methylation studies using the Illumina Infinium platform. *BMC Bioinformatics*.

[Zou, 2006] Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*.

[Zou and Hastie, 2005] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*.

Appendix A

Appendix A: The Resampling-based lasso feature selection and Ensembles of Regularized Regression method

```
setwd("/export/home/arpatil/Desktop/ge/parallel/parallel_corr_05/")
####################
#####---Import libraries
####################
library(mvtnorm)
library(doParallel)
library(class)
library(adabag)
library(tidyverse)
library(Biocomb)
library(praznik)
library(randomForest)
library(glmnet)
library(ncvreg)
library(e1071)
library(caret)
library(RankAggreg)
library(ggplot2)
################
```

```
#####----------set parameters ------#################

reps = 100  # number of iterations
M = 100     # number of trees for ensemble
rftree = 100 # number of trees for random forests
adatree = 100 # number of trees for adaboost
# used for convert scores functions ( greedy ensemble ranking)
distance="Spearman"
# used inside ensemble for scores
weighted= TRUE
# true number of important variables
true.imp <- c(1:25)
# performance metrics
validation = c("accuracy","sensitivity","specificity")
# algorithns inside ensemble
algorithms = c("lasso", "alasso", "bridge", "scad", "mcp")
top <- 75
################
##---------------- performance metrics------
################

accuracy <- function(truth, predicted)
{
  if(length(truth) > 0)
    sum(truth==predicted)/length(truth) else
      return(0)
}
```

```
sensitivity <- function(truth, predicted)
{
  if(sum(truth==1) > 0)
    sum(predicted[truth==1]==1)/sum(truth==1)    else
      return(0)
}

specificity <- function(truth, predicted)
{
  if(sum(truth==0) > 0)
    sum(predicted[truth==0]==0)/sum(truth==0)    else
      return(0)
}

convertScores <- function(scores)
{
  scores <- t(scores)
  ranks <- matrix(0, nrow(scores), ncol(scores))
  weights <- ranks
  for(i in 1:nrow(scores)){
    ms <- sort(scores[i,], decr=TRUE, ind=TRUE)
    ranks[i,] <- colnames(scores)[ms$ix]
    weights[i,] <- ms$x
  }
  list(ranks = ranks, weights = weights)
}
###############
##---------- main resampling methods ----------
```

```
###############

freq.matrix = function(train.x, train.y, method)
{
  if(method=="lasso")
  {
    fit = glmnet(as.matrix(train.x), train.y, alpha=1,
    family="binomial")
    beta.matrix = fit$beta
  }
  if(method=="bridge")
  {
    fit = glmnet(as.matrix(train.x), train.y, alpha=(1/2),
    family="binomial")
    beta.matrix = fit$beta
  }

  if(method=="scad")
  {
    fit = ncvreg(as.matrix(train.x), train.y,
    family="binomial",penalty="SCAD")
    beta.matrix = fit$beta[-c(1),]
  }

  if(method=="mcp")
  {
    fit = ncvreg(as.matrix(train.x), train.y,
    family="binomial",penalty="MCP")
```

```
    beta.matrix = fit$beta[-c(1),]
  }
  ##################################################################
  # record frequency with max regression coefficients
  result.matrix = coefs.matrix = matrix(0,nrow=ncol(beta.matrix),
  ncol=nrow(beta.matrix))
  colnames(result.matrix) = rownames(beta.matrix)
  for(i in 1:nrow(result.matrix))
  {
    idx = which(beta.matrix[,i]!=0)
    result.matrix[i,idx] = 1
    coefs.matrix[i,idx] = beta.matrix[idx,i]
  }
  # return frequency matrix
  return(list(result.matrix,coefs.matrix))
}

resampling = function(train.x,train.y,method)
{
  ## resampling
  idx = sample(1:nrow(train.x),as.integer(nrow(train.x)*0.7))
  s.x = train.x[idx,]
  s.y = train.y[idx]

  if(method=="lasso")
  {
    obj = freq.matrix(s.x,s.y,method)
  }
```

```
  if(method=="bridge")
  {
    obj = freq.matrix(s.x,s.y,method)
  }
  if(method=="scad")
  {
    obj = freq.matrix(s.x,s.y,method)
  }
  if(method=="mcp")
  {
    obj = freq.matrix(s.x,s.y,method)
  }
  # return frequency matrix and coeficients matrix
  return(obj)
}

add = function(obj.mat,option)
{
  if(option=="frequency"){s = apply(obj.mat,2,sum)}
  if(option=="coef"){s = apply(obj.mat,2,function(x)max(abs(x)))}
  return(s)
}
#########################################################################
resampling_results = function(train.x,train.y,method)
{
  # 100 resampling
  # return two lists
  # frequency and coefficients
```

```
  # for each variables
  mat0 = matrix(0,nrow=100,ncol=ncol(train.x))
  mat1 = matrix(0,nrow=100,ncol=ncol(train.x))
  colnames(mat0) = colnames(mat1) = colnames(train.x)
  lens = NULL
  for(i in 1:100)  ##############################################
  {
    # lasso
    obj0 = resampling(train.x,train.y,method=method)
    lens = c(lens,nrow(obj0[[1]]))
    mat0[i,] = add(obj0[[1]],"frequency")
    mat1[i,] = add(obj0[[2]],"coef")
  }
  freq.vec = apply(mat0,2,sum)/sum(lens)
  coef.vec = apply(mat1,2,function(x)max(abs(x)))
  return(list(freq.vec,coef.vec))
}

main = function(train.x,train.y,method)#,directory)
{
  results = resampling_results(train.x,train.y,method)
  mu.vec      = results[[1]]
  max.coefs.vec     = results[[2]]

  s.mu.vec = sort(mu.vec,decreasing=T,index.return=T)
  rank.mu.vec     = mu.vec[s.mu.vec$ix]
  return(rank.mu.vec)
}
```

```
#################################################################
##-- code for penalized logistic regression method
##-------- lasso --------
lasso = function(x,y)
{
  cv.lasso <- cv.glmnet(y = y, x = x, alpha=1, family="binomial")
  return(cv.lasso)
}
##---------- alasso ----------
alasso = function(x,y)
{
  ## Ridge Regression to create the Adaptive Weights Vector
  cv.ridge <- cv.glmnet(y = y, x = x, alpha=0, family='binomial')
  w3 <- 1/abs(matrix(coef(cv.ridge, s=cv.ridge$lambda.min)
  [, 1][2:(ncol(x)+1)] ))^1 ## Using gamma = 1
  ## Replacing values estimated as Infinite for 999999999
  w3[w3[,1] == Inf] <- 999999999

  ## Adaptive Lasso
  cv.alasso <- cv.glmnet(y = y, x = x, alpha=1,
  family="binomial", penalty.factor = w3)
  return(cv.alasso)
}
##---------- bridge ----------
enet = function(x,y)
{
  cv.enet <- cv.glmnet(y=y, x= x, alpha=0.5, family="binomial")
```

```
  return(cv.enet)
}
##------------------------------ SCAD ----------
scad = function(x,y)
{
  cv.scad <- cv.ncvreg(x, y,family="binomial",penalty="SCAD")
  return(cv.scad)
}

##---------- MCP ----------
mcp = function(x,y)
{
  cv.mcp <- cv.ncvreg(x, y,family="binomial",penalty="MCP")
  return(cv.mcp)
}
##########
######---------- Simulation Data Generation Functions ----
##########
data.gen <- function()
{
  feature_set = function(nsample, p, seed.index, beta, cors )
  {

    set.seed(seed.index)
    cor.mat = matrix(1, nrow = p, ncol = p)
    cor.mat[lower.tri(cor.mat)]=cors
    # cor.mat[upper.tri(cor.mat)]=cors
    cor.mat=t(cor.mat)
```

```
  cor.mat[lower.tri(cor.mat)] = cors
  mu.vec=rep(0,p)
  dat = rmvnorm(n = nsample, mean = mu.vec, sigma = cor.mat,
  method = "svd")  # change to eigen or chol or svd if needed
  return(dat)
}

generate_response_data = function(nsample, p, cors, theta0)
{
  cnt      = 0
  case.dat = NULL
  control.dat = NULL
  # generate response of equal control and case
  while(cnt==0)
  {
    seed.val = sample(1:1000000,1)
    data.set = feature_set(nsample, p, seed.val, beta, cors)
    data.set = scale(data.set)
    feta = data.set%*%theta0
    fprob = exp(feta)/(1+exp(feta))
    y = rbinom(nsample, 1, fprob)
    idx = which(y == 1)

    if((length(idx)==(nsample/2)))
    {
      result = list(y,data.set,seed.val)
      cnt    = 100
      return(result)
```

```
    }
  }
}

# Generate dataset along with response variable
# and true imp variable names
nsample = 200  # number of samples (change)
cors    = 0.5  # correlation (change)
p = 1000       # number of variables (change)
num.beta = 25  # True number of variables (change)
# regression coefs (beta) values for the total number of betas
beta  = runif(num.beta,2,4)
true.coefs = beta      # Copy the beta to other variable
# indices of the true regression coefs in the dataset
true.coefs.idx = c(1:num.beta)
theta0 = rep(0,p) # generate the empty list of values
# insert the regression coefs in the previous list
theta0[c(true.coefs.idx)] = c(true.coefs)

obj <- generate_response_data(nsample = nsample, p,
cors =  cors, theta0 = theta0)

y1      = obj[[1]]  # response variable
x1      = data.frame(obj[[2]]) # data set
df <- data.frame(x1,y1)
colnames(df)=paste("v",1:ncol(df),sep="")
colnames(df)[ncol(df)]<- "y"
true.var.names = paste("v",c(true.coefs.idx),sep="")
```

```
  return(df)
}
#########
### Real data generation function
########
data.gen <- function()
{
  data.SMK_CAN_187 <- readMat("SMK_CAN_187.mat")
  x1 <- as.matrix(data.SMK_CAN_187$X)
  y1 <- as.numeric(data.SMK_CAN_187$Y)
  y1 <- replace(y1, y1==1, 0)
  y1 <- replace(y1, y1==2, 1)

  s.idx <- sample(1:nrow(x1),as.integer(nrow(x1)*.7),replace = F)

  train.y = y1[s.idx]
  train.x = x1[s.idx,]
  test.x =  x1[-s.idx,]
  test.y =  y1[-s.idx]
  loglk  = NULL
  for(i in 1:ncol(train.x))
  {
    fit = glm(train.y~train.x[,i],family="binomial")
    loglk = c(loglk, -as.numeric(logLik(fit)))
  }
  s  = sort.int(loglk,index.return=T)
  train.x = train.x[,s$ix]
```

```
  test.x  = test.x[,s$ix]
  train.x = train.x[,1:10000]
  test.x  = test.x[,1:10000]

  data.train <- data.frame(train.x, train.y)
  data.test <- data.frame(test.x, test.y)

  return(list(data.train, data.test))
}

feature.selection = function(method, data)
{
  # Resampling method
  df <- data

  # Split the data into training and testing set
  bound <- floor((nrow(df)/4)*3)
  df <- df[sample(nrow(df)), ]    #sample rows
  df.train <- df[1:bound, ]       #get training set
  df.test <- df[(bound+1):nrow(df), ]

  #-------------------------
  train.x <-data.matrix(df.train[,-(ncol(df.train))])
  train.y <- as.numeric(df.train[,(ncol(df.train))])
  # Test set seperated into data matrix and response variable
  test.x <- data.matrix(df.test[,-(ncol(df.test))])
  test.y <- as.numeric(df.test[,(ncol(df.test))])
  # Last column name is train.y
```

```
data.train <- data.frame(train.x, train.y)
# Last column name is test.y
data.test <- data.frame(test.x, test.y)

if(method == "resampling")
{
  # for lasso resampling approach
  final_lasso <- main(train.x = train.x, train.y = train.y,
                      method = "lasso")
  set_lasso <- names(final_lasso[1:top])
  idx <- match(set_lasso, names(data.train))
}
# information gain
if(method=="information")
{
  # class label must be factor
  data.train[,ncol(data.train)]<-as.factor(
  data.train[,ncol(data.train)])
  disc<-"equal_interval_width"
  attrs.nominal <- numeric()
  fit.info = select.inf.gain(data.train,
  disc.method=disc,attrs.nominal=attrs.nominal)
  idx <- as.integer(fit.info$NumberFeature)[1:top]
}
# chi2
if(method=="chi2")
{
  # class label must be factor
```

```
    data.train[,ncol(data.train)]<-as.factor(
    data.train[,ncol(data.train)])
    disc<-"equal interval width"
    attrs.nominal <- numeric()
    fit.chi2 <- select.inf.chi2(data.train,
    disc.method=disc,attrs.nominal=attrs.nominal)
    idx <- as.integer(fit.chi2$NumberFeature)[1:top]

  }
  # mrmr
  if(method=="mrmr")
  {
    # class label doesnt matter
    fit.mrmr = MRMR(data.train[,-ncol(data.train)],
    data.train[,ncol(data.train)],
    k=ncol(data.train[,-ncol(data.train)]))
    fm<-fit.mrmr$score
    set <- names(fm)[1:top]
    idx <- match(set, names(data.train))

  }
  # return set of features
  return(list(data.train, data.test, idx))
}

# inside the test function

test.function <- function(feature.idx, method)
```

```
{
  fs <- feature.idx
  data.train <- fs[[1]]
  data.test <- fs[[2]]
  idx <- fs[[3]]
  # M = 100
  #-------------------------------------------------
  reduced.trainx <- data.train[,idx]
  reduced.trainy <- as.factor(data.train[,ncol(data.train)])

  test.x <- data.test[,idx]
  test.y <- as.factor(data.test[,(ncol(data.test))])
  #-------------------------------------------------
  dat.train <- data.frame(reduced.trainx, reduced.trainy)
  #-------------------------------------------------
  n <- length(reduced.trainy)
  nalg <- length(algorithms)
  nvm <- length(validation)
  ly <- levels(reduced.trainy)

  fittedModels <- list() #to keep the fitted algorithms
  coefs.mat = freq.mat = matrix(0,nrow=M,
                                ncol=ncol(reduced.trainx))
  colnames(coefs.mat)=colnames(freq.mat)=
  colnames(reduced.trainx)
  best.methods=NULL
  mat.result = matrix(0,nrow=nrow(test.x),ncol=M)
```

```
if(method=="ensemble")
{
  for(k in 1:M)
  {
    repeat
    {
      s <- sample(n, replace=TRUE)
      if(length(table(reduced.trainy[s])) >= 2 & length(
                        table(reduced.trainy[-s])) >= 2)
        break
    }
    fs <- 1:ncol(reduced.trainx)
    sub.trainingx <- reduced.trainx[s, fs]
    sub.testingx  <- reduced.trainx[-unique(s), fs]
    sub.trainingy <- reduced.trainy[s]
    sub.testingy <- reduced.trainy[-unique(s)]

    predicted <- list()
    Res <- list()
    # train all algorithms on the subset and fit the model
    # Fit the model and predict
    for(j in 1:nalg)
    {
      Res[[j]]<- switch(algorithms[j],

            "lasso"  =  {
            fit.lasso = lasso(as.matrix(sub.trainingx),
            sub.trainingy)
```

```
lasso.idx = which(abs(coef(fit.lasso)[,1])>0)[-c(1)]
lasso.m.idx = match(names(lasso.idx),colnames(
sub.testingx))
lasso.xb = as.matrix(cbind(1,as.matrix(
sub.testingx[,lasso.m.idx])))%*%as.matrix(
coef(fit.lasso)[c(1,lasso.idx),1])
lasso.pred = exp(lasso.xb)/(1+exp(lasso.xb))
lasso.pred <- ifelse(lasso.pred>0.5, 1, 0)
predicted[[j]] <- as.character(lasso.pred[,1])
                    },
"alasso"  =  {
 fit.alasso = alasso(as.matrix(sub.trainingx),
                    sub.trainingy)
 alasso.idx = which(abs(coef(fit.alasso)[,1])>0)
 [-c(1)]
 alasso.m.idx = match(names(alasso.idx),
 colnames(sub.testingx))
 alasso.xb = as.matrix(cbind(1,as.matrix(sub.testingx
 [,alasso.m.idx])))%*%as.matrix(coef(fit.alasso)
 [c(1,alasso.idx),1])
 alasso.pred = exp(alasso.xb)/(1+exp(alasso.xb))
 alasso.pred <- ifelse(alasso.pred>0.5, 1, 0)
 predicted[[j]] <- as.character(alasso.pred[,1])


                    },
"bridge"  = {
fit.enet = enet(as.matrix(sub.trainingx), sub.trainingy)
enet.idx = which(abs(coef(fit.enet)[,1])>0)[-c(1)]
```

```
enet.m.idx = match(names(enet.idx),colnames(sub.testingx))
enet.xb = as.matrix(cbind(1,as.matrix(sub.testingx
[,enet.m.idx])))%*%as.matrix(coef(fit.enet)
[c(1,enet.idx),1])
enet.pred = exp(enet.xb)/(1+exp(enet.xb))
enet.pred <- ifelse(enet.pred>0.5, 1, 0)
predicted[[j]] <- as.character(enet.pred[,1])
            },
"scad"  = {
 fit.scad  = scad(as.matrix(sub.trainingx), sub.trainingy)
 scad.idx = which(abs(coef(fit.scad))>0)[-c(1)]
 scad.m.idx = match(names(scad.idx),colnames(sub.testingx))
 scad.xb = as.matrix(cbind(1,as.matrix(sub.testingx
 [,scad.m.idx])))%*%as.matrix(coef(fit.scad)[c(1,scad.idx)])
 scad.pred = exp(scad.xb)/(1+exp(scad.xb))
 scad.pred <- ifelse(scad.pred>0.5, 1, 0)
 predicted[[j]] <- as.character(scad.pred[,1])
          },
 "mcp"  = {
 fit.mcp  = mcp(as.matrix(sub.trainingx), sub.trainingy)
 mcp.idx = which(abs(coef(fit.mcp))>0)[-c(1)]
 mcp.m.idx = match(names(mcp.idx),colnames(sub.testingx))
 mcp.xb = as.matrix(cbind(1,as.matrix(sub.testingx
 [,mcp.m.idx])))%*%as.matrix(coef(fit.mcp)[c(1,mcp.idx)])
 mcp.pred = exp(mcp.xb)/(1+exp(mcp.xb))
 mcp.pred <- ifelse(mcp.pred>0.5, 1, 0)
 predicted[[j]] <- as.character(mcp.pred[,1])
          }
```

```
  )
  attr(Res[[j]], "algorithm") <- algorithms[j]
}
# compute validation measures
scores <- matrix(0, nalg, nvm)
rownames(scores) <- algorithms
colnames(scores) <- validation

for(i in 1:nalg)
  for(j in 1:nvm)
    scores[i,j] <- switch(validation[j],
       "accuracy"     = accuracy(sub.testingy,
       factor(predicted[[i]], levels=ly)),
      "sensitivity" = sensitivity(sub.testingy,
      factor(predicted[[i]], levels=ly)),
      "specificity" = specificity(sub.testingy,
      factor(predicted[[i]], levels=ly))
    )
# Rank aggregation to select the best model
# based on the performance metrics
convScores <- convertScores(scores)
#-----------------------------------
if(nvm > 1 && nalg <= 5)
  if(weighted)
    fittedModels[[k]] <- Res[[which(algorithms ==
    BruteAggreg(convScores$ranks, nalg, convScores$weights,
    distance=distance)$top.list[1])]]
#-----------------------------------------
```

```
bestAlg <- unlist(sapply(fittedModels,
FUN = function(x) attr(x, "algorithm")))
newdata<- test.x
predicted_sub <- list()
Res_sub<- list()
# bestAlg = best.methods

for (s in 1:nalg)
{
  Res_sub[[s]]<- switch(bestAlg[s],
   "lasso"  =          {
  reslasso_sub <- predict(fit.lasso,
  newx = as.matrix(newdata), type = "class")
  predicted_sub[[s]]<- reslasso_sub[,1]
  },
  "alasso"  =          {
  resalasso_sub <- predict(fit.alasso,
  newx = as.matrix(newdata), type = "class")
  predicted_sub[[s]]<- resalasso_sub[,1]
  },
   "bridge"  =          {
  resenet_sub <- predict(fit.enet,
  newx = as.matrix(newdata), type = "class")
  predicted_sub[[s]]<- resenet_sub[,1]
   },
   "scad"  = {
  resscad_sub <- predict(fit.scad,
  X = as.matrix(newdata), type = "class")
```

```
        predicted_sub[[s]]<- resscad_sub
         },
         "mcp"   = {
        resmcp_sub <- predict(fit.mcp,
        X = as.matrix(newdata), type = "class")
        predicted_sub[[s]]<- resmcp_sub
         }
        )
      }
      #store predicted classification
      mat.result[,k] = Res_sub[[1]]

      # some output
      verbose=T
      if(verbose)
        cat("Iter ", k, "\n")
    }
    majority.vote = apply(mat.result,1,
    function(x)names(table(x))[which.max(table(x))])
    acc.ens <- accuracy(test.y, majority.vote)*100
    sens.ens <- sensitivity(test.y, majority.vote)*100
    spec.ens <- specificity(test.y, majority.vote)*100

    perf <- c(acc.ens, sens.ens, spec.ens)
  }
  else if(method=="lasso")
  {
    fit.lasso.ind = lasso(as.matrix(reduced.trainx),
```

```
  reduced.trainy)
  lasso.idx.ind = which(abs(coef(fit.lasso.ind)[,1])>0)[-c(1)]
  lasso.m.idx.ind = match(names(lasso.idx.ind),colnames(test.x))
  lasso.xb.ind = as.matrix(cbind(1,as.matrix(test.x
  [,lasso.m.idx.ind])))%*%as.matrix(coef(fit.lasso.ind)
  [c(1,lasso.idx.ind),1])
  lasso.pred.ind = exp(lasso.xb.ind)/(1+exp(lasso.xb.ind))
  lasso.pred.ind <- ifelse(lasso.pred.ind>0.5, 1, 0)
  predicted.lasso.ind <- as.character(lasso.pred.ind[,1])

  acc.lasso <- accuracy(test.y, predicted.lasso.ind)*100
  sens.lasso <- sensitivity(test.y, predicted.lasso.ind)*100
  spec.lasso <- specificity(test.y, predicted.lasso.ind)*100

  perf <- c(acc.lasso, sens.lasso, spec.lasso)
}
else if(method=="alasso")
{
  fit.alasso.ind = alasso(as.matrix(reduced.trainx),
  reduced.trainy)
  alasso.idx.ind = which(abs(coef(fit.alasso.ind)[,1])>0)[-c(1)]
  alasso.m.idx.ind = match(names(alasso.idx.ind),colnames
  (test.x))
  alasso.xb.ind = as.matrix(cbind(1,as.matrix(test.x
  [,alasso.m.idx.ind])))%*%as.matrix(coef(fit.alasso.ind)
  [c(1,alasso.idx.ind),1])
  alasso.pred.ind = exp(alasso.xb.ind)/(1+exp(alasso.xb.ind))
  alasso.pred.ind <- ifelse(alasso.pred.ind>0.5, 1, 0)
```

```
  predicted.alasso.ind <- as.character(alasso.pred.ind[,1])

  acc.alasso <- accuracy(test.y, predicted.alasso.ind)*100
  sens.alasso <- sensitivity(test.y, predicted.alasso.ind)*100
  spec.alasso <- specificity(test.y, predicted.alasso.ind)*100

  perf <- c(acc.alasso, sens.alasso, spec.alasso)
}
else if(method=="bridge")
{
  fit.enet.ind = enet(as.matrix(reduced.trainx), reduced.trainy)
  enet.idx.ind = which(abs(coef(fit.enet.ind)[,1])>0)[-c(1)]
  enet.m.idx.ind = match(names(enet.idx.ind),colnames(test.x))
  enet.xb.ind = as.matrix(cbind(1,as.matrix(test.x
  [,enet.m.idx.ind])))%*%as.matrix(coef(fit.enet.ind)
  [c(1,enet.idx.ind),1])
  enet.pred.ind = exp(enet.xb.ind)/(1+exp(enet.xb.ind))
  enet.pred.ind <- ifelse(enet.pred.ind>0.5, 1, 0)
  predicted.enet.ind <- as.character(enet.pred.ind[,1])

  acc.enet <- accuracy(test.y, predicted.enet.ind)*100
  sens.enet <- sensitivity(test.y, predicted.enet.ind)*100
  spec.enet <- specificity(test.y, predicted.enet.ind)*100

  perf <- c(acc.enet, sens.enet, spec.enet)
}
else if(method=="scad")
{
```

```
  fit.scad.ind  = scad(as.matrix(reduced.trainx), reduced.trainy)
  scad.idx.ind = which(abs(coef(fit.scad.ind))>0)[-c(1)]
  scad.m.idx.ind = match(names(scad.idx.ind),colnames(test.x))
  scad.xb.ind = as.matrix(cbind(1,as.matrix(test.x
  [,scad.m.idx.ind])))%*%as.matrix(coef(fit.scad.ind)
  [c(1,scad.idx.ind)])
  scad.pred.ind = exp(scad.xb.ind)/(1+exp(scad.xb.ind))
  scad.pred.ind <- ifelse(scad.pred.ind>0.5, 1, 0)
  predicted.scad.ind <- as.character(scad.pred.ind[,1])

  acc.scad <- accuracy(test.y, predicted.scad.ind)*100
  sens.scad <- sensitivity(test.y, predicted.scad.ind)*100
  spec.scad <- specificity(test.y, predicted.scad.ind)*100

  perf <- c(acc.scad, sens.scad, spec.scad)
}
else if(method=="mcp")
{
  fit.mcp.ind  = mcp(as.matrix(reduced.trainx), reduced.trainy)
  mcp.idx.ind = which(abs(coef(fit.mcp.ind))>0)[-c(1)]
  mcp.m.idx.ind = match(names(mcp.idx.ind),colnames(test.x))
  mcp.xb.ind = as.matrix(cbind(1,as.matrix(test.x
  [,mcp.m.idx.ind])))%*%as.matrix(coef(fit.mcp.ind)
  [c(1,mcp.idx.ind)])
  mcp.pred.ind = exp(mcp.xb.ind)/(1+exp(mcp.xb.ind))
  mcp.pred.ind <- ifelse(mcp.pred.ind>0.5, 1, 0)
  predicted.mcp.ind <- as.character(mcp.pred.ind[,1])
```

```
  acc.mcp <- accuracy(test.y, predicted.mcp.ind)*100
  sens.mcp <- sensitivity(test.y, predicted.mcp.ind)*100
  spec.mcp <- specificity(test.y, predicted.mcp.ind)*100

  perf <- c(acc.mcp, sens.mcp, spec.mcp)
}
else if(method=="svm.lin")
{
  #--- Support Vector Machine linear -----
  modelsvm.lin <- svm(reduced.trainx, reduced.trainy,
  kernel = "linear")
  pred_classsvm.lin <- predict(modelsvm.lin, test.x)
  # From library caret to calculate performance metrics
  accuracy_svm.lin <- accuracy(test.y, pred_classsvm.lin)*100
  sensivitiy_svm.lin <- sensitivity(test.y, pred_classsvm.lin)
  *100
  specificity_svm.lin <- specificity(test.y, pred_classsvm.lin)
  *100

  perf <- c(accuracy_svm.lin, sensivitiy_svm.lin,
  specificity_svm.lin)
}
else if(method=="svm.rad")
{
  #-----Support Vector Machine radial --------
  modelsvm.rad <- svm(reduced.trainx, reduced.trainy,
  kernel = "radial")
  pred_classsvm.rad <- predict(modelsvm.rad, test.x)
```

```
  # From library caret to calculate performance metrics
  accuracy_svm.rad <- accuracy(test.y, pred_classsvm.rad)*100
  sensivitiy_svm.rad <- sensitivity
  (test.y, pred_classsvm.rad)*100
  specificity_svm.rad <- specificity
  (test.y, pred_classsvm.rad)*100

  perf <- c(accuracy_svm.rad, sensivitiy_svm.rad,
  specificity_svm.rad)
}
else if(method=="rf")
{
  #------------ Random Forest ----------
  rfmodel <- randomForest(reduced.trainx, reduced.trainy,
                          ntree= rftree)
  pred_classrf <- predict(rfmodel, test.x, type= "class")

  accuracy_rf <- accuracy(test.y, pred_classrf)*100
  sensitivity_rf <- sensitivity(test.y, pred_classrf)*100
  specificity_rf <- specificity(test.y, pred_classrf)*100

  perf <- c(accuracy_rf, sensitivity_rf, specificity_rf)
}
else if(method=="lr")
{
  #------- logistic regression ------------------
  glm_lr <- cv.glmnet(as.matrix(reduced.trainx), reduced.trainy,
                      family="binomial")
```

```
    pred_lr <- predict(glm_lr, newx = as.matrix(test.x),
                       s = "lambda.min", type = "class")

    accuracy_lr <- accuracy(test.y, pred_lr)*100
    sensitivity_lr <- sensitivity(test.y, pred_lr)*100
    specificity_lr <- specificity(test.y, pred_lr)*100

    perf <- c(accuracy_lr, sensitivity_lr, specificity_lr)
  }
  else if(method=="adab")
  {
    #------------------ Ada Boost
    model_adab <- boosting(reduced.trainy~., data = dat.train,
    boos=TRUE, mfinal= adatree)
    pred_adab <- predict(model_adab, as.data.frame(test.x))

    accuracy_adab <- accuracy(test.y, pred_adab$class)*100
    sensitivity_adab <- sensitivity(test.y, pred_adab$class)*100
    specificity_adab <- specificity(test.y, pred_adab$class)*100

    perf <- c(accuracy_adab, sensitivity_adab, specificity_adab)
  }
  return(perf)
}

#####---------- RUN ALL THE METHODS DEFINED ABOVE IN PARALLEL
start_time <- Sys.time()
cores <- 8
```

```
cl <- makeCluster(cores) #not to overload your computer
registerDoParallel(cl)

# run data generation function for specified number of reps
res.data <- foreach(i = 1:reps,
                    # .combine=list,
                    .packages=c("mvtnorm")) %dopar% {
                      # set.seed(i)
                      data.gen()
                      # cat("Iteration Data ", i, "\n")
                    }
# run feature selection resampling for specified number of reps
feat.resampling <- foreach(i = 1: reps,
                    .packages = c("glmnet","ncvreg")) %dopar% {
                    feature.selection("resampling", res.data[[i]])
                    # cat("Iteration resampling ", i, "\n")
                          }
# run feature selection information for specified number of reps
feat.information <- foreach(i = 1: reps,
                    .packages = c("Biocomb")) %dopar% {
                    feature.selection("information", res.data[[i]])
                          }
# run feature selection chi2 for specified number of reps
feat.chi2 <- foreach(i = 1: reps,
                     .packages = c("Biocomb")) %dopar% {
                      feature.selection("chi2", res.data[[i]])
                     }
# run feature selection mrmr for specified number of reps
```

```
feat.mrmr <- foreach(i = 1: reps,
                     .packages = c("praznik")) %dopar% {
                       feature.selection("mrmr", res.data[[i]])
                     }
# run allmethods on resampling features
res.resampling <- foreach(i=1:reps,
        .combine=rbind,
        .packages=c("class", "glmnet", "tidyverse",
        "randomForest", "glmnet", "ncvreg", "e1071",
        "caret", "RankAggreg", "adabag")) %dopar% {
        res.ensemble <- test.function(feat.resampling[[i]],
        method="ensemble")
        res.lasso <- test.function(feat.resampling[[i]],
        method="lasso")
        res.alasso <- test.function(feat.resampling[[i]],
        method="alasso")
        res.bridge <- test.function(feat.resampling[[i]],
        method="bridge")
        res.scad <- test.function(feat.resampling[[i]],
        method="scad")
        res.mcp <- test.function(feat.resampling[[i]],
        method="mcp")
        res.svm.lin <- test.function(feat.resampling[[i]],
        method="svm.lin")
        res.svm.rad <- test.function(feat.resampling[[i]],
        method="svm.rad")
        res.rf <- test.function(feat.resampling[[i]],
        method="rf")
```

```
        res.lr <- test.function(feat.resampling[[i]],
        method="lr")
        res.adab <- test.function(feat.resampling[[i]],
        method="adab")
        res.rep <- c(res.ensemble, res.lasso, res.alasso,
        res.bridge, res.scad, res.mcp, res.svm.lin,
        res.svm.rad, res.rf, res.lr, res.adab)
        res.rep
         }                                        }
# run all methods on information features
res.information <-foreach(i=1:reps,
        .combine=rbind,
        .packages=c("class", "glmnet", "tidyverse",
        "randomForest", "glmnet", "ncvreg", "e1071",
        "caret", "RankAggreg", "adabag")) %dopar% {
        res.ensemble <- test.function(feat.resampling[[i]],
        method="ensemble")
        res.lasso <- test.function(feat.resampling[[i]],
        method="lasso")
        res.alasso <- test.function(feat.resampling[[i]],
        method="alasso")
        res.bridge <- test.function(feat.resampling[[i]],
        method="bridge")
        res.scad <- test.function(feat.resampling[[i]],
        method="scad")
        res.mcp <- test.function(feat.resampling[[i]],
        method="mcp")
        res.svm.lin <- test.function(feat.resampling[[i]],
```

```
        method="svm.lin")
        res.svm.rad <- test.function(feat.resampling[[i]],
        method="svm.rad")
        res.rf <- test.function(feat.resampling[[i]],
        method="rf")
        res.lr <- test.function(feat.resampling[[i]],
        method="lr")
        res.adab <- test.function(feat.resampling[[i]],
        method="adab")
        res.rep <- c(res.ensemble, res.lasso, res.alasso,
        res.bridge, res.scad, res.mcp, res.svm.lin,
        res.svm.rad, res.rf, res.lr, res.adab)
        res.rep
        }
                                }
# run allmethods on chi2 features
res.chi2 <-foreach(i=1:reps,
        .combine=rbind,
        .packages=c("class", "glmnet", "tidyverse",
        "randomForest", "glmnet", "ncvreg", "e1071",
        "caret", "RankAggreg", "adabag")) %dopar% {
        res.ensemble <- test.function(feat.resampling[[i]],
        method="ensemble")
        res.lasso <- test.function(feat.resampling[[i]],
        method="lasso")
        res.alasso <- test.function(feat.resampling[[i]],
        method="alasso")
        res.bridge <- test.function(feat.resampling[[i]],
```

```
        method="bridge")
        res.scad <- test.function(feat.resampling[[i]],
        method="scad")
        res.mcp <- test.function(feat.resampling[[i]],
        method="mcp")
        res.svm.lin <- test.function(feat.resampling[[i]],
        method="svm.lin")
        res.svm.rad <- test.function(feat.resampling[[i]],
        method="svm.rad")
        res.rf <- test.function(feat.resampling[[i]],
        method="rf")
        res.lr <- test.function(feat.resampling[[i]],
        method="lr")
        res.adab <- test.function(feat.resampling[[i]],
        method="adab")
        res.rep <- c(res.ensemble, res.lasso, res.alasso,
        res.bridge, res.scad, res.mcp, res.svm.lin,
        res.svm.rad, res.rf, res.lr, res.adab)
        res.rep
        }
# run allmethods on mrmr features for
res.mrmr <- foreach(i=1:reps,
        .combine=rbind,
        .packages=c("class", "glmnet", "tidyverse",
        "randomForest", "glmnet", "ncvreg", "e1071",
        "caret", "RankAggreg", "adabag")) %dopar% {
        res.ensemble <- test.function(feat.resampling[[i]],
        method="ensemble")
```

```
        res.lasso <- test.function(feat.resampling[[i]],
        method="lasso")
        res.alasso <- test.function(feat.resampling[[i]],
        method="alasso")
        res.bridge <- test.function(feat.resampling[[i]],
        method="bridge")
        res.scad <- test.function(feat.resampling[[i]],
        method="scad")
        res.mcp <- test.function(feat.resampling[[i]],
        method="mcp")
        res.svm.lin <- test.function(feat.resampling[[i]],
        method="svm.lin")
        res.svm.rad <- test.function(feat.resampling[[i]],
        method="svm.rad")
        res.rf <- test.function(feat.resampling[[i]],
        method="rf")
        res.lr <- test.function(feat.resampling[[i]],
        method="lr")
        res.adab <- test.function(feat.resampling[[i]],
        method="adab")
        res.rep <- c(res.ensemble, res.lasso, res.alasso,
        res.bridge, res.scad, res.mcp, res.svm.lin,
        res.svm.rad, res.rf, res.lr, res.adab)
        res.rep
        }
stopCluster(cl)
end_time <- Sys.time()
total_time <- end_time - start_time
```

```
################################
#####--- Performance Metrics ---
################################

perf.metrics <- function(results)
{
  ens.acc <- results[,1]; lasso.acc <- results[,4];
  alasso.acc <- results[,7]; enet.acc <- results[,10];
  scad.acc <- results[,13];
  mcp.acc <- results[,16]; svm.lin.acc <- results[,19];
  svm.rad.acc <- results[,22]; rf.acc <- results[,25];
  lr.acc <- results[,28]; adab.acc <- results[,31];

  model <- data.frame(Methods = c(rep("ensemble", reps),
  rep("lasso", reps), rep("alasso", reps),
  rep("enet", reps), rep("scad", reps), rep("mcp", reps),
  rep("svm.lin", reps), rep("svm.rad", reps), rep("rf", reps),
  rep("lr", reps), rep("adab", reps)),
  Accuracy = c(ens.acc, lasso.acc, alasso.acc, enet.acc, scad.acc,
  mcp.acc, svm.lin.acc, svm.rad.acc, rf.acc, lr.acc, adab.acc))


  ensemble.accuracy <- c(mean(results[,1]), sd(results[,1]))
  ensemble.sensitivity <- c(mean(results[,2]), sd(results[,2]))
  ensemble.specificity <- c(mean(results[,3]), sd(results[,3]))
  lasso.accuracy <- c(mean(results[,4]), sd(results[,4]))
  lasso.sensitivity <- c(mean(results[,5]), sd(results[,5]))
  lasso.specificity <- c(mean(results[,6]), sd(results[,6]))
```

```
alasso.accuracy <- c(mean(results[,7]), sd(results[,7]))
alasso.sensitivity <- c(mean(results[,8]), sd(results[,8]))
alasso.specificity <- c(mean(results[,9]), sd(results[,9]))
enet.accuracy <- c(mean(results[,10]), sd(results[,10]))
enet.sensitivity <- c(mean(results[,11]), sd(results[,11]))
enet.specificity <- c(mean(results[,12]), sd(results[,12]))
scad.accuracy <- c(mean(results[,13]), sd(results[,13]))
scad.sensitivity <- c(mean(results[,14]), sd(results[,14]))
scad.specificity <- c(mean(results[,15]), sd(results[,15]))
mcp.accuracy <- c(mean(results[,16]), sd(results[,16]))
mcp.sensitivity <- c(mean(results[,17]), sd(results[,17]))
mcp.specificity <- c(mean(results[,18]), sd(results[,18]))
svm.lin.accuracy <- c(mean(results[,19]), sd(results[,19]))
svm.lin.sensitivity <- c(mean(results[,20]), sd(results[,20]))
svm.lin.specificity <- c(mean(results[,21]), sd(results[,21]))
svm.rad.accuracy <- c(mean(results[,22]), sd(results[,22]))
svm.rad.sensitivity <- c(mean(results[,23]), sd(results[,23]))
svm.rad.specificity <- c(mean(results[,24]), sd(results[,24]))
rf.accuracy <- c(mean(results[,25]), sd(results[,25]))
rf.sensitivity <- c(mean(results[,26]), sd(results[,26]))
rf.specificity <- c(mean(results[,27]), sd(results[,27]))
lr.accuracy <- c(mean(results[,28]), sd(results[,28]))
lr.sensitivity <- c(mean(results[,29]), sd(results[,29]))
lr.specificity <- c(mean(results[,30]), sd(results[,30]))
adab.accuracy <- c(mean(results[,31]), sd(results[,31]))
adab.sensitivity <- c(mean(results[,32]), sd(results[,32]))
adab.specificity <- c(mean(results[,33]), sd(results[,33]))
```

```
ensemble.performance <- list(ensemble.accuracy,
ensemble.sensitivity, ensemble.specificity)
names(ensemble.performance) <- c("ensemble accuracy",
"ensemble sensitivity","ensemble specificity")
lasso.performance <- list(lasso.accuracy,
lasso.sensitivity, lasso.specificity)
names(lasso.performance) <- c("lasso accuracy",
"lasso sensitivity", "lasso specificity")
alasso.performance <- list(alasso.accuracy,
alasso.sensitivity, alasso.specificity)
names(alasso.performance) <- c("alasso accuracy",
"alasso sensitivity", "alasso specificity")
enet.performance <- list(enet.accuracy,
enet.sensitivity, enet.specificity)
names(enet.performance) <- c("enet accuracy",
"enet sensitivity","enet specificity")
scad.performance <- list(scad.accuracy,
scad.sensitivity, scad.specificity)
names(scad.performance) <- c("scad accuracy",
"scad sensitivity","scad specificity")
mcp.performance <- list(mcp.accuracy,
mcp.sensitivity, mcp.specificity)
names(mcp.performance) <- c("mcp accuracy",
"mcp sensitivity","mcp specificity")
svm.lin.performance <- list(svm.lin.accuracy,
svm.lin.sensitivity, svm.lin.specificity)
names(svm.lin.performance) <- c("svm.lin accuracy",
"svm.lin sensitivity","svm.lin specificity")
```

```
  svm.rad.performance <- list(svm.rad.accuracy,
  svm.rad.sensitivity, svm.rad.specificity)
  names(svm.rad.performance) <- c("svm.rad accuracy",
  "svm.rad sensitivity","svm.rad specificity")
  rf.performance <- list(rf.accuracy,
  rf.sensitivity, rf.specificity)
  names(rf.performance) <- c("rf accuracy",
  "rf sensitivity","rf specificity")
  lr.performance <- list(lr.accuracy,
  lr.sensitivity, lr.specificity)
  names(lr.performance) <- c("lr accuracy",
  "lr sensitivity","lr specificity")
  adab.performance <- list(adab.accuracy,
  adab.sensitivity, adab.specificity)
  names(adab.performance) <- c("adab accuracy",
  "adab sensitivity","adab specificity")

  return(list(ensemble.performance, lasso.performance,
  alasso.performance, enet.performance, scad.performance,
                mcp.performance, svm.lin.performance,
                svm.rad.performance, rf.performance,
                lr.performance, adab.performance,
                model))
}
resampling.metrics <- perf.metrics(res.resampling)
information.metrics <- perf.metrics(res.information)
chi2.metrics <- perf.metrics(res.chi2)
mrmr.metrics <- perf.metrics(res.mrmr)
```

```
##### BOXPLOTS

resample.match <- information.match <- fscore.match <-
mrmr.match <- chi2.match <- list()

for (i in seq_along(feat.resampling))
{
  resample.match[[i]] <-
  match(feat.resampling[[i]][[3]], true.imp)
}
for (i in seq_along(feat.information))
{
  information.match[[i]] <-
  match(feat.information[[i]][[3]], true.imp)
}

for (i in seq_along(feat.chi2))
{
  chi2.match[[i]] <- match(feat.chi2[[i]][[3]], true.imp)
}

for (i in seq_along(feat.mrmr))
{
  mrmr.match[[i]]     <- match(feat.mrmr[[i]][[3]], true.imp)
}

resample.len <- information.len <- fscore.len <-
```

```
mrmr.len <- chi2.len <- vector()
for(i in seq_along(resample.match))
{
  resample.len[i]     <- length(resample.match[[i]]
  [!is.na(resample.match[[i]])])/length(true.imp)*100
}

for (i in seq_along(information.match))
{
  information.len[i] <- length(information.match[[i]]
  [!is.na(information.match[[i]])])/length(true.imp)*100
}

for (i in seq_along(chi2.match))
{
  chi2.len[i]          <- length(chi2.match[[i]]
  [!is.na(chi2.match[[i]])])/length(true.imp)*100
}

for (i in seq_along(mrmr.match))
{
  mrmr.len[i]          <- length(mrmr.match[[i]]
  [!is.na(mrmr.match[[i]])])/length(true.imp)*100
}

resample.sel.avg <- sum(resample.len)/length(resample.len)
resample.sel.sd <- sd(resample.len)
information.sel.avg <- sum(information.len)/
```

```
length(information.len)
information.sel.sd <- sd(information.len)
chi2.sel.avg <- sum(chi2.len)/length(chi2.len)
chi2.sel.sd <- sd(chi2.len)
mrmr.sel.avg <- sum(mrmr.len)/length(mrmr.len)
mrmr.sel.sd <- sd(mrmr.len)

output1 <- list(resample.sel.avg, resample.sel.sd,
                information.sel.avg, information.sel.sd,
                chi2.sel.avg, chi2.sel.sd,
                mrmr.sel.avg, mrmr.sel.sd
)
names(output1) <- list("resample.sel.avg", "resample.sel.sd",
                       "information.sel.avg",
                       "information.sel.sd",
                       "chi2.sel.avg", "chi2.sel.sd",
                       "mrmr.sel.avg", "mrmr.sel.sd"
)

sink('res_sim_scenario_05.txt')

print("-------TOTAL TIME----------")
print(total_time)
print("-------selection probability ----------")
print(output1)
print("-------------------------- resampling ----------------------")
print("----------- Overall Accuracy and SD----------")
print(resampling.metrics)
```

```
print("------------------------ information gain ----------")
print("---------- Overall Accuracy and SD----------")
print(information.metrics)

print("------------------------ chi2 ----------------------")
print("---------- Overall Accuracy and SD----------")
print(chi2.metrics)

print("------------------------ mrmr ----------------------")
print("---------- Overall Accuracy and SD----------")
print(mrmr.metrics)

sink()

model.prob <- data.frame(Methods = c(rep("resampling", reps),
rep("information", reps),
rep("chi2", reps), rep("mRMR", reps)),
 Accuracy = c(resample.len, information.len,
 fscore.len, mrmr.len, chi2.len))

level_order.prob <- c('resampling', 'information',
'chi2', 'mRMR')
# save the box plot for selection probability
ggsave("bplot.sel.prob.png", ggplot(model.prob, aes(x= factor
(Methods, level = level_order.prob) ,
y = Accuracy), fill= Methods)
+ coord_cartesian(ylim = c(0, 50)) +
```

```
        geom_boxplot(fill=c("red","green","blue","pink"),
        alpha=0.5, outlier.colour = "red",
        outlier.size = 0.5, fatten = 1) +
        ggtitle("True variable selection average:
          correlation : 0.5") + xlab("Methods") +
        ylab("Accuracy (in %)") + theme(plot.title =
        element_text(face = "bold" ,hjust = 0.5, size= 8)
        ,axis.text.x = element_text(face = "bold",angle = 90,
        size = 8, hjust = 1)
        ,axis.text.y = element_text(face = "bold",angle = 0,
        size = 8, hjust =1)
        ,legend.position="none"
        ,axis.title.y =
        element_text(face = "bold", size = rel(0.7))
        ,axis.title.x =
        element_text(face = "bold", size = rel(0.7))),
      width = 3, height = 3, dpi = 1200,
      units = "in", device='png')

#####-------------        Set parameters for box plots  ----
level_order <- c('ensemble', 'lasso', 'alasso', 'enet', 'scad',
'mcp', 'rf', 'svm.lin', 'svm.rad', 'adab', 'lr')
box.col <- c("orange","green","blue","red","yellow","turquoise",
"purple", "pink", "grey", "black", "brown")
box.range <- c(60, 100)


##############################################
```

```
# save the box plot for resampling
ggsave("bplot.resampling.png", ggplot(resampling.metrics[[12]],
aes(x= factor(Methods, level = level_order) , y = Accuracy),
fill= Methods) + coord_cartesian(ylim = box.range) +
          geom_boxplot(fill= box.col, alpha=0.5,
          outlier.colour = "red",
          outlier.size = 0.5, fatten = 1) +
          ggtitle("Correlation 0.5: resampling") +
          xlab("Methods")
          + ylab("Accuracy (in %)") +
          theme(plot.title = element_text(face = "bold" ,
          hjust = 0.5, size= 8)
          ,axis.text.x = element_text(face = "bold",angle = 90,
          size = 8, hjust = 1)
          ,axis.text.y = element_text(face = "bold",angle = 0,
          size = 8, hjust =1)
          ,legend.position="none"
          ,axis.title.y =
          element_text(face = "bold", size = rel(0.7))
          ,axis.title.x =
          element_text(face = "bold", size = rel(0.7))
          ),
        width = 3, height = 3, dpi = 1200,
        units = "in", device='png')


# save the box plot for information
ggsave("bplot.information.png", ggplot(information.metrics[[12]],
```

```
aes(x= factor(Methods, level = level_order) , y = Accuracy),
fill= Methods) + coord_cartesian(ylim = box.range) +
          geom_boxplot(fill= box.col, alpha=0.5,
          outlier.colour = "red",
          outlier.size = 0.5, fatten = 1) +
          ggtitle("Correlation 0.5: information")
          + xlab("Methods") +
          ylab("Accuracy (in %)") + theme(plot.title =
          element_text(face = "bold" ,hjust = 0.5, size= 8)
,axis.text.x = element_text(face = "bold",angle = 90,
          size = 8, hjust = 1)
          ,axis.text.y = element_text(face = "bold",angle = 0,
          size = 8, hjust =1)
          ,legend.position="none"
          ,axis.title.y =
          element_text(face = "bold", size = rel(0.7))
          ,axis.title.x =
          element_text(face = "bold", size = rel(0.7))
          ),
        width = 3, height = 3, dpi = 1200,
        units = "in", device='png')


# save the box plot for chi2
ggsave("bplot.chi2.png", ggplot(chi2.metrics[[12]],
aes(x= factor(Methods, level = level_order) , y = Accuracy),
fill= Methods) + coord_cartesian(ylim = box.range) +
          geom_boxplot(fill= box.col, alpha=0.5,
```

```
        outlier.colour = "red", outlier.size = 0.5, fatten = 1)
        +
        ggtitle("Correlation 0.5: chi2") + xlab("Methods") +
        ylab("Accuracy (in %)") + theme(plot.title =
        element_text(face = "bold" ,hjust = 0.5, size= 8)
        ,axis.text.x = element_text(face = "bold",angle = 90,
        size = 8, hjust = 1)
        ,axis.text.y = element_text(face = "bold",angle = 0,
        size = 8, hjust =1)
        ,legend.position="none"
        ,axis.title.y =
        element_text(face = "bold", size = rel(0.7))
        ,axis.title.x =
        element_text(face = "bold", size = rel(0.7))
        ),
      width = 3, height = 3, dpi = 1200,
      units = "in", device='png')

# save the box plot for mrmr
ggsave("bplot.mrmr.png", ggplot(mrmr.metrics[[12]],
aes(x= factor(Methods, level = level_order) , y = Accuracy),
fill= Methods) + coord_cartesian(ylim = box.range) +
                  geom_boxplot(fill= box.col, alpha=0.5,
        outlier.colour = "red", outlier.size = 0.5, fatten = 1)
        +
        ggtitle("Correlation 0.5: chi2") + xlab("Methods") +
        ylab("Accuracy (in %)") + theme(plot.title =
        element_text(face = "bold" ,hjust = 0.5, size= 8)
```

```
        ,axis.text.x = element_text(face = "bold",angle = 90,
        size = 8, hjust = 1)
        ,axis.text.y = element_text(face = "bold",angle = 0,
        size = 8, hjust =1)
        ,legend.position="none"
        ,axis.title.y =
        element_text(face = "bold", size = rel(0.7))
        ,axis.title.x =
        element_text(face = "bold", size = rel(0.7))
        ),
      width = 3, height = 3, dpi = 1200,
      units = "in", device='png')

##############
#####----------SAVE THE RESULTS
##############
saveRDS(output1, file = "100iter_top75_sel_avg.RData")
saveRDS(res.data, file="Data_05.rds")
saveRDS(feat.resampling, file="feat_resampling.rds")
saveRDS(feat.information, file="feat_information.rds")
saveRDS(feat.chi2, file="feat_chi2.rds")
saveRDS(feat.mrmr, file="feat_mrmr.rds")

saveRDS(res.resampling, file="res_resampling.rds")
saveRDS(res.information, file="res_information.rds")
saveRDS(res.chi2, file="res_chi2.rds")
saveRDS(res.mrmr, file="res_mrmr.rds")
#########################################################
```

Appendix B

Appendix B: Adaptive lasso with normalized filtering scores

```
setwd("/export/home/arpatil/Desktop/modalasso/last/ar105")
library(glmnet)
library(ncvreg)
library(R.matlab)
library(praznik)
library(PredPsych)
library(Biocomb)
library(mvtnorm)



####
##---------- performance metrics--------------------##
######

max.iter <- 100

accuracy <- function(truth, predicted)
{
  if(length(truth) > 0)
    sum(truth==predicted)/length(truth) else
```

```
        return(0)
}

sensitivity <- function(truth, predicted)
{
  if(sum(truth==1) > 0)
    sum(predicted[truth==1]==1)/sum(truth==1)    else
        return(0)
}

specificity <- function(truth, predicted)
{
  if(sum(truth==0) > 0)
    sum(predicted[truth==0]==0)/sum(truth==0)    else
        return(0)
}

AUC <- function(truth, probs, plot=FALSE){
  # probs - probability of class 1
  q <- seq(0, 1, .01)
  sens <- rep(0, length(q))
  spec <- rep(0, length(q))
  ly <- levels(truth)

  for(i in 1:length(q)){
    pred <- probs >= q[i]
    pred[pred] <- 1
    #pred <- factor(pred, levels=ly)
```

```
    sens[i] <- sensitivity(truth, pred)
    spec[i] <- specificity(truth, pred)
  }

  # make sure it starts and ends at 0, 1
  sens <- c(1, sens, 0)
  spec <- c(0, spec, 1)

  trap.rule <- function(x,y) sum(diff(x)*(y[-1]+y[-length(y)]))/2
  auc <- trap.rule(rev(1-spec), rev(sens))

  if(plot){
    plot(1-spec, sens, type="l", xlab="1-Specificity",
         ylab="Sensitivity", main="ROC Curve")
    legend("bottomright", legend=paste("AUC = ", round(auc, 3)),
           bty="n")
  }
  auc
}

## Simulation data
data.gen <- function(seed.index)
{
  model = function(X)
  {
    return(exp(X)/(1+exp(X)))
  }
```

```
multivariate_normal_sampler = function(cov0,p,nsample)
{
  dat = NULL
  for(i in 1:p)
  {
    L = chol(cov0)
    Z = matrix(rnorm(nsample*ncol(cov0)),
    nrow=nsample,ncol=ncol(cov0))
    X = Z%*%L
    dat = cbind(dat,X)
  }
  return(dat)
}
# cov matrix
cov.mat = function(block,rho)
{
  tmp.cov = matrix(0, nrow =block, ncol=block)
  for(i in 1:block)
  {
    for(j in 1:block)
    {
      tmp.cov[i,j] = rho^(abs(i-j))
    }
  }
  return(tmp.cov)
}
# one single block AR(1)
feature_set = function(nsample, p, rho)
```

```
  {
    block   = c(1)
    dat = NULL
    for(i in block)
    {
      if(i==1)
      {
        cov0      = cov.mat(block,rho)
        tmp.dat   = multivariate_normal_sampler(cov0, p, nsample)
        tmp.dat   = model(tmp.dat)
      }
      dat       = cbind(dat,tmp.dat)
    }
    return(dat)
  }

  seeds <- seed.index
  dat = feature_set(nsample= 200, p = 1000, rho = 0.5)
  dat = scale(dat)
  colnames(dat)=paste("v",1:ncol(dat),sep="")

  return(dat)
}

data.generator <- function(dat)
{
  x1 <- dat
  ## weak
```

```
  beta <- c(1.036478, -1.073296,
  -1.250946, 1.138729, -1.128361, 1.145263)
  tiv <- sample(colnames((x1)), replace= FALSE)[1:6]
  ## generate y
  pi = (exp(x1[,tiv]%*%beta))/(1+exp(x1[,tiv]%*%beta))
  y1  = rbinom(nrow(x1),size=1,prob=pi)

  s.idx <- sample(1:nrow(x1),as.integer(nrow(x1)*.7),replace = F)
  train.x <- x1[s.idx,]; train.y <- y1[s.idx];
  test.x <-  x1[-s.idx,]; test.y <-  y1[-s.idx];

  return(list(train.x, test.x, train.y, test.y, tiv))
}

###########
## real data
# [Use either simulation or real data.gen() function each time]
data.gen <- function(seeds)
{
  x1 = read.csv("colon_2000_62_x.csv")
  y1 = read.csv("colon_2000_62_y.csv")[,1]
  n1 <- ceiling(nrow(x1)/log(nrow(x1)))
  s.idx <- sample(1:nrow(x1),as.integer(nrow(x1)*.7),replace = F)
  train.x <- x1[s.idx,]; train.y <- y1[s.idx];
  test.x <-  x1[-s.idx,]; test.y <-  y1[-s.idx];

  return(list(train.x, test.x, train.y, test.y, n1))
}
```

```
feature.selection = function(method, data)
{
  train.x <- data[[1]]; test.x <- data[[2]];
  train.y <- data[[3]]; test.y <- data[[4]];
  tiv <- data[[5]];

  n1 <- 38
  ## n1 <- ceiling(nrow(rbind(train.x, test.x))
  /log(nrow(rbind(train.x, test.x))))
  data.train <- data.frame(train.x, train.y)
  data.test <- data.frame(test.x, test.y)
  set <- list()

  ## fscore
  if(method=="fscore")
  {
    data.train[,ncol(data.train)]<-as.numeric
    (data.train[,ncol(data.train)])
    f.score <- fscore(data.train, classCol = ncol(data.train))
    sfs <- sort(f.score,decreasing = TRUE)[-1]
    set1 <- sfs[1:n1]
    set1 <- abs(set1)/mean(abs(set1))
    set <- list(set1)
  }

  ## information gain
  if(method=="information")
```

```
{
  # class label must be factor
  data.train[,ncol(data.train)]<-as.factor
  (data.train[,ncol(data.train)])
  disc<-"equal interval width"
  attrs.nominal <- numeric()
  fit.info = select.inf.gain(data.train,disc.method=disc,
              attrs.nominal=attrs.nominal)

  sfs <- fit.info$Information.Gain
  fb <- as.character(fit.info$Biomarker)
  names(sfs) <- fb
  set1 <- sfs[1:n1]
  set1 <- abs(set1)/mean(abs(set1))

  set <- list(set1)
}

## chi2
if(method=="chi2")
{
  # class label must be factor
  data.train[,ncol(data.train)]<-as.factor
  (data.train[,ncol(data.train)])
  disc<-"equal interval width"
  attrs.nominal <- numeric()
  fit.chi2 <- select.inf.chi2(data.train,disc.method=disc,
              attrs.nominal=attrs.nominal)
```

```
  sfs <- fit.chi2$ChiSquare
  fb <- as.character(fit.chi2$Biomarker)
  names(sfs) <- fb
  set1 <- sfs[1:n1]
  set1 <- abs(set1)/mean(abs(set1))

  set <- list(set1)
}
## mmle
if(method=="mmle")
{
  ## resampling
  rc.vec <- NULL
  se.vec <- NULL

  for(i in 1:ncol(train.x))
  {
    fit = glm(train.y ~ train.x[,i],family="binomial")
    rc.vec <- c(rc.vec, as.numeric
    (summary(fit)[12][[1]][,1][-1]))
    se.vec <- c(se.vec, as.numeric
    (summary(fit)[12][[1]][,2][-1]))
  }
  names(rc.vec) <- colnames(train.x)
  names(se.vec) <- colnames(train.x)
  rc.vec1 <- sort(abs(rc.vec), decreasing = TRUE)
  se.vec1 <- match(names(rc.vec1), names(se.vec))
```

```
    se.vec2 <- se.vec[se.vec1]
    rm(rc.vec, se.vec, se.vec1);
    sett <- abs(rc.vec1) / se.vec2;

    set1 <- sett[1:n1]
    set <- list(set1)
  }

  ## ridge
  if(method=="ridge")
  { # type.measure defaults to partial likelihood for cox model
    ridge1_cv <- cv.glmnet(as.matrix(train.x), train.y,
    alpha = 0, family = "binomial", standardize = F)
    ## The intercept estimate should be dropped.
    best_ridge_coef <- coef(ridge1_cv, s = ridge1_cv$lambda.min)
    best_ridge_coef <- abs(best_ridge_coef[,1][-1])
    set1 <- sort(best_ridge_coef, decreasing = TRUE)[1:n1]

    set <- list(set1)
  }
  ## return set of features
  return(list(set, train.x, train.y, test.x, test.y, tiv))
}

#########
##------------ code for penalized logistic regression method
##------------- lasso -------
lasso = function(x,y)
```

```
{
  cv.lasso <- cv.glmnet(y = y, x = as.matrix(x), alpha=1,
                        family="binomial")
  return(cv.lasso)
}

##---------- Proposed Modified Adaptive lasso ----
mod.alasso = function(x, y, weights)
{
  cv.mod.alasso <- glmnet::cv.glmnet(y = y, x = as.matrix(x),
  alpha=1,penalty.factor = 1/abs(weights),family="binomial")
  return(cv.mod.alasso)
}

##----alasso with ridge----
alasso = function(x, y)
{
  #ridge regression weights
  # type.measure defaults to partial likelihood for cox model
  ridge1_cv <- cv.glmnet(as.matrix(x), y, alpha = 0,
  family = "binomial")
  ## The intercept estimate should be dropped.
  best_ridge_coef <- coef(ridge1_cv, s = ridge1_cv$lambda.min)
  cv.alasso <- glmnet::cv.glmnet(as.matrix(x), y, alpha=1,
                penalty.factor = 1/abs(best_ridge_coef[,1][-1]),
                family="binomial")
  return(cv.alasso)
}
```

```
##--------alasso for mmle weights-------
alasso_mmle = function(x, y, set_mmle)
{
  cv.alasso <- glmnet::cv.glmnet(as.matrix(x), y, alpha=1,
                                 penalty.factor = 1/abs(set_mmle),
                                 family="binomial")
  return(cv.alasso)
}
################


acc.mod.alasso <- auc.mod.alasso <-
gmean.mod.alasso <- vector()
tpr_tiv <- vector()
tpr_mod_alasso <- tpr_lasso <- vector()
tpr_mod <- list()
#########

dat<- function(method, data)
{
  fs <- feature.selection(method, data)

  wi <- fs[[1]]
  train.x <- fs[[2]]
  train.y <- fs[[3]]
  test.x <- fs[[4]]
  test.y <- fs[[5]]
  tiv <- fs[[6]]
```

```
for (i in seq(wi))
{
  ## For feature selection method
  tpr_t <- match(names(wi[[i]]), tiv)
  tpr_tiv[i] <- length(tpr_t[!is.na(tpr_t)])

  red.train.x <- train.x[, names(wi[[i]])]
  red.test.x <- test.x[, names(wi[[i]])]

  mod.train.x <- red.train.x / wi[[i]][col(red.train.x)]
  mod.train.x <- mod.train.x[ , apply(mod.train.x, 2,
                  function(x) !any(is.na(x)))]

  mod.test.x <- red.test.x / wi[[i]][col(red.test.x)]
  mod.test.x <- mod.test.x[ , apply(mod.test.x, 2,
                  function(x) !any(is.na(x)))]

  ##############
  ##------ Modified Adaptive Lasso--
  ##############
  mod.alasso_cv <- mod.alasso(as.matrix(mod.train.x),
  train.y, wi[[i]])
  mod.alasso.idx = which(abs(coef(mod.alasso_cv)[,1])>0)
  [-c(1)]
  mod.alasso.m.idx = match(names(mod.alasso.idx),
  colnames(mod.test.x))
  mod.alasso.xb =
```

```
  as.matrix(cbind(1,as.matrix(mod.test.x
  [,mod.alasso.m.idx])))
  %*%as.matrix(coef(mod.alasso_cv)
  [c(1,mod.alasso.idx),1])
  mod.alasso.pred = exp(mod.alasso.xb)/
  (1+exp(mod.alasso.xb))
  mod.alasso.pred <- ifelse(mod.alasso.pred>0.5, 1, 0)
  mod.predicted.alasso <- as.character(
  mod.alasso.pred[,1])

  ## For modified adaptive lasso
  tpr_mod <-  match(names(mod.alasso.idx), tiv)
  tpr_mod_alasso[i] <- length(tpr_mod[!is.na(tpr_mod)])

  acc.mod.alasso[i] <- accuracy(test.y,
  mod.predicted.alasso)
  sens.mod.alasso <- sensitivity(test.y,
  mod.predicted.alasso)
  spec.mod.alasso <- specificity(test.y,
  mod.predicted.alasso)
  auc.mod.alasso[i] <- AUC(as.factor(test.y),
  mod.predicted.alasso)
  gmean.mod.alasso[i] <-
  sqrt(sens.mod.alasso*spec.mod.alasso)
  #########
}
return(list(acc.mod.alasso,
            auc.mod.alasso,
```

```
                gmean.mod.alasso,
                tpr_tiv, tpr_mod_alasso))

}
acc.alasso_unknown <- auc.alasso_unknown <- vector()
gmean.alasso_unknown <- tpr_ttiv <-
tpr_unknown <- vector()

dat_unknown<- function(method, data)
{
  fs <- feature.selection(method, data)

  wi <- fs[[1]]
  train.x <- fs[[2]]
  train.y <- fs[[3]]
  test.x <- fs[[4]]
  test.y <- fs[[5]]
  tiv <- fs[[6]]

  for (i in seq(wi))
  {
    tpr_t <- match(names(wi[[i]]), tiv)
    tpr_ttiv[i] <- length(tpr_t[!is.na(tpr_t)])

    red.train.x <- train.x[, names(wi[[i]])]
    red.test.x <- test.x[, names(wi[[i]])]

    ##############################################
```

```
##Adaptive Lasso with ridge or mmle weights
fit.alasso_unknown = alasso_mmle(red.train.x,
train.y, wi[[i]])
alasso_unknown.idx = which(abs(coef(fit.alasso_unknown)
                          [,1])>0)[-c(1)]
alasso_unknown.m.idx = match(names(alasso_unknown.idx),
                            colnames(red.test.x))
alasso_unknown.xb = as.matrix(cbind(1,as.matrix
                        (red.test.x[,alasso_unknown.m.idx])))
%*%as.matrix(coef(fit.alasso_unknown)
[c(1,alasso_unknown.idx),1])
alasso_unknown.pred = exp(alasso_unknown.xb)/
                          (1+exp(alasso_unknown.xb))
alasso_unknown.pred <- ifelse
(alasso_unknown.pred>0.5, 1, 0)
predicted.alasso_unknown <- as.character
(alasso_unknown.pred[,1])

## For ridge or mmle
tpr_rid <-  match(names(alasso_unknown.idx), tiv)
tpr_unknown[i] <- length(tpr_rid[!is.na(tpr_rid)])

acc.alasso_unknown[i] <- accuracy(test.y,
predicted.alasso_unknown)
sens.alasso_unknown <- sensitivity(test.y,
predicted.alasso_unknown)
spec.alasso_unknown <- specificity(test.y,
predicted.alasso_unknown)
```

```
    auc.alasso_unknown[i] <- AUC(as.factor(test.y),
                                 predicted.alasso_unknown)
    gmean.alasso_unknown[i] <- sqrt(sens.alasso_unknown*
                                 spec.alasso_unknown)

  }
  return(list(acc.alasso_unknown,
              auc.alasso_unknown,
              gmean.alasso_unknown,
              tpr_ttiv, tpr_unknown))

}

tpr_lasso <- tpr_alasso <- vector()
acc.lasso <- auc.lasso <- gmean.lasso <- vector()
acc.alasso <- auc.alasso <- gmean.alasso <- vector()

dat_unfilter<- function(data)
{
  train.x <- data[[1]]
  test.x <- data[[2]]
  train.y <- data[[3]]
  test.y <- data[[4]]
  tiv <- data[[5]]
  for (i in 1)
  {
    ############
    ##------------------------------------ Lasso----------
```

```
#############
fit.lasso = lasso(as.matrix(train.x), train.y)
lasso.idx = which(abs(coef(fit.lasso)[,1])>0)[-c(1)]
lasso.m.idx = match(names(lasso.idx),colnames(test.x))
lasso.xb = as.matrix(cbind(1,as.matrix(test.x
[,lasso.m.idx])))
%*%as.matrix(coef(fit.lasso)[c(1,lasso.idx),1])
lasso.pred = exp(lasso.xb)/(1+exp(lasso.xb))
lasso.pred <- ifelse(lasso.pred>0.5, 1, 0)
predicted.lasso <- as.character(lasso.pred[,1])

## For lasso
tpr_las <-  match(names(lasso.idx), tiv)
tpr_lasso[i] <- length(tpr_las[!is.na(tpr_las)])

acc.lasso[i] <- accuracy(test.y, predicted.lasso)
sens.lasso <- sensitivity(test.y, predicted.lasso)
spec.lasso <- specificity(test.y, predicted.lasso)
auc.lasso[i] <- AUC(as.factor(test.y),predicted.lasso)
gmean.lasso[i] <- sqrt(sens.lasso*spec.lasso)

###############
##------------------------------------ Adaptive Lasso-------
fit.alasso = alasso(as.matrix(train.x), train.y)
alasso.idx = which(abs(coef(fit.alasso)[,1])>0)[-c(1)]
alasso.m.idx = match(names(alasso.idx),colnames(test.x))
alasso.xb = as.matrix(cbind(1,as.matrix(
test.x[,alasso.m.idx])))
```

```
    %*%as.matrix(coef(fit.alasso)[c(1,alasso.idx),1])
    alasso.pred = exp(alasso.xb)/(1+exp(alasso.xb))
    alasso.pred <- ifelse(alasso.pred>0.5, 1, 0)
    predicted.alasso <- as.character(alasso.pred[,1])

    ## For alasso
    tpr_alas <-  match(names(alasso.idx), tiv)
    tpr_alasso[i] <- length(tpr_alas[!is.na(tpr_alas)])

    acc.alasso[i] <- accuracy(test.y, predicted.alasso)
    sens.alasso <- sensitivity(test.y, predicted.alasso)
    spec.alasso <- specificity(test.y, predicted.alasso)
    auc.alasso[i] <- AUC(as.factor(test.y),predicted.alasso)
    gmean.alasso[i] <- sqrt(sens.alasso*spec.alasso)
  }

  return(list(acc.lasso,
              auc.lasso,
              gmean.lasso, tpr_lasso,
              acc.alasso,
              auc.alasso,
              gmean.alasso, tpr_alasso))

}

fscore.output <- information.output <- chi2.output <- list()
lasso.output <- ridge.output <- mmle.output <- list()
```

```
## Main function
main.fun <- function()
{
  seeds = set.seed(sample(1:100000,1));
  dat <- data.gen(seeds)

  # Call all the functions
  for (i in  1: max.iter)
  {
    data <- data.generator(dat)
    ## Case 1
    fscore.output[[i]] <- dat("fscore", data)
    ## Case 2
    information.output[[i]] <- dat("information", data)
    ## Case 3
    chi2.output[[i]] <- dat("chi2", data)
    ## Case 4
    ## without filtering lasso and alasso
    lasso.output[[i]] <- dat_unfilter(data)
    ## Case 5
    ## with ridge weights
    ridge.output[[i]] <- dat_unknown("ridge", data)
    ## Case 6
    ## with mmle weights
    mmle.output[[i]] <- dat_unknown("mmle", data)

    # some output
    verbose=T
```

```
    if(verbose)
      cat("Iteration Number ", i, "\n")
  }
  return(list(fscore.output, information.output, chi2.output,
        lasso.output, ridge.output, mmle.output))
}

## call the entire function
output <- main.fun()

# Function to calculate the performance metrics
perf.fun <- function(op)
{
  dff <- op
  acc.mod.alasso <- auc.mod.alasso <- gmean.mod.alasso <- list()
  tpr_tiv <- tpr_mod_alasso <- list()

  for (i in seq_along(dff))
  {
    acc.mod.alasso[[i]] <- dff[[i]][[1]]
    auc.mod.alasso[[i]] <- dff[[i]][[2]]
    gmean.mod.alasso[[i]] <- dff[[i]][[3]]
    tpr_tiv[[i]] <- dff[[i]][[4]]
    tpr_mod_alasso[[i]] <- dff[[i]][[5]]
  }

  tpr_tiv.n1 <- vector()
  for (i in seq(tpr_tiv))
```

```
{
  tpr_tiv.n1[i] <- tpr_tiv[[i]][[1]]
}
avg.tpr_tiv.n1 <- mean(tpr_tiv.n1);
sd.tpr_tiv.n1 <- sd(tpr_tiv.n1);

tpr_mod_alasso.n1 <- vector()
for (i in seq(tpr_mod_alasso))
{
  tpr_mod_alasso.n1[i] <- tpr_mod_alasso[[i]][[1]]
}
avg.tpr_mod_alasso.n1 <- mean(tpr_mod_alasso.n1);
sd.tpr_mod_alasso.n1 <- sd(tpr_mod_alasso.n1);

av.tpr_tiv <- list(avg.tpr_tiv.n1, sd.tpr_tiv.n1)
av.tpr_mod_alasso <- list(avg.tpr_mod_alasso.n1,
sd.tpr_mod_alasso.n1)
#################
##--- Performance metrics for proposed modified alasso
## Accuracy
acc.mod.alasso.n1 <- vector()
for (i in seq(acc.mod.alasso))
{
  acc.mod.alasso.n1[i]  <- acc.mod.alasso[[i]][[1]]
}
# Calculate the average accuracy for modified alasso
avg.acc.mod.alasso.n1 <- mean(acc.mod.alasso.n1);
sd.acc.mod.alasso.n1 <- sd(acc.mod.alasso.n1)
```

```
## AuC
auc.mod.alasso.n1 <- vector()
for (i in seq(auc.mod.alasso))
{
  auc.mod.alasso.n1[i]  <- auc.mod.alasso[[i]][[1]]
}
# Calculate the average aucuracy for modified alasso
avg.auc.mod.alasso.n1 <- mean(auc.mod.alasso.n1);
sd.auc.mod.alasso.n1 <- sd(auc.mod.alasso.n1)

## gmean
gmean.mod.alasso.n1 <- vector()
for (i in seq(gmean.mod.alasso))
{
  gmean.mod.alasso.n1[i]  <- gmean.mod.alasso[[i]][[1]]
}
# Calculate the average gmean for modified alasso
avg.gmean.mod.alasso.n1 <- mean(gmean.mod.alasso.n1);
sd.gmean.mod.alasso.n1 <- sd(gmean.mod.alasso.n1)

##########
## Proposed Modified Lasso
acc.av.mod.alasso <- list(avg.acc.mod.alasso.n1)
acc.sd.mod.alasso <- list(sd.acc.mod.alasso.n1)
auc.av.mod.alasso <- list(avg.auc.mod.alasso.n1)
auc.sd.mod.alasso <- list(sd.auc.mod.alasso.n1)
gmean.av.mod.alasso <- list(avg.gmean.mod.alasso.n1)
```

```
  gmean.sd.mod.alasso <- list(sd.gmean.mod.alasso.n1)

  return(list( acc.av.mod.alasso, acc.sd.mod.alasso,
               auc.av.mod.alasso, auc.sd.mod.alasso,
               gmean.av.mod.alasso, gmean.sd.mod.alasso,
               av.tpr_tiv, av.tpr_mod_alasso))
}
## Function to calculate the performance metrics for unknown
perf.fun_unfilter <- function(op)
{
  dff <- op
  acc.lasso <- auc.lasso <- gmean.lasso <- list()
  acc.alasso <- auc.alasso <- gmean.alasso <- list()
  tpr_lasso <- tpr_alasso <- list()

  for (i in seq_along(dff))
  {
    acc.lasso[[i]] <- dff[[i]][[1]]
    acc.alasso[[i]] <- dff[[i]][[5]]

    auc.lasso[[i]] <- dff[[i]][[2]]
    auc.alasso[[i]] <- dff[[i]][[6]]

    gmean.lasso[[i]] <- dff[[i]][[3]]
    gmean.alasso[[i]] <- dff[[i]][[7]]

    tpr_lasso[[i]] <- dff[[i]][[4]]
    tpr_alasso[[i]] <- dff[[i]][[8]]
```

```
}
#############
##---Performance metrics for lasso
tpr_lasso.n1 <- vector()
for (i in seq(tpr_lasso))
{
  tpr_lasso.n1[i] <- tpr_lasso[[i]][[1]]
}
avg.tpr_lasso.n1 <- mean(tpr_lasso.n1);
sd.tpr_lasso.n1 <- sd(tpr_lasso.n1);

######
##---Performance metrics for alasso
tpr_alasso.n1 <- vector()
for (i in seq(tpr_alasso))
{
  tpr_alasso.n1[i] <- tpr_alasso[[i]][[1]]
}
avg.tpr_alasso.n1 <- mean(tpr_alasso.n1);
sd.tpr_alasso.n1 <- sd(tpr_alasso.n1);

av.tpr_lasso <- list(avg.tpr_lasso.n1, sd.tpr_lasso.n1)
av.tpr_alasso <- list(avg.tpr_alasso.n1, sd.tpr_alasso.n1)
##########
##------- Performance metrics for lasso
## Accuracy
acc.lasso.n1 <- vector()
for (i in seq(acc.lasso))
```

```
{
  acc.lasso.n1[i] <- acc.lasso[[i]][[1]]
}
# Calculate the average accuracy for lasso
avg.acc.lasso.n1 <- mean(acc.lasso.n1);
sd.acc.lasso.n1 <- sd(acc.lasso.n1)

## AuC
auc.lasso.n1 <- vector()
for (i in seq(auc.lasso))
{
  auc.lasso.n1[i] <- auc.lasso[[i]][[1]]
}
# Calculate the average aucuracy for lasso
avg.auc.lasso.n1 <- mean(auc.lasso.n1);
sd.auc.lasso.n1 <- sd(auc.lasso.n1)
## gmean
gmean.lasso.n1 <- vector()
for (i in seq(gmean.lasso))
{
  gmean.lasso.n1[i] <- gmean.lasso[[i]][[1]]
}
# Calculate the average gmean for lasso
avg.gmean.lasso.n1 <- mean(gmean.lasso.n1);
sd.gmean.lasso.n1 <- sd(gmean.lasso.n1)
######
##------ Performance metrics for proposed alasso
```

```
## Accuracy
acc.alasso.n1 <- vector()
for (i in seq(acc.alasso))
{
  acc.alasso.n1[i] <- acc.alasso[[i]][[1]]

}
# Calculate the average accuracy for alasso
avg.acc.alasso.n1 <- mean(acc.alasso.n1);
sd.acc.alasso.n1 <- sd(acc.alasso.n1)

## AuC
auc.alasso.n1 <- vector()
for (i in seq(auc.alasso))
{
  auc.alasso.n1[i] <- auc.alasso[[i]][[1]]
}
# Calculate the average aucuracy for alasso
avg.auc.alasso.n1 <- mean(auc.alasso.n1);
sd.auc.alasso.n1 <- sd(auc.alasso.n1)

## gmean
gmean.alasso.n1 <- vector()
for (i in seq(gmean.alasso))
{
  gmean.alasso.n1[i] <- gmean.alasso[[i]][[1]]
}
# Calculate the average gmean for alasso
```

```
  avg.gmean.alasso.n1 <- mean(gmean.alasso.n1);
  sd.gmean.alasso.n1 <- sd(gmean.alasso.n1)

  ####################
  ## lasso
  acc.av.lasso <- list(avg.acc.lasso.n1)
  acc.sd.lasso <- list(sd.acc.lasso.n1)
  auc.av.lasso <- list(avg.auc.lasso.n1)
  auc.sd.lasso <- list(sd.auc.lasso.n1)
  gmean.av.lasso <- list(avg.gmean.lasso.n1)
  gmean.sd.lasso <- list(sd.gmean.lasso.n1)

  ## alasso
  acc.av.alasso <- list(avg.acc.alasso.n1)
  acc.sd.alasso <- list(sd.acc.alasso.n1)
  auc.av.alasso <- list(avg.auc.alasso.n1)
  auc.sd.alasso <- list(sd.auc.alasso.n1)
  gmean.av.alasso <- list(avg.gmean.alasso.n1)
  gmean.sd.alasso <- list(sd.gmean.alasso.n1)

  return(list( acc.av.lasso, acc.sd.lasso,
               auc.av.lasso, auc.sd.lasso,
               gmean.av.lasso, gmean.sd.lasso,
               acc.av.alasso, acc.sd.alasso,
               auc.av.alasso, auc.sd.alasso,
               gmean.av.alasso, gmean.sd.alasso,
               av.tpr_lasso, av.tpr_alasso))
}
```

```
########################
## case 1
## with fscore filtering
fscore.metrics <- perf.fun(output[[1]])
## Case 2
## with ig filtering
information.metrics <- perf.fun(output[[2]])
## Case 3
## with chi2 filtering
chi2.metrics <- perf.fun(output[[3]])
## Case 4
## without filtering
lasso.metrics <- perf.fun_unfilter(output[[4]])
## Case 5
## with alasso with ridge filtering
ridge.metrics <- perf.fun(output[[5]])
## Case 6
## with alasso with mmle filtering
mmle.metrics <- perf.fun(output[[6]])

#####-----SAVE THE RESULTS-----#####
sink('ar105.txt')
print("---------- fscore ----------")
print("--------- Overall performance ---------")
print(fscore.metrics)
print("---------- information gain -------")
print("--------- Overall performance ---------")
print(information.metrics)
```

```
print("------------------------ chi2 ----------")
print("--------- Overall performance ---------")
print(chi2.metrics)
print("------------------------ lasso ---------")
print("--------- Overall performance ---------")
print(lasso.metrics)
print("------------------------ ridge ---------")
print("--------- Overall performance ---------")
print(ridge.metrics)
print("------------------------ mmle ----------")
print("--------- Overall performance ---------")
print(mmle.metrics)
sink()
# Save the results for generating the plots
saveRDS(output, file = "ar105.rds")
```

Appendix C

Appendix C: Resampling-based Group Lasso Ranking method

```
setwd(""/export/home/arpatil/Desktop/gglasso/sim)
##########################################
#                                        #
#   new feature selection method: RGLR   #
#                                        #
##########################################
new_filtering_method = function(cors)
{

  library(gglasso)
  library(SIS)
  library(glmnet)
  library(ncvreg)
  library(caret)
  library(PredPsych)
  library(randomForest)
  library(praznik)
  library(varbvs)
  library(e1071)
  library(nnet)
  library(Biocomb)
```

```
### penalized regression methods for
## average ranking with resampling

# 1. LASSO
lasso = function(x,y)
{
  cv.lasso <- cv.glmnet(y=y, x= x, alpha=1,family="binomial")
  return(cv.lasso)
}
# y should be -1 or 1 for gglasso
group.lasso = function(x,y,group0)
{
  cv.glasso <-  cv.gglasso(as.matrix(x),y,
  group=group0,loss="logit",nfolds=10)
  return(cv.glasso)
}

##########  main resampling methods

freq.matrix = function(train.x,train.y,group0=NULL)
{
  fit = gglasso(as.matrix(train.x),train.y,group=group0)
  beta.matrix = fit$beta

  ##############################################################
  # record frequency with max regression coefficients
  result.matrix = coefs.matrix = matrix(0,
  nrow=ncol(beta.matrix),ncol=nrow(beta.matrix))
```

```
  colnames(result.matrix) = rownames(beta.matrix)
  for(i in 1:nrow(result.matrix))
  {
    idx = which(beta.matrix[,i]!=0)
    result.matrix[i,idx] = 1
    coefs.matrix[i,idx] = beta.matrix[idx,i]
  }
  # return frequency matrix
  return(list(result.matrix,coefs.matrix))
}

resampling = function(train.x,train.y,group0=NULL)
{
  ## resampling
  idx = sample(1:nrow(train.x),
  as.integer(nrow(train.x)),replace=T) #####
  s.x = train.x[idx,]
  s.y = train.y[idx]
  obj = freq.matrix(s.x,s.y,group0)
  # return frequency matrix and coeficients matrix
  return(obj)
}

add = function(obj.mat,option)
{
  if(option=="frequency"){s = apply(obj.mat,2,sum)}
  if(option=="coef"){s = apply(obj.mat,2,
  function(x)max(abs(x)))}
```

```
  return(s)
}

resampling_results = function(train.x,train.y,group0=NULL)
{
  mat0 = matrix(0,nrow=100,ncol=ncol(train.x))
  mat1 = matrix(0,nrow=100,ncol=ncol(train.x))
  colnames(mat0) = colnames(mat1) = colnames(train.x)
  lens = NULL
  for(i in 1:100)  ##########################################
  {
    # lasso
    obj0 = resampling(train.x,train.y,group0)
    lens = c(lens,nrow(obj0[[1]]))
    mat0[i,] = add(obj0[[1]],"frequency")
    mat1[i,] = add(obj0[[2]],"coef")
  }
  freq.vec = apply(mat0,2,sum)/sum(lens)
  coef.vec = apply(mat1,2,function(x)max(abs(x)))
  return(list(freq.vec,coef.vec))
}

main = function(train.x,train.y,group0)
{
  results = resampling_results(train.x,train.y,group0)
  mu.vec      = results[[1]]
  max.coefs.vec     = results[[2]]
```

```
  s.mu.vec = sort(mu.vec, decreasing=T, index.return=T)
  rank.mu.vec      = mu.vec[s.mu.vec$ix]
  return(rank.mu.vec)
}

##### filtering methods
# F-score
fisher.dat = function(x,y)
{
  nump = ncol(x)
  dat = data.frame(y,x)
  f = fscore(dat, classCol=1, featureCol=c(1:nump))
  s.p = sort(f, decreasing = T, index.return = T)
  new.x = x[,s.p$ix]
  return(new.x)
}

# Info.gain
Info.gain0 = function(x,y)
{
  new.x            = as.matrix(data.frame(x,y))
  disc             = "equal interval width"
  attrs.nominal    = numeric()
  fit              = select.inf.gain(new.x,
  disc.method=disc, attrs.nominal=attrs.nominal)
  new.x            = x[,fit[,3]]
  return(new.x)
}
```

```
MRMR0 = function(x,y)
{
   obj = MRMR(x,y,k=ncol(x))
   #rank based on scores
   names0= names(obj[[2]])
   m.idx = match(names0,colnames(x))
   new.x = x[,c(m.idx)]
   return(new.x)
}
# random forest
rfs = function(x,y)
{
   x = data.frame(x)
   fit=randomForest(factor(y)~.,data=x,
   importance=T,proximity = T,ntree=100)
   fit = fit$importance
   #gini index
   fit = fit[order(fit[,4],decreasing = T),]
   m.idx = match(rownames(fit),colnames(x))
   new.x = x[,m.idx]
   return(new.x)
}
# bayesian variable selection
bv.selection.data = function(x,y)
{
   library(varbvs)
   fit = varbvs(X=as.matrix(x),y=y,Z=NULL,family="binomial")
```

```
  fit1 = summary(fit,nv=ncol(x))$top.vars
  names = fit1$variable
  m.idx = match(names,colnames(x))
  new.x = x[,c(m.idx)]
  return(new.x)
}
################################## metrics
accuracy <- function(truth, predicted)
  if(length(truth) > 0)
    sum(truth==predicted)/length(truth) else
      return(0)

sensitivity <- function(truth, predicted)
  # 1 means positive (present)
  if(sum(truth==1) > 0)
    sum(predicted[truth==1]==1)/sum(truth==1) else
      return(0)

specificity <- function(truth, predicted)
  if(sum(truth==0) > 0)
    sum(predicted[truth==0]==0)/sum(truth==0) else
      return(0)



AUC <- function(truth, probs, plot=FALSE){
  # probs - probability of class 1
  q <- seq(0, 1, .01)
  sens <- rep(0, length(q))
```

```
  spec <- rep(0, length(q))
  ly <- levels(truth)

  for(i in 1:length(q)){
    pred <- probs >= q[i]
    pred[pred] <- 1
    #pred <- factor(pred, levels=ly)
    sens[i] <- sensitivity(truth, pred)
    spec[i] <- specificity(truth, pred)
  }

  # make sure it starts and ends at 0, 1
  sens <- c(1, sens, 0)
  spec <- c(0, spec, 1)

  trap.rule <- function(x,y) sum(diff(x)*
  (y[-1]+y[-length(y)]))/2
  auc <- trap.rule(rev(1-spec), rev(sens))

  if(plot){
    plot(1-spec, sens, type="l", xlab="1-Specificity",
         ylab="Sensitivity", main="ROC Curve")
    legend("bottomright", legend=
    paste("AUC = ", round(auc, 3)), bty="n")
  }
  auc
}
##################### methods for performance check
```

```
# 1. Naive Bayes Classifier
NB0 = function(train.x,train.y,test.x,test.y)
{
  m.idx       = match(colnames(train.x),colnames(test.x))
  test.x      = test.x[,m.idx]
  x           = data.frame(train.x,train.y)
  model       = naiveBayes(train.y~.,data=x)
  pred.v      = predict(model,test.x,type="raw")
  probs       = pred.v[,2]
  predicted   = apply(pred.v,1,function(x)
  as.numeric(x[1]<x[2]))
  acc         = accuracy(test.y,predicted)
  sens        = sensitivity(test.y,predicted)
  spec        = specificity(test.y,predicted)
  auc0        = AUC(test.y,probs)
  G.mean      = sqrt(sens*spec)
  result      = list(acc,sens,spec,G.mean,auc0)
  return(result)
}

# 2. Support Vector Machine
SVM0 = function(train.x,train.y,test.x,test.y)
{
  m.idx       = match(colnames(train.x),colnames(test.x))
  test.x      = test.x[,m.idx]
  x           = data.frame(train.x,train.y)
  model       = svm(train.y ~ .,data=x,type =
```

```
  'C-classification', kernel="radial",probability=TRUE)
  predicted = predict(model,newdata=test.x)
  predicted = as.numeric(as.matrix(predicted))
  probs     = predict(model,newdata=test.x,probability=TRUE)
  probs     = attr(probs,"probabilities")[,2]
  acc       = accuracy(test.y,predicted)
  sens      = sensitivity(test.y,predicted)
  spec      = specificity(test.y,predicted)
  auc0      = AUC(test.y,probs)
  G.mean    = sqrt(sens*spec)
  result    = list(acc,sens,spec,G.mean,auc0)
  return(result)
}
# 3. Random Forest
RF0 = function(train.x,train.y,test.x,test.y)
{
  m.idx     = match(colnames(train.x),colnames(test.x))
  test.x    = test.x[,m.idx]
  train.x   = data.frame(train.x)
  x         = data.frame(train.x,train.y)
  fit       = randomForest(factor(train.y)~.,data=x,
  importance=T,proximity = T,ntree=100)
  predicted = predict(fit,newdat=test.x)
  predicted = as.numeric(as.matrix(predicted))
  probs     = as.numeric(predict
  (fit,newdata=test.x,type="prob")[,2])
  acc       = accuracy(test.y,predicted)
  sens      = sensitivity(test.y,predicted)
```

```
  spec        = specificity(test.y, predicted)
  auc0        = AUC(test.y, probs)
  G.mean      = sqrt(sens*spec)
  result      = list(acc, sens, spec, G.mean, auc0)
  return(result)
}
#4. Neural network
NN0           = function(train.x, train.y, test.x, test.y)
{
  m.idx       = match(colnames(train.x), colnames(test.x))
  test.x      = test.x[,m.idx]
  fit         = nnet(train.x, train.y  , maxit=200, size=5,
  MaxNWts = 20000)
  probs       = predict(fit, test.x, type="raw")
  predicted   = as.numeric(probs>0.5)
  acc         = accuracy(test.y, predicted)
  sens        = sensitivity(test.y, predicted)
  spec        = specificity(test.y, predicted)
  auc0        = AUC(test.y, probs)
  G.mean      = sqrt(sens*spec)
  result      = list(acc, sens, spec, G.mean, auc0)
  return(result)
}

#####  Simulation Data

model0 = function(X)
{
```

```
  return(exp(X)/(1+exp(X)))
}
response = function(X,B)
{
  return(exp(sum(X * B))/ (1 + exp(sum(X * B))))
}

multivariate_normal_sampler = function(cov0,num.gene,nsample)
{
  dat = NULL
  for(i in 1:num.gene)
  {
    L = chol(cov0)
    Z = matrix(rnorm(nsample*ncol(cov0)),nrow=nsample,
    ncol=ncol(cov0))
    X = Z%*%L
    dat = cbind(dat,X)
  }
  return(dat)
}

#for generating each of data of blocks
cov.mat = function(num.CpG,rho)
{
  tmp.cov = matrix(0,nrow =num.CpG,ncol=num.CpG)
  for(i in 1:num.CpG)
  {
    for(j in 1:num.CpG)
```

```
    {
      tmp.cov[i,j] = rho^(abs(i-j))
    }
  }
  return(tmp.cov)
}

library(mvtnorm)
feature_set = function(nsample,rho)
{
  #rho         =  0.2
  #nsample     = 500
  s           = 4 # scale parameter
  scale_p      = sqrt(s)
  num.genes = c(100,150,rep(50,7))
  num.CpGs  = c(1:9)
  dat = NULL
  for(i in num.CpGs)
  {
    #Block1 : 2 CpG sites for each of 150 genes
    if(i==1)
    {
      tmp.dat = scale_p*matrix(rnorm(nsample*num.genes[i]),
      nrow=nsample,ncol=num.genes[i])
      tmp.dat = model0(tmp.dat)
    }
    #Block2-9
    if(i>1)
```

```
    {
      num.gene = num.genes[i]
      num.CpG  = num.CpGs[i]
      cov0     = cov.mat(num.CpG, rho)
      tmp.dat  = scale_p*
      multivariate_normal_sampler(cov0,num.gene,nsample)
      tmp.dat  = model0(tmp.dat)
    }
    dat      = cbind(dat,tmp.dat)
  }
  return(dat)
}

calculate_true_coefs = function(all, delta)
{
  if(all == TRUE)
  {
    theta   = NULL
    num.CpG = 1:9
    for(Pg in num.CpG)
    {
      for(k in 1:Pg)
      {
        tmp.theta = ((-1)^(Pg+1)*delta)/sqrt(Pg)
        theta = c(theta,tmp.theta)
      }
    }
  }
```

```
  if(all == FALSE)
  {
    theta   = NULL
    num.CpG = 1:9
    for(Pg in num.CpG)
    {
      for(k in 1:ceiling(Pg/2))
      {
        tmp.theta = ((-1)^(Pg+1)*delta)/sqrt(Pg)
        theta = c(theta,tmp.theta)
      }
    }
  }
  return(theta)
}

generate_response_data = function(nsample,rho,theta0)
{
  cnt      = 0
  case.dat = NULL
  control.dat = NULL
  # generate response of 250 control and 250 case each
  while(cnt==0)
  {
    seed.val = sample(1:1000000,1)
    data.set = feature_set(nsample,rho)
    data.set = scale(data.set)
    feta = data.set%*%theta0
```

```
    fprob = exp(feta)/(1+exp(feta))
    y = rbinom(nsample, 1, fprob)
    idx = which(y == 1)

    if((length(idx)==(nsample/2)))
    {
      result = list(y,data.set,seed.val)
      cnt    = 200
      return(result)
    }
  }
}
grouping = function(g=9)
{
  g1 = rep(1:100,each=1)
  g2 = rep(c(101:250),each=2)
  g3 = rep(c(251:300),each=3)
  g4 = rep(c(301:350),each=4)
  g5 = rep(c(351:400),each=5)
  g6 = rep(c(401:450),each=6)
  g7 = rep(c(451:500),each=7)
  g8 = rep(c(501:550),each=8)
  g9 = rep(c(551:600),each=9)
  group = c(g1,g2,g3,g4,g5,g6,g7,g8,g9)
  return(group)
}

### end of data generation
```

```
### true positive rate from
## ranked features with increment of 25
tpr = function(true.var.names,rank.bv,rank.f,rank.info,
rank.rf,rank.glasso,rank.mrmr)
{
  d1=d2=d3=d4=d5=d6=NULL
  r1=r2=r3=r4=r5=r6=NULL
  seqs = seq(25,2500,25)
  # number of tv included in each ranking sets
  for(i in seqs)
  {
    d1 = c(d1,length(intersect(true.var.names,
    colnames(rank.bv[,1:i]))))
    d2 = c(d2,length(intersect(true.var.names,
    colnames(rank.f[,1:i]))))
    d3 = c(d3,length(intersect(true.var.names,
    colnames(rank.info[,1:i]))))
    d4 = c(d4,length(intersect(true.var.names,
    colnames(rank.rf[,1:i]))))
    d5 = c(d5,length(intersect(true.var.names,
    colnames(rank.glasso[,1:i]))))
    d6 = c(d6,length(intersect(true.var.names,
    colnames(rank.mrmr[,1:i]))))
  }

  result0 = data.frame(d5,d2,d3,d6,d4,d1)
  colnames(result0) = c("glasso","f","info",
  "mrmr","rf","bv")
```

```
  # each of 25 variables' ranking poistion
  r1 = match(true.var.names,colnames(rank.bv))
  r2 = match(true.var.names,colnames(rank.f))
  r3 = match(true.var.names,colnames(rank.info))
  r4 = match(true.var.names,colnames(rank.rf))
  r5 = match(true.var.names,colnames(rank.glasso))
  r6 = match(true.var.names,colnames(rank.mrmr))
  result1 = data.frame(r5,r2,r3,r6,r4,r1)
  colnames(result1) = c("glasso","f","info",
  "mrmr","rf","bv")

  result01 = list(result0,result1)

  return(result01)
}

sim.study = function(delta,nsample,rho,
topf.idx,theta0,group0)
{

  delta = delta
  ###############  generate 2500 CpG sites and response
  nsample = nsample
  rho     = rho       ############### change
  obj     = generate_response_data(nsample,rho,theta0)
  y1      = obj[[1]]
  x1      = data.frame(obj[[2]])
  colnames(x1)=paste("V",1:2500,sep="")
```

```
idx = which(y1==0)
gglasso.y1 = NULL
gglasso.y1       = y1
# (-1,1) is used instead of (0,1)
gglasso.y1[idx] = -1
gglasso.y1[-idx]= 1

s.idx <- sample(1:nrow(x1),
as.integer(nrow(x1)*.7),replace = F)
train.y = y1[s.idx]
gl.train.y = gglasso.y1[s.idx]
train.x = x1[s.idx,]
test.x =  x1[-s.idx,]
test.y =  y1[-s.idx]
gl.test.y = gglasso.y1[-s.idx]

# ranked data across filtering methods##############
### glasso based bootstrap filtering
a0 = main(train.x,gl.train.y,group0)
m.idx          = match(names(a0),colnames(train.x))
rank.glasso       = train.x[,m.idx]

### f-score
rank.f    = fisher.dat(train.x,train.y)
m.idx     = match(colnames(rank.f),colnames(train.x))

### Info.gain
```

```
rank.info       = Info.gain0(train.x, train.y)
m.idx = match(colnames(rank.info),colnames(train.x))

### random forest
rank.rf = rfs(train.x, train.y)
m.idx   = match(colnames(rank.rf),colnames(train.x))

### Maximum relevance Minimum redundancy
rank.mrmr       = MRMR0(data.frame(train.x), train.y)
m.idx = match(colnames(rank.mrmr),colnames(train.x))

### Bayesian variable selection
rank.bv         = bv.selection.data(train.x, train.y)
m.idx = match(colnames(rank.bv),colnames(train.x))
##############################################

# output 1 : True positive rates
# from 25 to 2500 with increment of 25
op1 = tpr(true.var.names, rank.bv,
rank.f, rank.info, rank.rf,
rank.glasso, rank.mrmr)
# combine all data set with top 100 features
## with the order of as follows.
op2 = list(rank.glasso[,topf.idx],
           rank.f[,topf.idx], rank.info[,topf.idx],
           rank.rf[,topf.idx], rank.mrmr[,topf.idx],
           rank.bv[,topf.idx])
```

```
    result = list(op1,op2,train.y,test.x,test.y)
  return(result)
}

method.performance = function(obj,train.y,test.x,test.y,
method,tv.idx)
{
  pf = NULL
  for(i in 1:length(obj))
  {
    tmp.dat     = obj[[i]][,c(1:tv.idx)]
    if(method=="NB")tmp.result = as.numeric(NB0(tmp.dat,
    train.y,test.x,test.y))
    if(method=="SVM")tmp.result = as.numeric(SVM0(tmp.dat,
    train.y,test.x,test.y))
    if(method=="RF")tmp.result = as.numeric(RF0(tmp.dat,
    train.y,test.x,test.y))
    if(method=="NN")tmp.result = as.numeric(NN0(tmp.dat,
    train.y,test.x,test.y))
    pf = rbind(pf,tmp.result)
  }
  # row is method name and column is performance:
  # ACC, SENS, SPEC, G-mean, and AUC
  result = pf
  return(result)
}

# main function of simulation study
```

```
delta=1
# 45 and 25
true.coefs= calculate_true_coefs(all=TRUE, delta)
half.true.coefs = calculate_true_coefs(all=FALSE, delta)
#disease related CpG sites :
#18 disease genes out of 600 genes
#true.coefs.idx       = c(1, 101:102,401:403,551:554,
751:755,1001:1006,1301:1307,1651:1658,2051:2059)
half.true.coefs.idx = c(2, 103,     404:405,555:556,
756:758,1007:1009,1308:1311,1659:1662,2060:2064)
theta0                = rep(0,2500)
true.idx              = c(half.true.coefs.idx)
theta0[true.idx ]=c(half.true.coefs)
idx = which(abs(theta0)>0)
true.var.names = paste("V",idx,sep="")
group0 = grouping(g=9)
var.n  = paste("V",1:2500,sep="")
group0 = data.frame(var.n,group0)

##########################################
result          = sim.study(delta=1,nsample=400,
rho= cors,topf.idx=1:2500,theta0,group0[,2])
ranking         = result[[1]][[1]]

true.loc        = result[[1]][[2]]
train.y         = result[[3]]
test.x          = result[[4]]
test.y          = result[[5]]
```

```
  #Check the performance of Top 25, 50, 75, 100
  # using Bayesian Classifier, SVM, Random Forest,
  and Neural Network
  top.v          = seq(10,200,by=10)
  nb2 = svm2 = rf2 = nn2 = NULL
  for(k in 1:length(top.v))
  {
    nb2 = cbind(nb2,method.performance(obj=result[[2]],
    train.y,test.x,test.y,method="NB",top.v[k]))
    svm2 = cbind(svm2,method.performance(obj=result[[2]],
    train.y,test.x,test.y,method="SVM",top.v[k]))
    rf2 = cbind(rf2,method.performance(obj=result[[2]],
    train.y,test.x,test.y,method="RF",top.v[k]))
    nn2 = cbind(nn2,method.performance(obj=result[[2]],
    train.y,test.x,test.y,method="NN",top.v[k]))
  }
  tpr.result     = rbind(nb2,rbind(svm2,rbind(rf2,nn2)))

  obj   = list(rank = ranking, trueloc = true.loc,
  tpr = tpr.result,data = result[[2]],
  train.y = train.y,test.x=test.x,
  teset.y=test.y,true.var.names=true.var.names)

  return(obj)
}

## packages for parallel computing
library(parallel)
```

```
library(foreach)
# library(doSNOW)
## Much better than doSNOW library when working on windows
## reduces computational time
## as it works best between the cores of one node
library(doParallel)

reps = 5
start_time = Sys.time()
# cores <- detectCores()
cores <- 3
cl <- makeCluster(cores -1) #not to overload your computer
registerDoParallel(cl)

# run data generation function for specified number of reps
res.data01 <- foreach(i = 1:reps,
                      # .combine=list,
                .packages=c("gglasso","SIS","glmnet","ncvreg",
                 "caret","PredPsych","randomForest","praznik",
                          "varbvs","e1071","nnet")) %dopar% {
                        new_filtering_method(0.1)
                                }

res.data04 <- foreach(i = 1:reps,
                      # .combine=list,
               .packages=c("gglasso","SIS","glmnet","ncvreg",
               "caret", "PredPsych","randomForest","praznik",
               "varbvs","e1071","nnet")) %dopar% {
```

```
                new_filtering_method(0.4)
                                    }
res.data07 <- foreach(i = 1:reps,
                      # .combine=list,
              .packages=c("gglasso","SIS","glmnet","ncvreg",
              "caret","PredPsych","randomForest","praznik",
              "varbvs","e1071","nnet")) %dopar% {
               new_filtering_method(0.7)
                                    }
stopCluster(cl)
Sys.time() - start_time

save(res.data01, file = "sim_result_100_0.1.RData")
save(res.data04, file = "sim_result_100_0.4.RData")
save(res.data07, file = "sim_result_100_0.7.RData")
```

Curriculum Vitae

Abhijeet R Patil earned his bachelor of engineering degree in Computer Science and Engineering from Sai Vidya Institute of Technology, Bangalore, India in 2012. He then earned his masters degree in Bioinformatics from R.V. College of Engineering, Bangalore, India. While pursuing a master's degree in Bioinformatics, he worked as a Teaching Assistant in the Biotechnology department. In 2015 he joined Sharnbasva University (previously called Appa Institute of Engineering and Technology), he worked as Assistant Professor (Teaching) where he taught various computer science and engineering courses over two years. Abhijeet joined the Computation Science program at The University of Texas at El Paso (UTEP) to pursue his Doctoral degree in fall 2017. Abhijeet was a Graduate Teaching Assistant at UTEP. He has first-authored four articles that are published and also have other articles currently under review.

Mr. Abhijeet's dissertation, "Gene Selection and Classification in High-throughput Biological Data With Integrated Machine Learning Algorithms and Bioinformatics Approaches" was supervised by Dr. Ming-Ying Leung and Dr. Sourav Roy.

Email address: arpatil@miners.utep.edu