General Penalized Logistic Regression For Gene Selection In High-Dimensional Microarray Data Classification

Derrick Kwesi Bonney
University of Texas at El Paso

GENERAL PENALIZED LOGISTIC REGRESSION FOR GENE SELECTION IN HIGH-DIMENSIONAL MICROARRAY DATA CLASSIFICATION

DERRICK KWESI BONNEY

Master's Program in Statistics

APPROVED:

To my

MOTHER and FATHER

with love

GENERAL PENALIZED LOGISTIC REGRESSION FOR GENE SELECTION IN HIGH-DIMENSIONAL MICROARRAY DATA CLASSIFICATION

by

DERRICK KWESI BONNEY

THESIS

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

MASTER OF SCIENCE

Department of Mathematical Sciences

THE UNIVERSITY OF TEXAS AT EL PASO

August 2020

Abstract

High-dimensional data has become a major research area in the field of genetics, bio-informatics and bio-statistics due to advancement of technologies. Some common issues of modeling high-dimensional gene expression data are that many of the genes may not be relevant. Also, reducing the dimensions of the data using penalized logistic regression is one of the major challenges when there exists a high correlation among genes.

High-dimension data correspond to the situation where the number of variables is greater or larger than the number of observations. Gene selection proved to be an effective way to improve the results of many classification methods. Many different methods have been proposed, however, these methods face a critical challenge in practical applications when there are high correlations among genes.

Penalized logistic regression using the Least Absolute Shrinkage Selection Operator (Lasso) has been criticized for being biased in gene selection. Adaptive Lasso (Alasso) was proposed to overcome the selection bias by assigning a consistent weight to each gene yet faces practical problems when choosing the type of initial weight. To address this problem, penalized logistic regression is proposed with the aim of obtaining an efficient subset of genes with high classification capabilities by combining the screening approach as a filter method and Adaptive Lasso with a new weight.

An alternative weight in adaptive penalized logistic regression is proposed to solve this problem. We worked on existing data set and we empirically verified the proposed method performed better than other existing methods. We then used Leukemia Cancer and Colon Cancer data set to test our proposed method. The experimental results reveal the proposed method is quite efficient and feasible and hence exhibits competitive performance in both classification accuracy and gene selection.

Keywords: Penalized Logistic Regression, Lasso, Alasso, Gene Expression Data.

Table of Contents

Chapter 1

Introduction

The emergence of DNA microarray technology is one of the major advancements made in the field of biology, medicine, bio-informatics and genetics research. Microarray is the study of how certain genes are turned on or off in cells and tissues. It is also a tool used to detect thousands of genes at the same time. Dealing with high-dimensional data set can be very challenging and time consuming because the DNA microarray produces thousands of genes. In relation to this, the number of variables and the number of observations are the two dimensions of the data (Algamal et al., 2015).

Overfitting, computational difficulties and estimation instability are mostly the statistical issues associated with modeling high-dimensional datasets. These issues make statistical microarray classification methods very difficult (Piao and Ryu, 2012).

Gene expression has been applied in many fields such as cancer classification, tumor detection (Algamal et al., 2015). Gene expression data sets often contain a large number of genes, d with only a few samples n resulting in the gene expression dataset matrix having fewer rows than columns, $p \gg n$ (Algamal et al., 2015).

The curse of dimensionality (Bellman, 1957) emerges when there is a large number of variables (p) and large number of observations (n). In gene expression datasets, there is a large number of genes d with few or a small number of observations n. One of the significant and vital properties of microarray data is that the number of genes p exceeds the number of samples n (Alonso-González, 2012). Dealing with the situation $p \gg n$, which is commonly known as high-dimensional data, poses a challenging task or problem in the application of statistical classification methods.

Overfitting and multi-collinearity are the most common problems that arises when applying classification problems in high-dimensional setting (Chen and Angelia, 2013). Reducing the dimensionality is one of the research areas in statistical applications and one best way to deal with the high dimensional data is by reducing the dimension of the dataset.

Irrelevant genes may introduce noise and reduce the classification accuracy. From statistical perspective, too many genes may lead to overfitting which negatively influence the classification.

Gene selection has received increasing attention, motivated by the desire to understand the structure in the high-dimensional gene expression datasets. Gene selection deals with selecting a small number of genes from a high-dimensional gene dataset. With these types of datasets, typically many genes are irrelevant and redundant, which could potentially impair the classification performance of the model. Accordingly, it is preferable to reduce the dimensionality of these datasets (Liu, 2012).

There have been many proposed methods such as Lasso, Alasso, Elastic Net, Minimax Concave Penalty (MCP) and Smoothly Clipped Absolute Deviation (SCAD). In high-dimensional gene classification, estimating the gene coefficients and penalized gene selection are obtained by applying the penalized methods. The most widely used penalty is the ℓ_1 penalty leading to the least absolute shrinkage and selection operator (Tibshirani, 1996).

Least absolute shrinkage and selection operator (Lasso) mainly got its popularity from high-dimensional data. Lasso as one of the methods, performs variable selection by assigning some coefficients to zero because of the ℓ_1 penalty property. (Tibshirani, 1996) used least absolute shrinkage and selection operator to estimate and find the regression coefficients through ℓ_1-norm penalty.

Chapter 2

Penalized Logistic Regression

In the fields of medicine and social science, logistic regression is considered one of the most important methods used in binary classification problems, where the response variable has two values coded as zero (0) and one (1). Also, in fields such as biomedical imaging, DNA micro-arrays and genomics, applying logistic regression to high-dimensional data, where the number of variables p , exceeds the number of sample size n, is one of the major problem and challenge that researchers face. Logistic regression approach deals with binary classification problems. The logistic regression is one of the most frequently and commonly used methods for assessing the relationship between a binary outcome and a set of covariates. Logistic regression is a widely used classification method in different classification fields. It combines classical logistic regression with a penalty to perform gene selection and classification simultaneously.

In this study, we use penalized logistic regression for the gene expression classification problems. The penalized logistic regression model penalizes the model because of too many variables in the model. In classification, we also encounter a lot of difficulties and challenges because the number of variables is more than the number of samples (high dimensionality). The purpose of the gene selection approach is mainly improving classification performance and also gives cost-effective genes.

Due to the high dimensionality of the data, some classification methods may not be applicable for analyzing gene expression data directly. We, therefore, remove the irrelevant, less contributive and noisy genes from the genes expression data. After removing irrelevant genes, we then analyze the microarray gene expression data.

The standard linear model performs poorly where you have a large multivariate data set containing a number of variables superior to the number of samples. So a better alternative is the penalized regression (allowing to create a linear regression model that is penalized, for having too many variables in the model). The consequence of imposing this penalty is to reduce (i.e; shrink) the coefficient value toward zero. This allows the less contributive variables to have a coefficient close to zero or equal zero.

In this thesis, we will highlight some common used penalized logistic regression methods such as Lasso regression, Ridge regression, Adaptive Lasso regression, Elastic Net regression, SCAD and MCP. In classification, the response variable of the logistic regression has two value either 1 for the positive class or 0 for the negative class.

In a penalized logistic regression, our main aim and purpose is to classify the response variable y, which is coded as 0 and 1, from high-dimensional explanatory variables. In general, in logistic regression, the response variable y is a Bernoulli random variable, and the conditional probability that y is equal to 1 given $\boldsymbol{x} \in \mathbb{R}^p$ which is denoted as $\pi(\boldsymbol{x})$ is

$$P(y_i = 1|x_{ij}) = \hat{\pi}_i = \frac{\exp\left(\beta_0 + \sum_{j=1}^{p} x_{ij}^T \beta_j\right)}{1 + \exp\left(\beta_0 + \sum_{j=1}^{p} x_{ij}^T \beta_j\right)} \qquad \text{where} \qquad j = 1, 2, ..., p \tag{2.1}$$

$$\ln\left(\frac{\pi_i}{1 - \pi_i}\right) = \beta_0 + \sum_{j=1}^{p} x_{ij}^T \beta_j \qquad \text{where} \qquad i = 1, 2, ..., n \tag{2.2}$$

The log likelihood function is

$$\begin{aligned} L(\boldsymbol{\beta}, y_i) &= \prod_{i=1}^{n} f(y_i) = \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i)^{1-y_i} \qquad \text{where} \qquad i = 1, 2, ..., n \\ &= -\left[\sum_{i=1}^{n} [(y_i \log(\pi(x_i)) + (1 - y_i) \log(1 - \pi(x_i))]\right] \end{aligned} \tag{2.3}$$

The maximum log-likelihood estimator of logistic regression (MLR) $\boldsymbol{\beta}_{MLR}$ is defined as follows:

$$\hat{\boldsymbol{\beta}}_{MLR} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\max} \left[\sum_{i=1}^{n} [(y_i \log(\pi(x_i)) + (1 - y_i) \log(1 - \pi(x_i)))\right] \tag{2.4}$$

Therefore the penalized log-likelihood is

$$\text{PLR} = -\sum_{i=1}^{n} \left[(y_i \log(\pi(x_i)) + (1 - y_i) \log(1 - \pi(x_i)) + \lambda P(\boldsymbol{\beta})) \right] \tag{2.5}$$

where $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_p)^T \in \mathbb{R}^{p+1}$ is a $(p+1) \times 1$ vector of unknown gene coefficients. Minimize to estimate the $\boldsymbol{\beta}$ vector

$$\hat{\boldsymbol{\beta}}_{PLR} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \left[-\sum_{i=1}^{n} [(y_i \log(\pi(x_i)) + (1 - y_i) \log(1 - \pi(x_i)) + \lambda P(\boldsymbol{\beta})) \right] \tag{2.6}$$

The $\lambda P(\boldsymbol{\beta})$ is the penalty term that truly penalizes the estimate of the model. The penalty term depends on the positive tuning parameter, λ. With loss of generality, we standardize the genes $\sum_{i=1}^{n} x_{ij} = 0$ and $\frac{1}{n} \sum_{i=1}^{n} x_{ij}^2 = 1$ where $j = 1, 2..., p$.

The penalty term penalizes the estimates and it genuinely depends on a positive tuning parameter. The strength of shrinkage of the explanatory variable is controlled by the tuning parameter. The larger the value of λ, the more weight will be given to the penalty term. Because the value of λ depends on the data, it can be determined using cross-validation method (Tibshirani, 2013). The tuning parameter find balance between the bias and variance to minimize the misclassification error. The penalized logistic regression adds a non-negative penalty term such that size of the gene coefficients in high dimension can controlled.

2.1 Lasso Penalized Logistic Regression

The least absolute shrinkage and selector operation (Lasso) was proposed by (Tibshirani, 1996) is a generalization of the ordinary least squares. The purpose of Lasso was to perform variable selection and regularization which will increase the prediction accuracy of the model. Lasso compresses and shrinks the regression coefficients toward zero by penalizing the regression model with a penalty term called l_1 norm. The Lasso uses the l_1-norm on

the logistic regression coefficients of the model. It applies both shrinkage and automatic variable selection simultaneously.
However, the Lasso has an advantage in that it is computationally feasible for high-dimension-al classification data. The most widely and popular penalized term is the least absolute shrinkage and selection operator (Lasso) (Tibshirani, 1996). Although Lasso has received significant interest, it suffers from some drawbacks. The main reasons why it has become very popular for high-dimensional estimation problems are its statistical accuracy for prediction and variable selection coupled with its computational feasibility.
When there are high correlations and group structure among the relevant genes, Lasso arbitrarily selects one gene and ignores the rest. In addition, Lasso inconsistently selects genes because the same amount of shrinkage is applied to each gene coefficient.
Lasso deals with a reduced number of predictors which gives a simpler model and it also performs better when there are some large coefficients and while coefficients are small. Lasso achieved and earned its popularity in high-dimensional data. To conclude, the Lasso regression helps to reduce overfitting in the model and also feature selection.

2.1.1 Limitations of Lasso

- It cannot select more genes than the number of samples.
- Another drawback of Lasso is that in high-dimensional data when $p \gg n$, it chooses at most n explanatory variables.
- When there is a high pairwise correlation among genes in the microarray data. Lasso selects only one gene or a few of them among a group of correlated genes.
- Lasso has bias in gene selection because it penalizes all the genes coefficients equally (Fan and Barut, 2014).
- Lasso does not have oracle properties (the probability of selecting the right set of

genes (with non-zero coefficients) converging to one and the estimators of non-zero coefficients are asymptotically normal with the same means and covariance (Fan and Li, 2001).

- Lasso cannot handle the effects of grouping (it tends to select only one gene from the whole group and does not give account or information of the one selected).

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{Lasso} &= \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\left[-\sum_{i=1}^{n}\{y_i\log(\pi_i)+(1-y_i)\log(1-\pi_i)\}+\lambda P(\boldsymbol{\beta})\right] \\ &= \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\left[L(\boldsymbol{\beta},y_i)+\lambda P(\boldsymbol{\beta})\right]\end{aligned} \tag{2.7}$$

where

$$\lambda P(\boldsymbol{\beta}) = \lambda\sum_{j=1}^{p}|\beta_j|$$

and lambda (λ) is a tuning parameter. It reduce the MLE estimator when $\lambda = 0$. Also, regression coefficient vanishes as $\lambda \to \infty$.

2.2 Alasso Penalized Logistic Regression

Adaptive Lasso was introduced to address the bias of Lasso and the bias of the estimate of the model is found by λ. Under Alasso, adaptive weight are used for penalizing different coefficients in the ℓ_1 penalty (Fan and Lv, 2008). The weight penalty is used to reduce the bias of Lasso by assigning smaller weight to variables with larger coefficients and vice versa (larger weight is assigned to variables with small coefficients and lower weight is also assigned to variables with large coefficients). While reducing the selection bias of the Lasso, we must also retain its sparsity property. We mainly use the Adaptive Lasso to penalize different coefficients in the ℓ_1 penalty and it is truly enjoying the oracle property.

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{Alasso} &= \underset{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}}}{\arg\min}\left[-\sum_{i=1}^{n}\{y_i\log(\pi_i)+(1-y_i)\log(1-\pi_i)\}+\lambda\sum_{j=1}^{p}w_j|\beta_j|\right]\\ &= \underset{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}}}{\arg\min}\left[L(\boldsymbol{\beta},y_i)+\lambda\sum_{j=1}^{p}w_j|\beta_j|\right]\end{aligned} \tag{2.8}$$

where $\mathbf{w}=(w_1,w_2,...,w_p)^T$ is a $p\times 1$ data driven weight vector.

2.3 Ridge Penalized Logistic Regression

Ridge regression mostly deals with analyzing data set which has the problem of multicollinearity because the number of predictors is greater than the number of observations. The purpose and aim of ridge regression is to prevent overfitting and reduce model completely. The penalty term ℓ_2 penalizes the regression model by shrinking the coefficients. The coefficients are shrunk via the ridge regression model and it helps reduce the model complexity via multicollinearity.

The penalty term called lambda (λ). For $\lambda = 0$, the penalty term has no effect and gives very good least-square coefficients. But as λ increases to infinite, the ridge regression coefficients get close to zero. When working with ridge regression, one has to first standardize because it is highly affected by a large number of predictors.

One advantage of the ridge regression, it performs well than Ordinary Least Square method in situation where you have a large multivariate data and the disadvantage is that it include all the predictors in the final model.

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{ridge} &= \underset{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}}}{\arg\min}\left[-\sum_{i=1}^{n}\{y_i\log(\pi_i)+(1-y_i)\log(1-\pi_i)\}+\lambda\sum_{j=1}^{p}\beta_j^2\right]\\ &= \underset{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}}}{\arg\min}\left[L(\boldsymbol{\beta},y_i)+\lambda\sum_{j=1}^{p}\beta_j^2\right]\end{aligned} \tag{2.9}$$

- λ is the shrinkage parameter.
- λ controls the amount of **regularization**.
- As $\lambda \downarrow 0$,we obtain the logistic regression solutions.
- As $\lambda \uparrow \infty$,we have $\hat{\boldsymbol{\beta}}^{\text{ridge}} = 0$,

2.4 Elastic Net Penalized Logistic Regression

Under the Elastic Net, the regression model is penalized with both ℓ_1-norm and ℓ_2-norm and the purpose is to correctly shrink coefficient (in the case of ridge regression) and also set some coefficients to zero (in the case of Lasso regression). The Elastic Net method combines the properties of Lasso and Ridge regression because it linearly combines ℓ_1 and ℓ_2 penalties (Zoul and Zhang, 2008).

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{Elastic} &= \underset{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}}}{\arg\min}\left[-\sum_{i=1}^{n}\{y_i\log(\pi_i)+(1-y_i)\log(1-\pi_i)\}+\lambda[(1-\alpha)||\boldsymbol{\beta}||_2^2/2+\alpha||\boldsymbol{\beta}||_1\right] \\ &= \underset{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}}}{\arg\min}\left[L(\boldsymbol{\beta},y_i)+\lambda[(1-\alpha)||\boldsymbol{\beta}||_2^2/2+\alpha||\boldsymbol{\beta}||_1\right]\end{aligned} \tag{2.10}$$

where

$$||\boldsymbol{\beta}||_1 = \sum_{j=1}^{p}|\beta_j|. \tag{2.11}$$

similarly, $||\boldsymbol{\beta}||_2$ is called the ℓ_2 or Euclidean norm

$$|\boldsymbol{\beta}|_2 = \sqrt{\sum_{j=1}^{p}\beta_j^2}. \tag{2.12}$$

2.5 SCAD Penalized Logistic Regression

SCAD regression method is known as Smoothly Clipped Absolute Deviation. This method was proposed to completely overcome the limitation of variable selection when the number of covariates is very large and it truly enjoys the oracle properties as compared to Lasso and Elastic Net (Chen and Angelia, 2013).

SCAD with concave penalty function overcomes the limitation of Lasso. The Lasso thresholding penalty functions do not simultaneously satisfy the mathematical conditions for continuity, sparsity, and unbiasedness.

The continuous differentiable penalty function defined by:

$$P_\lambda(\theta) = \left\{ I(\lambda \leq \lambda) + \frac{(\alpha\lambda - \theta)}{(\alpha - 1)\lambda} + I(\theta > \lambda) \right\}, \text{ for some} \quad \alpha > 2 \quad \text{and} \quad \theta > 0. \tag{2.13}$$

which improves the properties of the ℓ_1 penalty. We call this penalty function the smoothly clipped absolute deviation (SCAD) penalty. It corresponds to a quadratic spline function with knots at λ and $\alpha\lambda$. This penalty function leaves large values of θ not excessively penalized and makes the solution continuous. The smoothly clipped absolute deviation method not only selects important variables consistently, but also produces parameter estimators as efficient as if the true model were known, i.e., the oracle estimator, a property not enjoyed by the Lasso. The above features of the smoothly clipped absolute deviation method rely on the proper choice of tuning or regularization parameter, which is usually selected by cross validation. The SCAD regression method always estimate the non-zero parameter and also set the true model given the data set. SCAD helps us to get a very good goodness of fit and also promote a sparse estimate of the regression.

2.6 MCP Penalized Logistic Regression

MCP is also known as the Minimax Concave Penalty (Zhang, 2010). The rationale behind MCP is to eliminate unimportant predictors from the model while leaving the important predictors unpenalized. MCP is oracle-efficient and has oracle property. Hence, the penalty function for the MCP is in a concave or non-convex form. MCP exhibits weaker group behavior and tends to select very little groups (Zhang, 2010).

The idea behind the MCP is very similar to the SCAD. The continuously differentiable function of the penalty is defined by:

$$P_\lambda(\theta) = \left(\lambda - \frac{|\theta|}{\alpha}\right) \operatorname{sign}(\theta), \text{ for some } \quad \alpha > 1. \tag{2.14}$$

Unlike the SCAD with respect to the penalty term, MCP starts out by applying the same rate of penalization as the Lasso, then smoothly relaxes the rate down to zero as the absolute value of the coefficient increases. The MCP relaxes the penalization rate immediately while with SCAD, the rate remains at for a while before decreasing.

2.7 Marginal Maximum Likelihood Estimation (MMLE)

We use Sure Independence Screening as the variable selection or screening method. (Fan and Lv, 2008) introduced the SIS to decrease the number of variables p from the ultra-high to a reduced subset m. We use the MMLE technique because it helps us to get information about the true parameters of the penalized logistic regression that really describe the gene expression dataset. One assumption of the MMLE is all the observations are independent. The parameters of the model are estimated given the observed data and also set the parameter of the model to values which maximize the likelihood of the parameter given the data.

The method helps reduce the number of variables from p to d in order to fit the model. In high dimensionality, p (number of variables) $\gg n$ (number of samples) making it incapable to develop and fit a penalized logistic regression. But when it is reduced to d number of variables, a very good model can be fitted.

The main aim of this thesis is to improve the classification accuracy by using the MMLE and weights by ranking and resampling of genes and selection of contributive genes. During the

first stage, which is the screening stage, we arrange the genes according to their importance and relevancy. The main purpose of screening is to identify the genes which are very contributive and important.
After the screening stage, our focus is to find the genes that most relevant and related and we select them by implementing a penalized logistic regression method with Alasso. In the proposed method, we proposed the weights to address the issue of high correlation during the screening stage.

Chapter 3

Literature Review

This chapter is to present a literature review on penalized logistic regression for high-dimensional data, the steps involved in variable selection and screening steps and methods. Limitations of variable selection methods in high-dimensional settings is also addressed. Literature on variable screening methods is further examined.

3.1 Variable Selection

In most practical problems, the analyst has a rather large pool of possible candidate variables, of which only a few are likely to be important. Finding an appropriate subset of variables for a model is often called the variable selection problem. This problem generally falls under the exploratory category of observational studies (e.g., exploration of numerous gene sets that might not all be associated with the continuous response). This is because a model with numerous explanatory variables may be difficult to work with. Also, the presence of many highly inter-correlated explanatory variables may worsen the model's predictive ability. In recent years, there has been much research efforts on dealing with the challenging problem of variable selection in high dimensional data (small-n with a large p). This is largely due to modern applications in medical studies, genetic research, bioinformatics, and other fields. The consistency of various traditional methods of variable selection has been established over the years. These methods include but are not limited to the Lasso (Tibshirani, 1996), the SCAD (Zhang, 2010), and related models. These methods have been applied for simultaneously selecting important variables and estimating their effects in high-dimensional statistical inference, and have demonstrated excellent performance in simulation studies. All these methods have been shown to be useful and can be formulated as penalized optimization problems which could be selection consistent in a low dimensional data setup (Zoul and Zhang, 2008). However, in a high-dimensional setup, these methods may not work well due to the simultaneous challenges of computational expediency, statistical accuracy, and algorithmic stability. When the predictor dimension is much larger than the sample size, efficient algorithms exist for methods like Lasso where the objective functions are strictly convex. Similarly, for methods like SCAD, it is computationally difficult or non-trivial task to know how to optimize these non-convex objective functions (Zou and Li, 2008). Though most of these variable selection methods above tend to have good theoretical properties; the difficulty in choosing a penalty function still remains a challenge. To address the problem of variable selection in high-dimensional setup, (Ing and Lai, 2011) introduced a fast stepwise regression algorithm called high-dimensional information criterion (HDIC) which has been shown to have the oracle property of being equivalent to least squares regression on an asymptotically minimal set of relevant predictors under a strong sparsity assumption. This method was shown to have impressive performance. The question still remains; how we can perform variable selection in a high-dimensional setup in logistic regression.

Variable selection is used for improving the model performance and give better predictions. With many irrelevant, noisy or unreliable variables, removal of these will typically improve the predictions and or reduce the model complexity. Improvement of statistical properties can also be a reason for doing variable selection. In some situations, the purpose of variable selection is to obtain a model that is easier to understand, for example, by getting rid of all the variables that do not contribute significantly to the model. This may not give better predictions. They may even get colloquial worse. However, adequate models using as few variables as possible are sometimes desired. The main purpose of the variable selection is to disclose only those variables related to the response. Screening and model building are the two steps involved in variable selection. The screening step is to reduce the large number of variables into moderate size while maintaining most of the informative variables relevant to the clinical response.

3.2 Variable Screening

One reasonable solution to variable selection in a high dimensional setup is variable screening. As an example, consider a gene expression data set $p \gg n$ with $p = 6000$ genes as predictor variables, there would be 2^{6000} possible regression models to be considered. This would be an overwhelming task hence we need do some variables screening. Variable screening is therefore useful in reducing the dimensionality of the variable space to a moderate one and then we can apply variable selection techniques. Motivated by these concerns, there has been a dramatic growth in the development of statistical methodology in the analysis of high-dimensional data.

According to (Fan and Lv, 2008), a common practice in variable screening is using independence learning which treats the features as independent and thus applies marginal regression techniques. Motivated by the aforementioned fundamental challenges of ultra-high dimensional data analysis, the sure independence screening (SIS) was formally introduced and justified by (Fan and Lv, 2008) to address both issues of scalability and noise accumulation. SIS method as proposed by (Fan and Lv, 2008) for linear regression first filters out the variables that have weak correlation with the response, effectively reducing the dimensionality to a moderate scale below the sample size n, and then performing variable selection and parameter estimation through a lower dimensional penalized least squares method.

According to (Liu, 2012), although this approach is well-liked and popular, it does not perform well in some situations. First, unimportant variables that are heavily correlated with important predictors are more highly likely to be selected than relevant variables that are weakly associated with the response. Second, important variables that are not marginally signifi-cantly related to the response are screened out. Finally, there may be collinearity between variables that may affect the calculations of the individual predictors. An important methodology extension, Iterative Sure Independence Screening (ISIS), was also proposed by (Fan and Lv, 2008) to handle cases where regularity conditions may fail, such as when some important variables are marginally uncorrelated with the response, or when an unimportant predictor has higher marginal correlation than some important features. This method iteratively performs variable selection to recruit small number of predictors, computing residuals based on the model fitted using these recruited predictors, and then using as the working response variable to continue recruiting new predictors.

3.2.1 Screening Approach

The SIS (Sure Independence Screening) is a way to screen genes and it has shown excellent performance in reducing the high dimensionality of the data. To improve the prediction stability and to expedite computation time. (Fan and Lv, 2008) proposed an SIS procedure for high-dimensional linear model. One special and unique property of the SIS is that it has the ability to retain all truly contributive and important with a probability tending towards one (Fan and Lv, 2008). Under the logistic regression, we use MMLE and the SIS to rank relevant variables. We will use the marginal maximum likelihood estimator to rank the significance of each variable and after they are arranged in order of relevance. The genes with the largest or bigger marginal maximum likelihood estimator are placed at the top and hence they are the most important genes. (Fan and Lv, 2008) introduced the SIS to decrease the number of variables p from the ultra-high to a reduced subset size m. To determine the cutoff value of the most top screened genes, a predefined threshold is usually used. We choose or calculate $m \leq [\frac{n}{\log(n)}]$ for continuous outcome or $m \leq [\frac{n}{4\log(n)}]$ for category case depending on the different variants of SIS. This threshold guarantees that the number of screened genes is less than the number of samples. One problem of the Sure Independence Screening (SIS) is that it is unable to select important variables that are weakly and marginally associated.

Sure Independence Screening (SIS)

The sure independence screening (SIS) was instituted to reduce the high dimension below the sample size to efficiently choose the best subset of variables to predict clinical responses. Although this approach is popular, it does not perform well in some situations.
First, important variables that are not marginally significantly related or important to the response are screened out. Second,when the unimportant variables that are heavily correlated with important variables in the model,they are more highly likely to be selected than important variables that are weakly associated with the response.
Finally, there may exist collinearity between variables that may influence the calculations of the individual predictors. According to the paper by (Fan and Lv, 2008), a common practice for variable screening is using independence learning which treats the features as independent and thus applies marginal regression techniques. Yet the theoretical properties of such computationally expedient procedures were not well understood for a long while. Motivated by the aforementioned fundamental challenges of ultra-high dimensional data analysis, the sure independence screening (SIS) was formally introduced and rigorously justified by (Fan and Lv, 2008) to address both issues of scalability and noise accumulation.

3.3 Dimension Reduction

Dimension reduction or feature selection is an effective strategy to deal with high dimensionality. With dimensionality reduced from high to low, computational burden can be reduced drastically. Meanwhile, accurate estimation can be obtained by using some well-developed lower dimensional method to reduce dimensionality p from a large scale (say, $\exp((n\epsilon))$ for some $\varepsilon > 0$) to a moderate d by a fast and efficient method. We achieve this by introducing the concept of sure screening and proposing a sure screening method (SIS). Here and in the sequel, by sure screening we mean a property that all the important variables survive after variable screening with probability tending to one. This dramatically narrows down

the search for important predictors.
A challenge with high dimensionality is that important predictors can be highly correlated with some unimportant predictors, which usually increases with dimensionality.

3.4 Importance of Feature Selection

The following are some reasons for feature selection:

1. It reduces overfitting.
2. It enables machine learning algorithms to train faster.
3. It improves the accuracy of a model if the right subset is chosen.
4. It reduces the complexity of a model and makes it easier to interpret.

Chapter 4

Methodology

In this chapter, we describe the variable screening methods which reduces the high dimension of the datasets using Sure Independence Screening (SIS). We then describe methods of variable screening (Lasso, Alasso, SCAD and MCP). Finally, we present the methods needed to assess and evaluate variable selection methods with penalized logistic regression model.

4.1 The Proposed Method

The goal of gene selection is to improve classification performance, to provide faster and more cost-effective genes and to achieve a better knowledge of the underlying classification problem.

High dimensionality can negatively influence and affect the classification performance of a classifier by increasing the risk of overfitting and increase the computational time (Liu, 2012). Therefore, removing irrelevant and noise genes from the original microarray gene expression data is essential for applying classification methods to analyze the microarray gene expression data. First, Lasso has proven inconsistent in gene selection but SCAD and Alasso have proven their consistent in gene selection. However, they can not address the highly correlated genes.

The main aim of the thesis is to improve the classification accuracy by exploring a penalized logistic regression method by building rank of genes and selection of relevant genes.

In cancer classification, genes exhibit certain natural grouping structures; for example, gene expression profiles may be grouped according to their pathways, and it is often preferable that a group of genes are either kept or eliminated from the classification together. Furthermore, the regularization method that selects the correct subset of genes with probability one is desired. The adaptive elastic net was successfully applied to gene selection in cancer classification (Algamal et al., 2015).

In our proposed method, we develop a formula for the weight used in the Adaptive Lasso to help penalize the model. Other methods have been proposed such as the Lasso, Alasso, Elastic Net, MCP and SCAD. The weight used in the Adaptive Lasso was randomly generated. The proposed new weight will penalize the logistic model. To verify our proposed weight, we randomly generated a datasets with the response variable y having either 0's or 1's because it is a logistic regression model. After generating the datasets, we split the data set into 70% training set and 30% testing set. The training set is used to fit the logistic regression model or parameters while testing set is used to assess the performance of the model.

After splitting the datasets into training and testing, we fit the marginal maximum likelihood and hence reduce the feature from p to d using the Sure Independence Screening. We build the logistic regression model and hence generate the average regression coefficients and average standard error. The proposed weight is the ratio of the average regression coefficients and average standard error. Then we plug our proposed weight into the Adaptive Lasso, and move on to assess the performance metrics of the model. We check the model performance such as C-index, root mean square (RMS) and Area under the ROC Curve (AUC) for the model against the proposed method. After the simulation, we apply this proposed method to the Leukemia and Colon cancer datasets.

Next, we present a brief description of the proposed weight used in this thesis. A good weight is non-negative and the j^{th} weight value W_j is the weight for the i^{th} observation. The resampling technique is used to find the weight $W_j = \frac{\text{Average Regression Coefficient}}{\text{Average Standard Error}}$ where $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\beta}}_1, \hat{\boldsymbol{\beta}}_2, ..., \hat{\boldsymbol{\beta}}_p)^T$ be the vector of regression estimate, $\mathbf{s} = (\mathbf{s}_{\hat{\boldsymbol{\beta}}_1}, \mathbf{s}_{\hat{\boldsymbol{\beta}}_2}, ..., \mathbf{s}_{\hat{\boldsymbol{\beta}}_p})^T$ be the vector of the standard error of the regression. Then $W_j = \frac{\hat{\boldsymbol{\beta}}_j}{\mathbf{s}_{\hat{\boldsymbol{\beta}}_j}}$ and $\mathbf{w}_{ratio} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_p)^T$ be the ratio weight vector where $j = 1, 2, 3, .., p$. Then we plug the new weight and new predictor variable in the Adaptive Lasso.

We remark that observations with larger weights have more influence in the analysis of the model than observations with smaller weights. The weight defined is related to the variance of the observation and the regression coefficient of the model. Observations with small weights have a relatively large variance.
Regression coefficients describe the relationship between each predictor variable and response. In practice, we take the average of all the regression coefficients and average standard error. Next standardize the regression coefficients and standard error on the same scale. Since they are all matrices, we divide the average regression coefficients and average standard coefficients to obtain the new weights. After obtaining the weight for the i^{th} observation, we apply the weight to the Adaptive Lasso.
Alternatively, we can make use of an built-in function of weight in R programming with Repository (CRAN) where $weight$ = `avg.std.stat`. Using this method, we find the average standard statistics and measure the number of variations in the data set. When values are close to the mean; it means that it has a low standard deviation and when the values are away from the mean, it has high standard deviation.

4.2 Algorithm of the Proposed Method

We outline below the steps in our algorithm:
Step 1: Generate a data set. We can choose to select any any sample size for the number of n (the number of observations) and p (the number of variables).
Step 2: Split the data set into 70% training set and 30% testing set. The training set is used to fit the model or parameters while testing set are used to assess the performance of the model. Other researchers prefer using training and testing datasets in the following ratio (80:20), (60:40), i.e., 80% training set and 20% testing set.
Step 3: Fit MMLE and reduce the features p to d using the SIS Condition, e.g, $m = \frac{n}{\log n}$. Find the MMLE using the SIS. Hence, we get our new n and p.

Step 4: Build logistic regression model and hence find the regression coefficient $(b_1, \cdots, b_k)$ and the standard error of the logistic regression model $(se_1, se_2, \cdots, se_k)$. After getting the output or result for regression coefficient and standard error. we find the average regression coefficient and average standard error. This produces the Weight (W_i)

Step 5: Use the resampling technique to find the weight where $W_j = \frac{\text{Average Regression Coefficient}}{\text{Average Standard Error}}$ where $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\beta}}_1, \hat{\boldsymbol{\beta}}_2, ..., \hat{\boldsymbol{\beta}}_p)^T$ be the vector of regression estimate, $\mathbf{s} = (\mathbf{s}_{\hat{\boldsymbol{\beta}}_1}, \mathbf{s}_{\hat{\boldsymbol{\beta}}_2}, ..., \mathbf{s}_{\hat{\boldsymbol{\beta}}_p})^T$ be the vector of the standard error of the regression. Then $W_j = \frac{\hat{\boldsymbol{\beta}}_j}{\mathbf{s}_{\hat{\boldsymbol{\beta}}_j}}$ and $\mathbf{w}_{ratio} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_p)^T$ be the ratio weight vector where $j = 1, 2, 3, .., p$. Replicate 100 times.

Step 6: Use the W_i in the Adaptive Lasso.

Step 7: Estimate and assess the performance metrics of the model. We use the model performance metric such as C-index, RMS and AUC score for the model against the proposed method.

4.3 Performance Metrics and Evaluation

Performance metrics are used to evaluate the performance of Machine Learning algorithms, classification as well as regression algorithms. To evaluate the predictive performance of the proposed method. We will use performance metrics for classification problems to assess the performance of the model.

4.3.1 Performance Metrics for Classification Problems

Confusion Matrix is one of the way to measure the performance of a classification problem where the output can be of two or more type of classes (Liu, 2012). The confusion matrix has "Actual" and "Predicted" and furthermore, both the dimensions have "True Positives (TP)", "True Negatives (TN)", "False Positives (FP)", "False Negatives (FN)". Explanation of the terms associated with confusion matrix are as follows:

1. **True Positives(TP)**

   It is the case when both actual class and predicted class of data point is 1.

2. **True Negatives(TN)**

   It is the case when both actual class and predicted class of data point is 0.

3. **False Positives(FP)**

   It is the case when actual class of data point is 0 and predicted class of data point is 1.

Accuracy

It is the most common performance metric for classification procedures. It can be defined as the number of correct predictions made as a ratio of all predictions made. Accuracy measures all the correct prediction of the classifier compared to the overall data points. It however does not give us the best picture of the cost of misclassification or unbalanced testing data set. A typical classification method should maximize the accuracy.

Accuracy $= \frac{\text{TP + TN}}{\text{TP +FP +TN + FN}} \times 100\%$

Precision

Precision is the number of correct positives returned by our Machine Learning model. Precision is how consistent results are when measurements are repeated

Precision $= \frac{\text{TP}}{\text{TP + FP}}$

Recall or Sensitivity

Recall or sensitivity is the number of positives returned by our Machine Learning model.

Recall $= \frac{\text{TP}}{\text{TP + FN}}$.

Specificity

Specificity is the number of negatives returned by our Machine Learning model.

Specificity $= \frac{\text{TN}}{\text{TN + FP}}$.

$\mathbf{F_1}$ Score

F_1 score is the weighted average of the precision and recall. The best value of F_1 would be

1 and worst would be 0.

F_1 Score $= \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$.

AUC (Area Under ROC Curve)

AUC (Area Under Curve)-ROC (Receiver Operating Characteristic) is a performance metric for classification problems that is based on varying threshold values. ROC is a probability curve and AUC measure the separability. AUC-ROC metric will tell us about the capability of model in distinguishing the classes.The higher the AUC, the better the model. AUC - ROC curve is a performance measure for classification problem at various thresholds settings. A prefect overall classification produces an AUROC = 1 whereas a random overall classification has an AUROC = 0.5

Geometric Mean

The Geometric Mean of sensitivity and specificity was used to check the joint performance.

G-Mean $=\sqrt{\text{Sensitivity} \times \text{Specificity}}$

Chapter 5

Data Analysis

5.1 Datasets description

Table 5.1: Experimental study-Datasets description.

Data Set	Number of Samples	Number of Genes	Class
Colon	62	2000	40 tumor/22 normal
Leukemia	72	7129	47 ALL/25 AML
Prostate	102	12600	52 tumor/50 normal
Lung	181	12533	31 MPM/ 150 ADCA

5.2 Real-World Dataset Application

The datasets that have been exploited to test the effectiveness of our proposed method were composed of microarray gene expression data.
The four high dimensional gene expression data to be used are:

- colon cancer (Alon et al, 1999)
- leukemia (Golub et al, 1999)
- prostate cancer (Singh et al, 2002)
- lung cancer (Gordon et al, 2002)

5.2.1 Issues related to Real World Data (Data Processing)

Real world data are mostly imperfect and colloquial. Data processing refers to steps that one takes to make data finally ready for a supervised or unsupervised task. Some datasets may be incomplete (special attributes are missing), noisy (contain errors) and inconsistent. When data sources are fragmented and inaccurate data are some possible issues related to real word data (Pyle, 1999).
Possible Steps in data preparation;

1. Data reduction

2. Sampling

3. Data cleaning

4. Variable transformation

5. Data integration

The amount of data can be reduced in both size (number of rows or observations) and dimension (number of variables or columns). Dimension reduction, which is also called feature selection or feature extraction, transforms the data in the high-dimensional space to a space of fewer dimensions.

1. Data reduction deals with the transformation of unstructured information into a corrected, ordered, and simple form. The amount can be reduced by removing redundant observations or variables.

2. Sampling deals with partitioning the data into different sets. It also a process where a predetermined number of observation is derived from the whole population (Little, 1987).

3. Data cleaning also involves identifying wrongly or inappropriately recorded values. It is always necessary and time-consuming in nearly all real world data analysis projects.

It can be an outlier detection problem. Detecting and deleting redundant rows is a necessary step in data cleaning. Identifying wrongly recorded values is always a necessary and time-consuming step in all real world analysis project or research.

4. Data integration is when datasets from multiple sources are integrated by merging or concatenating into a data matrix or a data cube.

5. Variable transformation is the situation where variables can be transformed in different ways, to serve better for the analytic purposes.

5.3 Analysis of Data

To prove the effectiveness of the proposed weight, a randomly generated dataset was used to assess the performance of the model and also the number of variables selected. Furthermore, DNA microarray datasets with different sample sizes and number of genes were used.

Table 5.2: Number of Variables Selected

Method	Number of Selected Genes
Lasso	21
Alasso	26
Ridge	50
SCAD	15

From the above output, we can clearly see that the Ridge method selected more variables than other methods such as Lasso, Alasso, SCAD, Elastic net and MCP.

Table 5.3: Classification Performance results

Method	Accuracy
Lasso	0.9318
Alasso	0.9491
Elastic Net	0.9276
SCAD	0.9478
MCP	0.9478

Table 5.4: Computational time for the Model

Method	Number of Seconds
Model fitting	0.01
Cross validation	0.52

From Table 5.3 and 5.4, The adaptive lasso with the proposed weight produced a higher accuracy value of 0.9491 which was relatively higher than Lasso, SCAD, MCP and Elastic Net. The model fitting and cross validation takes 0.01 and 0.52 seconds respectively. Below are some diagrams to support the above outputs.

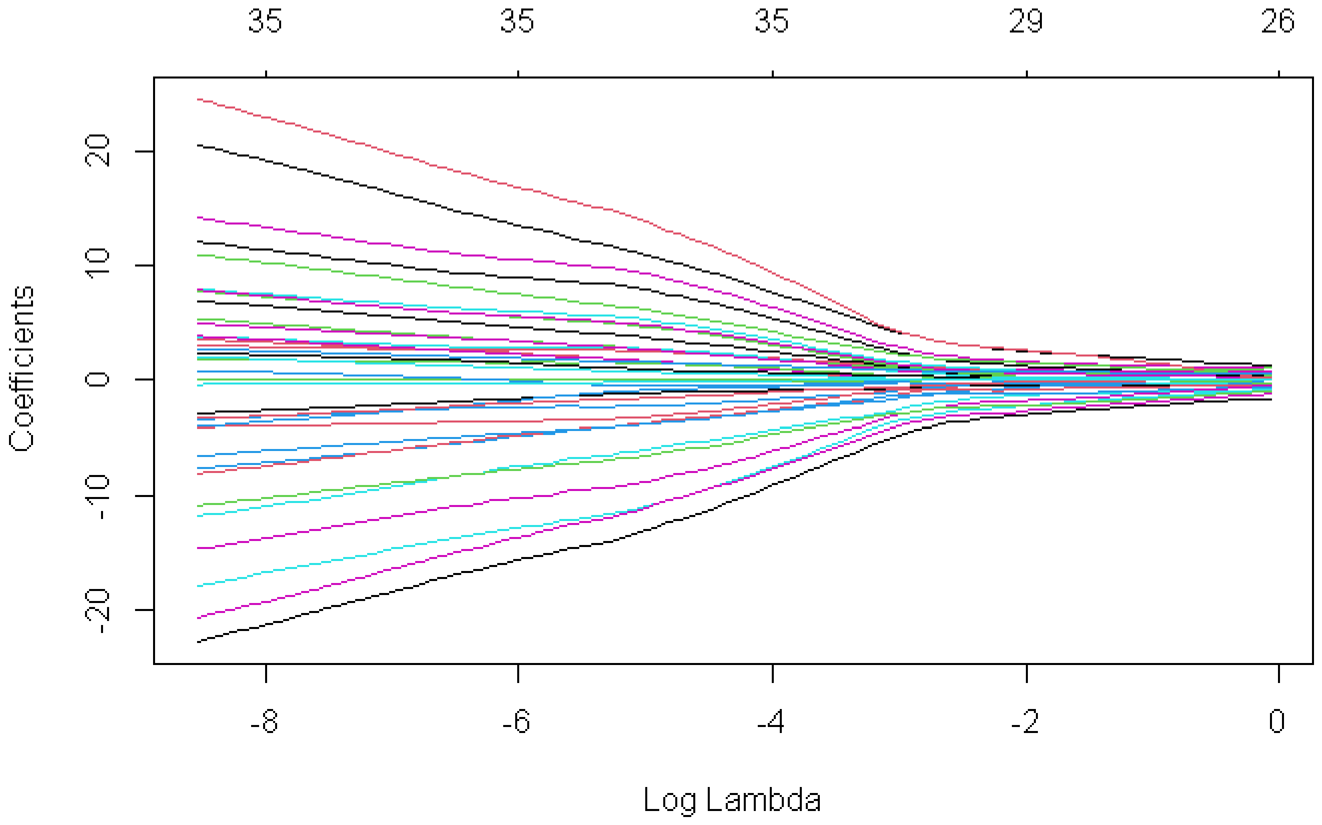

Figure 5.1: Alasso plot

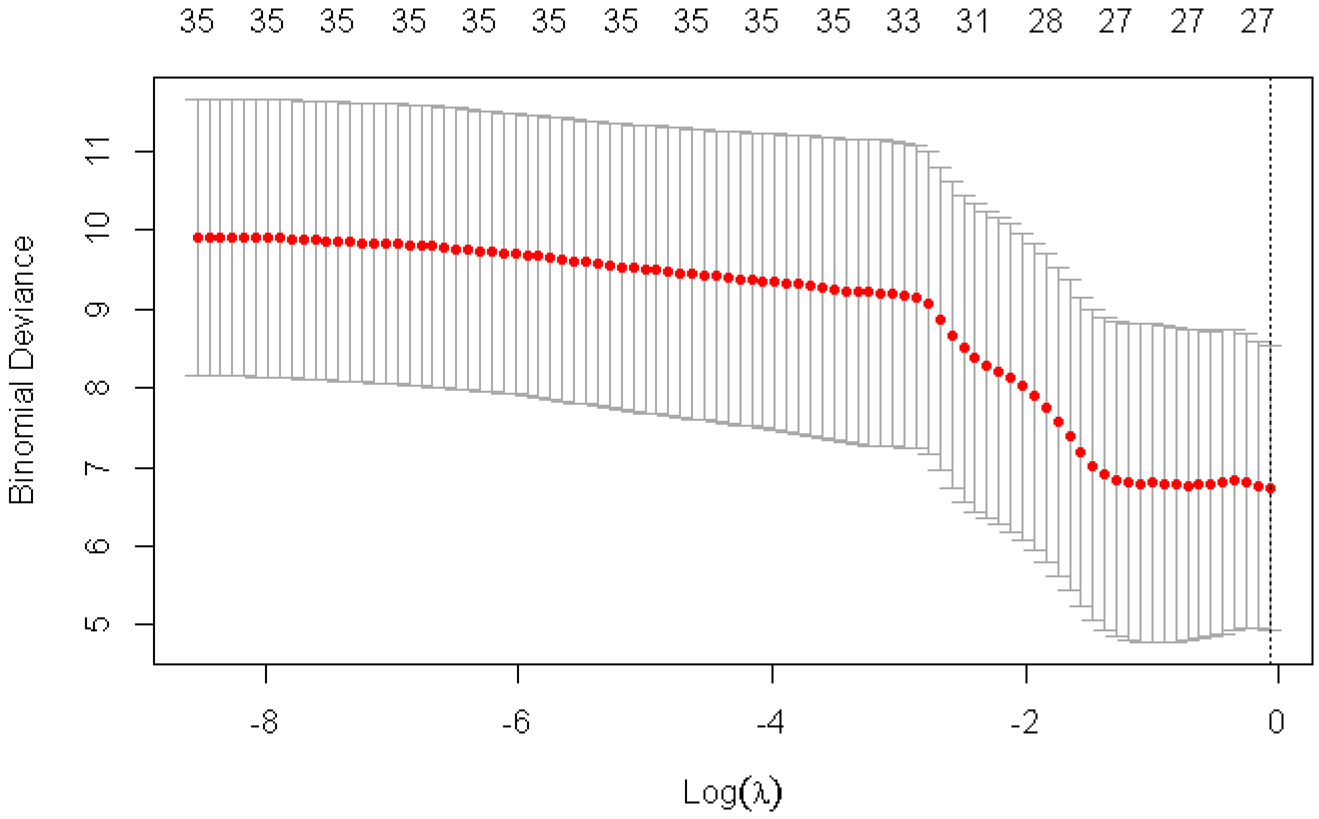

Figure 5.2: CV of Alasso plot

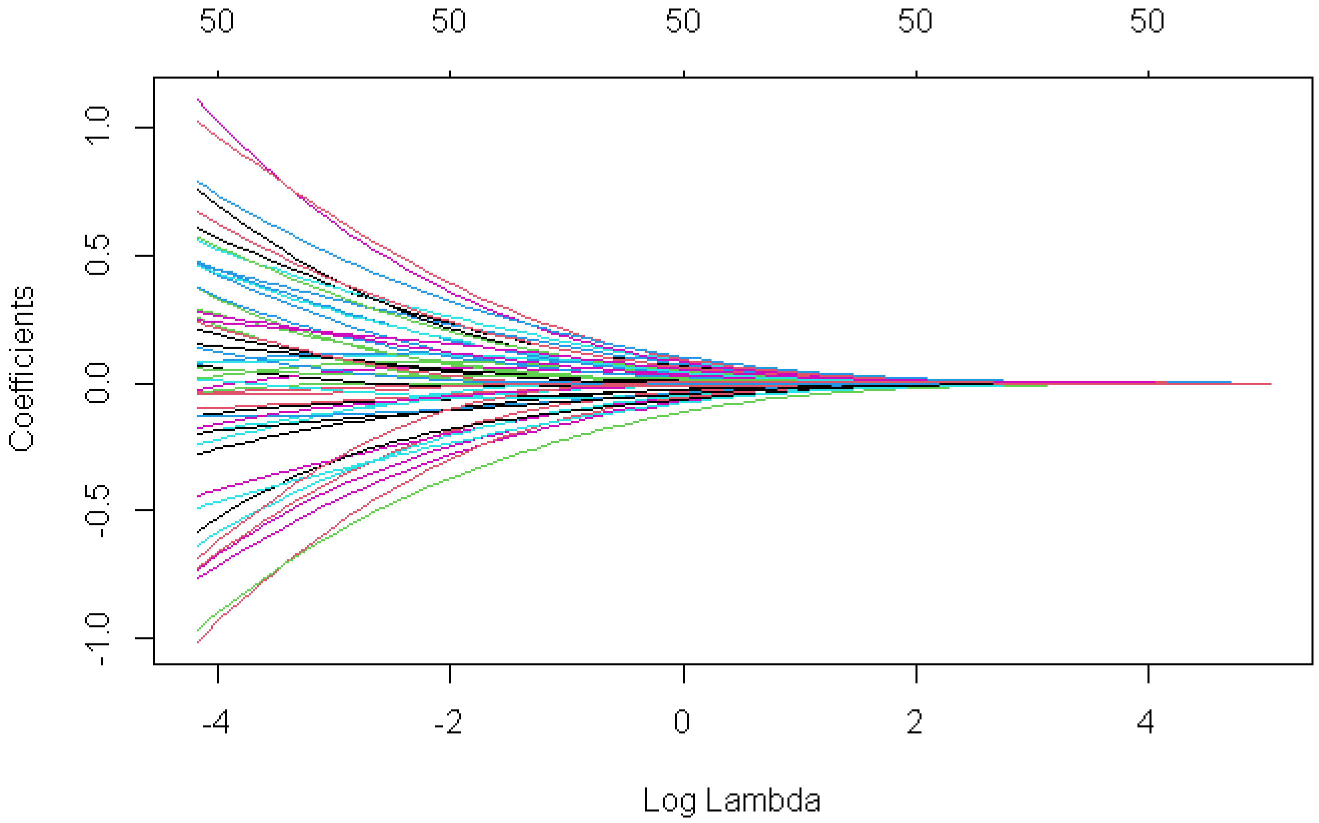

Figure 5.3: Ridge plot

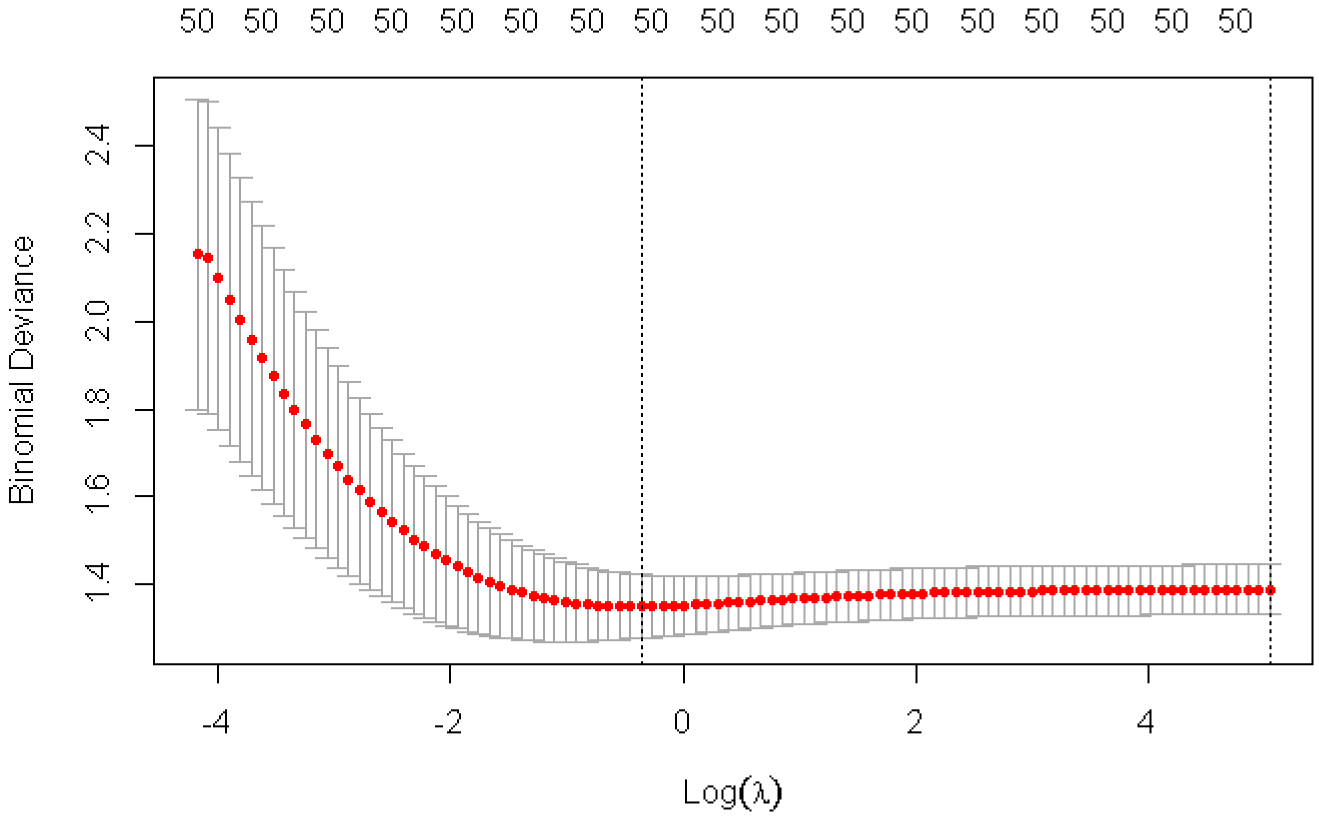

Figure 5.4: CV of Ridge plot

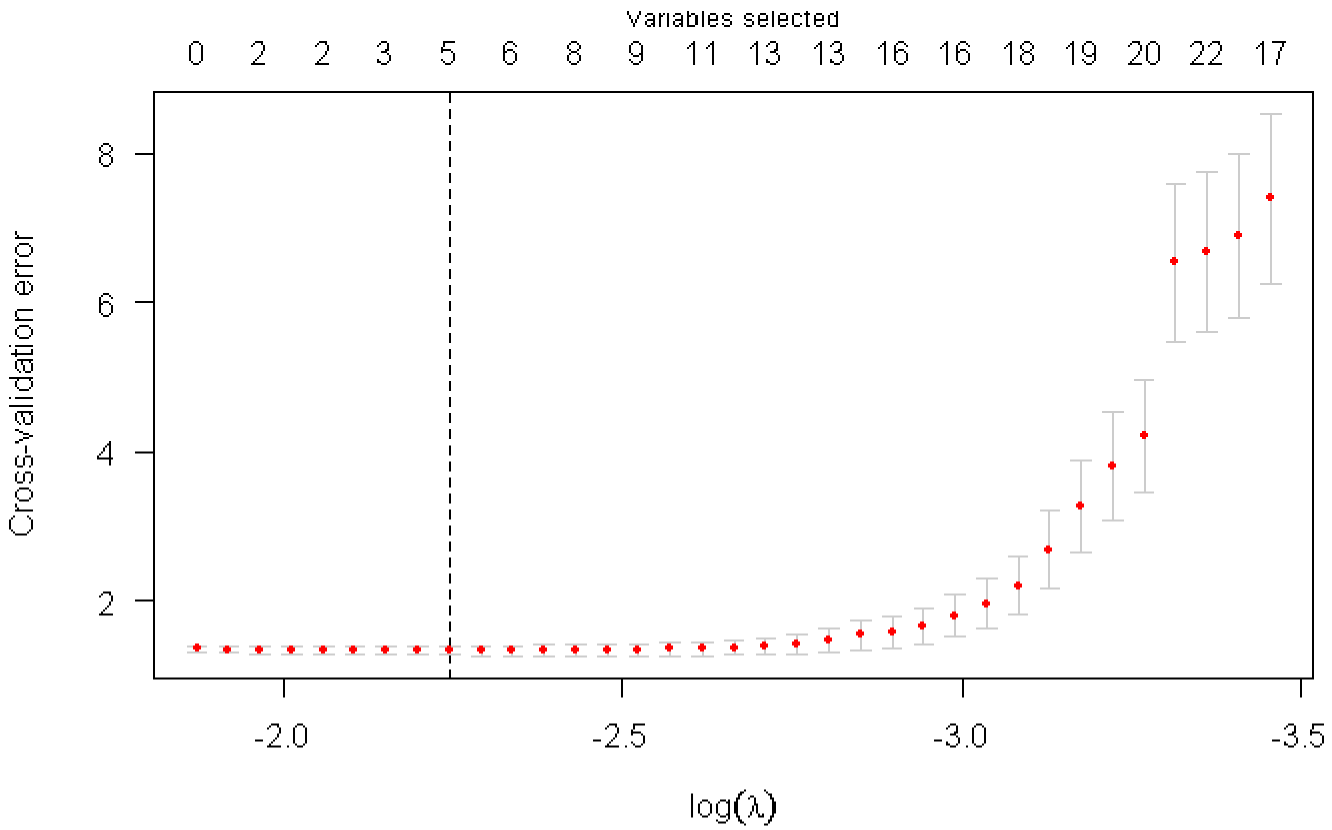

Figure 5.5: SCAD plot for number of selected variables

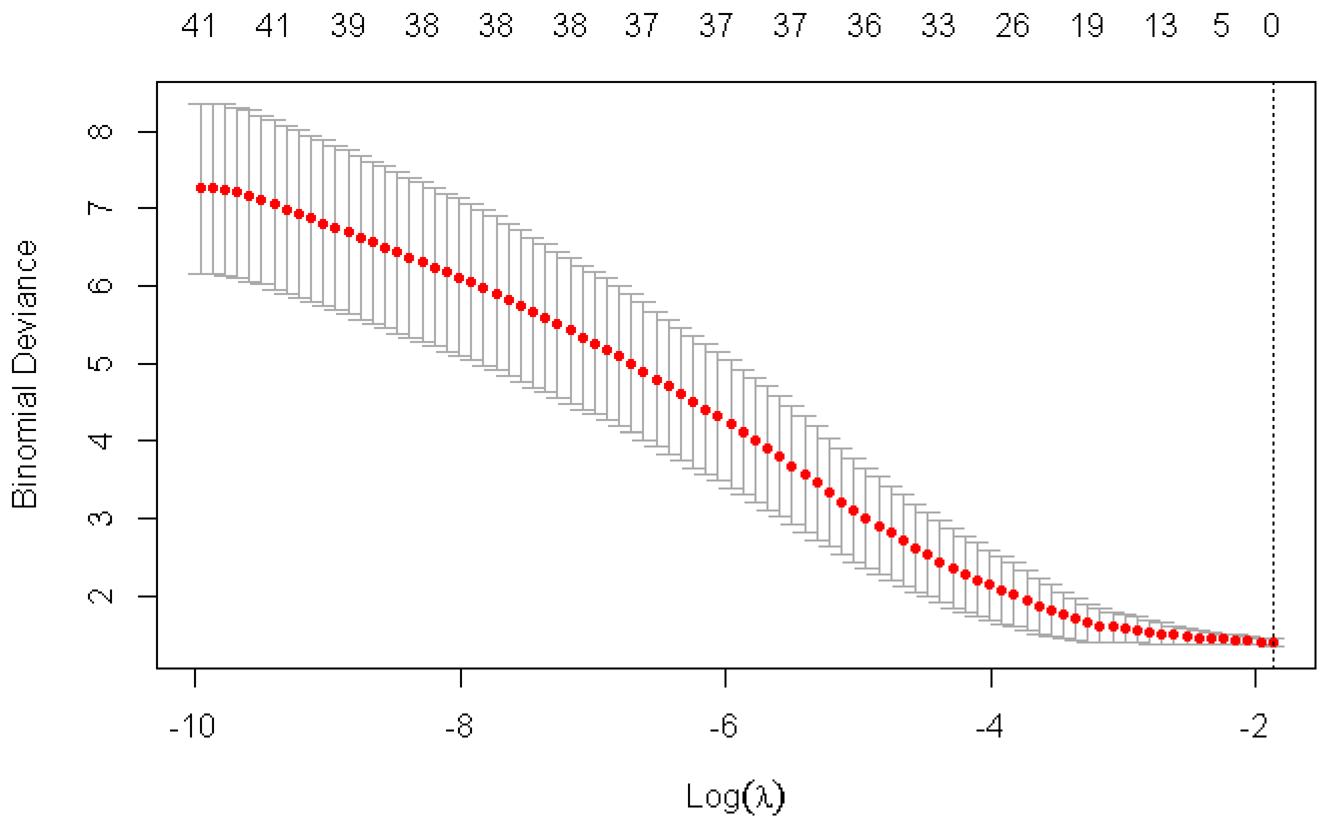

Figure 5.6: CV of Lasso plot

From the above diagrams,we realized that the ridge regression selects more variables as compared to other methods but might not produce the best accuracy results. For the classification performance, the Adaptive Lasso with the proposed weight gave a better result than other methods.

5.4 Real Data Analysis

We analyzed the leukemia data (Golub et al, 1999), and the colon data (Alon et al, 1999) to test the performance of the methods. The colon data contains 62 samples, which included 40 colon tumors and 22 normal colon tissue samples and 2000 genes, whereas the leukemia data contains 72 samples, which included 23 leukemia cancer and 49 normal tissue samples and 7129 genes. We estimated the average accuracy, AUROC, and G-mean using the various methods mentioned above. The results are reported in Table 1. To assess the performance of Lasso, Alasso, Elastic Net and also the proposed method, we analyzed colon cancer gene expression data. The dataset contains 62 samples, which included 40 colon tumors and 22 normal colon tissue samples and 2000 genes whose gene expression information was extracted from DNA microarray data resulting from preprocessing; all 2,000 genes have unique expressed tags (ESTs) named. We also analyzed leukemia cancer gene expression data. The dataset includes 72 samples, which included 23 leukemia cancer and 49 normal tissue samples and 7129 genes. These data sets are examples of high-dimensional data where the number of predictors, p, is much greater than the number of observations, n. The data set is an already prepared data set thus the gene expressions are normalized with no missing values.

5.5 Leukemia Cancer Data Analysis

We analyzed the leukemia data to test the performance of the methods. The leukemia data contains 72 samples, which included 23 leukemia cancer and 49 normal tissue samples and 7129 genes. We estimated the average accuracy, AUROC, and G-mean using the various methods mentioned above.

Table 5.5: Classification performance for leukemia

Method	Accuracy	G-Mean	AUROCs
Lasso	0.9545455	0.9660745	0.9375
Alasso	1.00	1.00	1.00
Elastic Net	0.9545455	0.9660745	0.9375

From the output (Table 5.5) below, we can clearly see that all the methods produced higher accuracy, Geometric Mean and Area under the Curve. The Proposed Adaptive Lasso performed better than other existing methods with Accuracy, G-Mean and AUROC of 1. From the output below, both specificity and sensitivity have a value of 1 respectively.

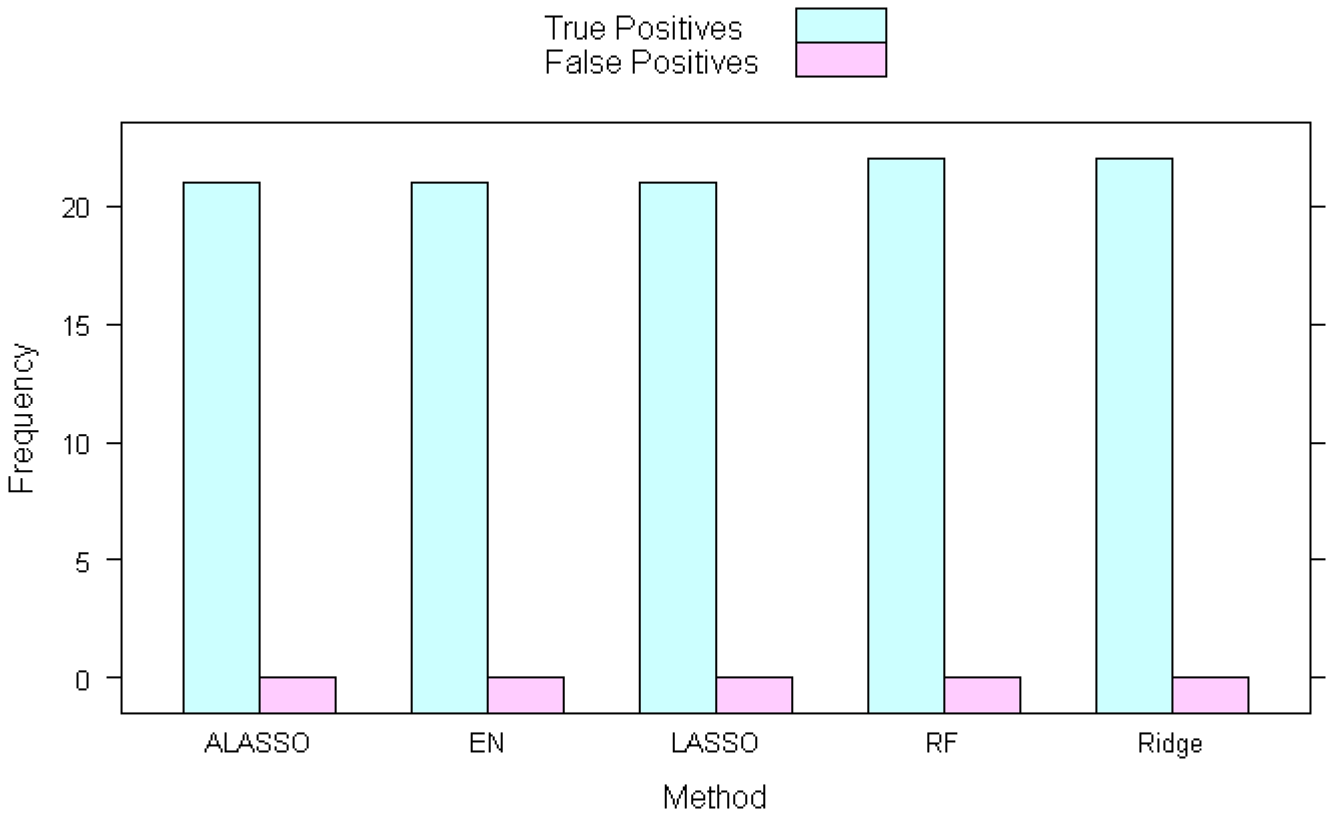

Figure 5.7: Bar Chart of True Positives, False Positives against Method for Leukemia data.

We can clearly see from Figure 5.7, all methods used produced higher True Positives than False Positives meaning the cases we predicted Yes (they have the disease) and they do actually have the disease. Nevertheless, all the methods performed equally well by recording every low number of False Positives.

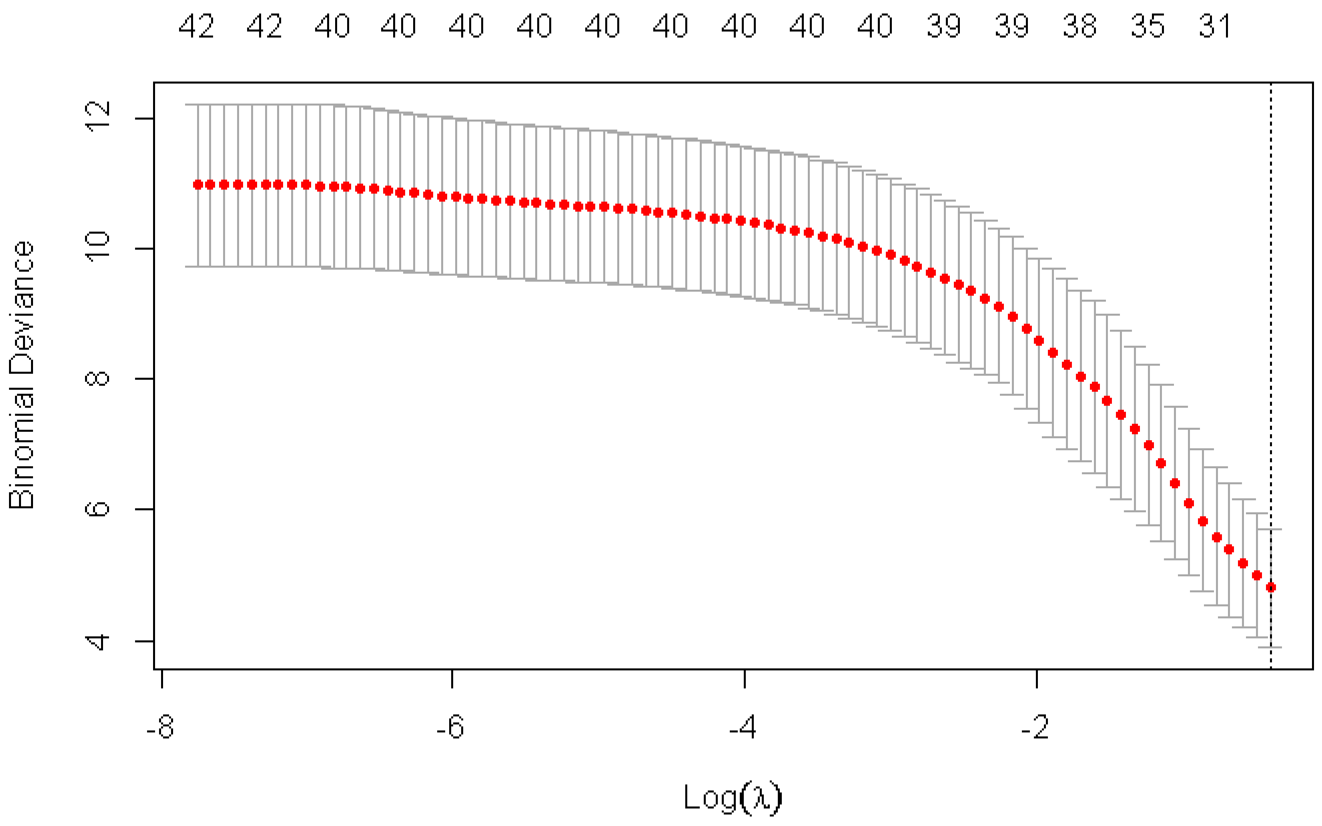

Figure 5.8: Penalty vs CV MSE Plot

5.6 Colon Cancer Data Analysis

We analyzed the colon data to test the performance of the methods. The colon data contains 62 samples, which included 40 colon tumors and 22 normal colon tissue samples and 2000 genes. We estimated the average accuracy, AUROC, and G-mean using the various methods mentioned above.

Table 5.6: Classification performance for Colon Cancer

Method	Accuracy	G-Mean	AUROC
Lasso	0.8421053	0.7464315	0.8083
Alasso	0.85	0.84	0.83
Elastic Net	0.8421053	0.7464315	0.8083

From the above output (Table 5.6), we can clearly see that all the methods produced higher accuracy, Geometric Mean and Area under the Curve. The Proposed Adaptive Lasso performed better than other existing methods with Accuracy, G-Mean and AUROC.

Table 5.7: Misclassification rate and Number of selected genes for Colon Cancer

Method	Number of Selected Genes	Misclassification rate
Lasso	17	0.263
Alasso	16	0.210

Table 5.8: Misclassification rate and Number of selected genes for Prostate Cancer

Method	Number of Selected Genes	Misclassification rate
Lasso	26	0.129
Alasso	26	0.096

Alasso selected a few number of selected genes as compared to Lasso but it gave a very low misclassification rate of 0.210 for the Colon cancer data. From Table 5.8, Both Lasso and Alasso selected the same number of selected genes but Alasso had a smaller rate of misclassification.

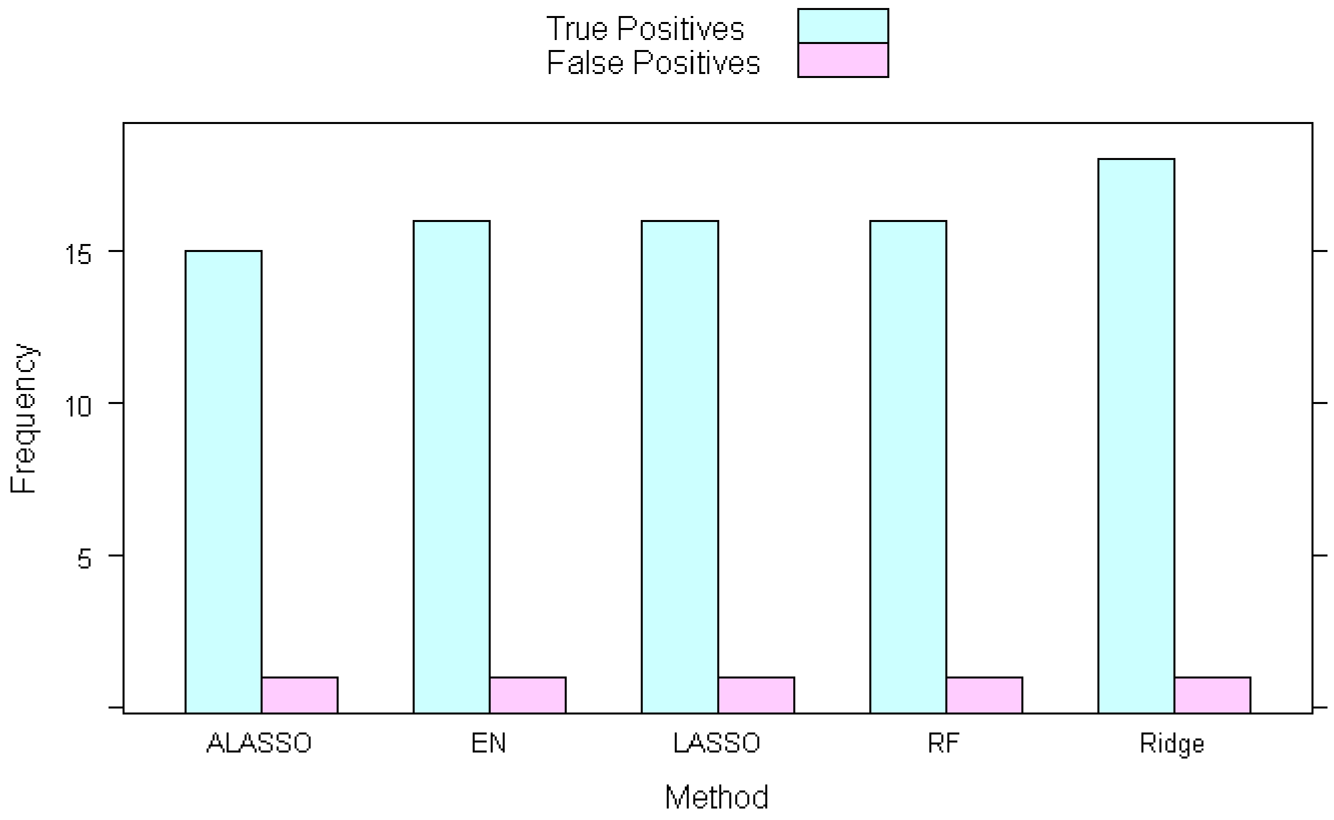

Figure 5.9: Bar Chart of True Positives, False Positives against Method for Colon data

We evaluate and assess the prediction accuracy by the Area Under the Receiver Operating Characteristic curve (AUROC) which is simply a plot of the true positive rate (TPR) against the false positive rate (FPR). We can clearly see from Figure 5.9, all used methods produced a higher True Positive rate than False Positive rate meaning that, the cases we predicted Yes (they have the disease) and they do have the disease. Nevertheless, all the methods performed equally well by recording every low number of False Positives.

We used all the above mentioned methods to analyzed the same set of data to compare their classification performance. We observed from Table 1 that the performance of the methods increased with increased sample size (sample size of colon data = 62, sample size of leukemia = 72) which was expected. As the sample size increases, the performance metric also improves.

Chapter 6

Discussion and Conclusion

6.1 Summary

In genomic studies, we encounter a number of genes which is usually much larger than the sample size. Biologically, only a small subset of these genes are related to a disease under study. Most of the genes are unrelated and have noise which can greatly influence the performance of classification. Therefore, choosing at least a minimal gene list is one of the most important applications in the analysis of classification data, which are effective in complex diseases. The essence of variable screening is to screen out these genes which might be unrelated to the disease under study greatly reducing the dimensionality of the classification data. These screened variables are further applied to various shrinkage methods to further select those genes that are important or related the disease under study. The aim is to select a subset of useful and appropriate genes to diagnose the disease among the whole genes. This improves the accuracy of classification. It has been shown extensively from numerical studies that the proposed method for adaptive lasso can be a very useful variable screening method to discover all relevant predictors, even if the number of predictors is larger than the sample size. Our proposed method was experimentally tested and compared with other existing methods based on our well-know gene expression datasets. The classification performance of the proposed method was shown through three aspects: high classification accuracy, G-Mean and AUC. These three metrics nominates the proposed method as an efficient gene selection method. A small number of selected genes significantly improves the interpretability of the model, its applicability and usefuleness in other high-dimensional data.

6.2 Recommendations for Future Work

For recommendations for future work, we first of all address the limitations and strengths of our proposed method and then we propose a possible solution as future work. We would also like to apply the proposed method to survival analysis using the Cox proportional hazard model. Survival analysis deals analyzing the expected duration of time until one or more events happen, such as death in biological organisms. The purpose is to analyze data in which the time until the event is of interest. The response is often referred to as a failure time, survival time, or event time. We will apply it to a real world survival analysis data set and see if the proposed method performs well.

Chapter 7

Appendix

R Codes

```
Attached is the R codes.

library(class)

library(plyr)

library(caret)

library(ggplot2)

library(glmnet)

library(SIS)

library(tidyverse)

library(caret)

library(glmnet)

library(randomForest)

library(graphics)

library(SIS)

#install.packages("graphics")

#install.packages("ncvreg")

library(ncvreg)




## Set directorate and load data
```

```
setwd("/Users/UTEPCSS/Desktop/dataset")

colon_x <- read.csv("colon_2000_62_x.csv", sep = ',', header = T)
colon_y <- read.csv("colon_2000_62_y.csv", sep = ',', header = T)
colon <- cbind(colon_x, colon_y)
leukemia_x <- read.csv("leukemia_7129_72_x.csv", sep = ',', header = T)
leukemia_y <- read.csv("leukemia_7129_72_y.csv", sep = ',', header = T)
leukemia <- cbind(leukemia_x, leukemia_y)

## Split data into training and testing set
# splits the data into training and testing by ratio 0.7
sampleColon <- sample(nrow(colon), 0.7*nrow(colon), replace = FALSE)
sampleleukemia <- sample(nrow(leukemia), 0.7*nrow(leukemia), replace = FALSE)
# train and test set
trainColon <- as.matrix(colon[sampleColon,])
trainColon_x <- trainColon[,-2001]
trainColon_y <- trainColon[,2001]
testColon <- as.matrix(colon[-sampleColon,])
testColon_x <- testColon[,-2001]
testColon_y <- testColon[,2001]
trainleukemia <- as.matrix(leukemia[sampleleukemia,])
trainleukemia_x <- trainleukemia[,-7130]
trainleukemia_y <- trainleukemia[,7130]
testleukemia <- as.matrix(leukemia[-sampleleukemia,])
testleukemia_x <- testleukemia[,-7130]
testleukemia_y <- testleukemia[,7130]
x.train4=as.data.frame(trainleukemia_x)
x.train5=as.data.frame(trainColon_x)
```

```
###################################################################
### leukemia Data

## Penalised Logistic Regression
### leukemia Data

##building a logistic model
logistic_model = glm(trainleukemia_y ~., family= binomial(link="logit")

, data= x.train4)
summary(logistic_model)

##finding the regession coefficient and standard errors
mg<-summary(logistic_model)
mg

a<-coef(mg)
a[,1] ## average regression coefficient
a[,2] ## average standard error


##finding the weights
weight1= a[,1]/a[,2]  ##weights
weight1
```

```
library(glmnet)
### ALASSO
cv.alasso<- cv.glmnet(trainleukemia_x,trainleukemia_y, family = "binomial",

alpha = 1, penalty.factor = 1 / (weight1))

mod_alasso <- glmnet(trainleukemia_x,trainleukemia_y,family = "binomial",

alpha = 1, penalty.factor = 1 / (weight1))
# Make predictions on the test data
x.test <- model.matrix(x ~., as.data.frame(testleukemia))[,-1]
probabilities_alasso <- predict(mod_lasso, x.test, type = "class")
# Model accuracy
accuracy_alasso <- mean(probabilities_alasso == testleukemia_y)
accuracy_alasso

## Penalty vs CV MSE plot
plot(cv.alasso)



# LASSO
# Find the best lambda using cross-validation
cv.lasso <- cv.glmnet(trainleukemia_x, trainleukemia_y, alpha = 1,
                      family = "binomial")
# Fit the final model on the training data
mod_lasso <- glmnet(trainleukemia_x, trainleukemia_y, alpha = 1,
                    family = "binomial", lambda = cv.lasso$lambda.min)
```

```
# Make predictions on the test data
x.test <- model.matrix(x ~., as.data.frame(testleukemia))[,-1]
probabilities_lasso <- predict(mod_lasso, x.test, type = "class")
# Model accuracy
accuracy_lasso <- mean(probabilities_lasso == testleukemia_y)
accuracy_lasso

# Elastic Net
# Find the best lambda using cross-validation
cv.eln <- cv.glmnet(trainleukemia_x, trainleukemia_y, alpha = 0.5,
                    family = "binomial")
# Fit the final model on the training data
mod_eln <- glmnet(trainleukemia_x, trainleukemia_y, alpha = 0.5,
                  family = "binomial", lambda = cv.eln$lambda.min)
# Make predictions on the test data
x.test <- model.matrix(x ~., as.data.frame(testleukemia))[,-1]
probabilities_eln <- predict(mod_eln, x.test, type = "class")
# Model accuracy
accuracy_eln <- mean(probabilities_eln == testleukemia_y)
accuracy_eln

# Ridge
# Find the best lambda using cross-validation
cv.ridge <- cv.glmnet(trainleukemia_x, trainleukemia_y, alpha = 0,
                      family = "binomial")
# Fit the final model on the training data
mod_ridge <- glmnet(trainleukemia_x, trainleukemia_y, alpha = 0,
                    family = "binomial", lambda = cv.ridge$lambda.min)
```

```
# Make predictions on the test data
x.test <- model.matrix(x ~., as.data.frame(testleukemia))[,-1]
probabilities_ridge <- predict(mod_ridge, x.test, type = "class")
# Model accuracy
accuracy_ridge <- mean(probabilities_ridge == testleukemia_y)
accuracy_ridge



# Penalized regression
roc(probabilities_lasso, testleukemia_y)$auc
roc(probabilities_eln, testleukemia_y)$auc
roc(probabilities_ridge, testleukemia_y)$auc
roc(probabilities_alasso, testleukemia_y)$auc

### Confusion matrix

# Penalized regression
confusionMatrix(table(probabilities_lasso, testleukemia_y), positive = "1")
confusionMatrix(table(probabilities_eln, testleukemia_y), positive = "1")
confusionMatrix(table(probabilities_ridge, testleukemia_y), positive = "1")
confusionMatrix(table(probabilities_alasso, testleukemia_y), positive = "1")

sen_leukemia <- c(1,1,1,1,1)
spec_leukemia <- c(0.9333, 1.0, 0.9333, 0.9333, 1.0)
g_mean_leukemia <- sqrt(sen_leukemia*spec_leukemia)
method <- c("ALASSO", "RF", "LASSO", "EN", "Ridge")
FP <- c(0,0,0,0,0)
TP <- c(21,22,21,21,22)
```

```
data <- data.frame(method, FP, TP)


library(lattice)

barchart(TP+FP ~ method, data = data, ylab = "Frequency", xlab = "Method",

         auto.key = list(text= c("True Positives", "False Positives")))



###################################################################

### Colon Data


## Penalised Logistic Regression


library(glmnet)

##building a logistic model

logistic_model1 = glm(trainColon_y ~., family= binomial(link="logit"), data= x.train5)

summary(logistic_model1)


##finding the regession coefficient and standard errors

mb<-summary(logistic_model1)



b<-coef(mb)

b[,1] ## regression coefficient

b[,2] ## standard error



##finding the weights

weight2= b[,1]/b[,2]  ##weights
```

```
weight2


library(glmnet)
### ALASSO
cv.alasso<- cv.glmnet(trainColon_x,trainColon_y, family = "binomial",
alpha = 1,penalty.factor = 1 / (weight2))

mod_alasso <- glmnet(trainColon_x,trainColon_y,family = "binomial",
alpha = 1,penalty.factor = 1 / (weight2))
# Make predictions on the test data
x.test <- model.matrix(x ~., as.data.frame(testColon))[,-1]
probabilities_alasso <- predict(mod_lasso, x.test, type = "class")
# Model accuracy
accuracy_alasso <- mean(probabilities_alasso == testColon_y)
accuracy_alasso

## Penalty vs CV MSE plot
plot(cv.alasso)



# LASSO
# Find the best lambda using cross-validation
cv.lasso <- cv.glmnet(trainColon_x, trainColon_y, alpha = 1, family = "binomial")
# Fit the final model on the training data
mod_lasso <- glmnet(trainColon_x, trainColon_y, alpha = 1, family = "binomial",
                    lambda = cv.lasso$lambda.min)
# Make predictions on the test data
```

```
x.test <- model.matrix(x ~., as.data.frame(testColon))[,-1]
probabilities_lasso <- predict(mod_lasso, x.test, type = "class")
# Model accuracy
accuracy_lasso <- mean(probabilities_lasso == testColon_y)
accuracy_lasso

# Elastic Net
# Find the best lambda using cross-validation
cv.eln <- cv.glmnet(trainColon_x, trainColon_y, alpha = 0.5, family = "binomial")
# Fit the final model on the training data
mod_eln <- glmnet(trainColon_x, trainColon_y, alpha = 0.5, family = "binomial",
                  lambda = cv.eln$lambda.min)
# Make predictions on the test data
x.test <- model.matrix(x ~., as.data.frame(testColon))[,-1]
probabilities_eln <- predict(mod_eln, x.test, type = "class")
# Model accuracy
accuracy_eln <- mean(probabilities_eln == testColon_y)
accuracy_eln

# Ridge
# Find the best lambda using cross-validation
cv.ridge <- cv.glmnet(trainColon_x, trainColon_y, alpha = 0, family = "binomial")
# Fit the final model on the training data
mod_ridge <- glmnet(trainColon_x, trainColon_y, alpha = 0, family = "binomial",
                    lambda = cv.ridge$lambda.min)
# Make predictions on the test data
x.test <- model.matrix(x ~., as.data.frame(testColon))[,-1]
probabilities_ridge <- predict(mod_ridge, x.test, type = "class")
```

```
# Model accuracy
accuracy_ridge <- mean(probabilities_ridge == testColon_y)
accuracy_ridge

# random forest
roc(probabilities_rf, testColon_y)$auc
# Penalized regression
roc(probabilities_lasso, testColon_y)$auc
roc(probabilities_eln, testColon_y)$auc
roc(probabilities_ridge, testColon_y)$auc
roc(probabilities_alasso, testColon_y)$auc

### Confusion matrix
library(caret)
library(e1071)
# random forest
confusionMatrix(table(probabilities_rf ,testColon_y), positive = "1")
# Penalized regression
confusionMatrix(table(probabilities_lasso, testColon_y), positive = "1")
confusionMatrix(table(probabilities_eln, testColon_y), positive = "1")
confusionMatrix(table(probabilities_ridge, testColon_y), positive = "1")
confusionMatrix(table(probabilities_alasso, testColon_y), positive = "1")

sen_colon <- c(0.9286,0.9286,0.9286,0.9286,0.9286)
spec_colon <- c(0.4, 0.6, 0.6, 0.6, 1.0)
g_mean_colon <- sqrt(sen_colon*spec_colon)
method <- c("ALASSO", "RF", "LASSO", "EN", "Ridge")
FP <- c(1,1,1,1,1)
```

~~~
TP <- c(15,16,16,16,18)
data <- data.frame(method, FP, TP)

library(lattice)
barchart(TP+FP ~ method, data = data, ylab = "Frequency", xlab = "Method",
         auto.key = list(text= c("True Positives", "False Positives")))

##############
---
title: "stimu"
author: "owusu stuff 1"
date: "6/6/2020"
output: html_document
---

```{r}
install.packages("tidyverse")
install.packages("caret")
library(tidyverse)
library(caret)
library(glmnet)
#Logisitic regression model
set.seed(1000) # Set seed for reproducibility

#results <- matrix(nrow = 100, ncol = 4)
#for (i in 1:100) {
 ##results[i,2] <- # accuracy value
  #results[i,3] <- # AUC value
~~~

```
  #results[i,4] <- # s value
#}
#colMeans(results, na.rm = T)

n = 100 # Number of observations
m = (n/log10(n)) # Number of predictors included in model
x <- matrix(rnorm(n*m), nrow=n, ncol=m)
y= as.matrix(rbinom(n= 100 , size= 1, prob =0.5))
dim(x)

dat <- data.frame(x,y)
dim(dat)

#Logistic gregression
# Split data into train (70) and test (30) sets
## SPLITTING OF DATA INTO TRAINING AND TESTING
train_rows <- sample(1:n, 0.7*n)
traindat <- as.matrix(dat[train_rows,])
testdat <- as.matrix(dat[-train_rows,])
x.train <- x[train_rows, ]
x.test <- x[-train_rows, ]

y.train <- y[train_rows]
y.test <- y[-train_rows]

x.train2= as.data.frame(x.train) # change to  train data to a dataframe
y.train2= as.data.frame(y.train)
```

```
x.train3= as.matrix(x.train)
y.train3= as.matrix(y.train)

##building a logistic model
logistic_model = glm(y.train ~., family= binomial(link="logit"), data= x.train2)
summary(logistic_model)

##finding the regession coefficient and standard errors
mg<-summary(logistic_model)
mg

a<-coef(mg)
a[,1] ## regression coefficient
a[,2] ## standard error


##finding the weights
weight1= a[,1]/a[,2]  ##weights
weight1



‘‘‘



‘‘‘{r}
## Penalised Logistic Regression
```

```
#Adaptive lasso
#install.packages("glmnet")
library(glmnet)
require(glmnet)

## Perform adaptive LASSO
alasso3 <- glmnet(x=as.matrix(x.train),y= as.matrix(y.train),
                  ## Multinomial regression
                  family = "binomial",
                 alpha = 1,penalty.factor = 1 / (weight1))
plot(alasso3, xvar = "lambda")



## Perform adaptive LASSO with 10-fold CV
alasso4_cv <- cv.glmnet(x=as.matrix(x.train),y= as.matrix(y.train),
type.measure = "deviance", nfold = 10,
                     ## Multinomial regression
                     family = "binomial",
                     alpha = 1,
                     penalty.factor = 1 /weight1,
                     keep = TRUE)
alasso4_cv
## Penalty vs CV MSE plot
plot(alasso4_cv)
alasso4_cv$lambda.min
```

```
## s: Value(s) of the penalty parameter ‘lambda’ at which
##    predictions are required. Default is the entire sequence used
##    to create the model.
coef(alasso4_cv, s = alasso4_cv$lambda.min)




##
pred.probs_3<- as.numeric(predict(alasso4_cv,
 newx = x.test, s = alasso4a_cv$lambda.min, type = "response"))
pred.probs_3
alasso4a_cv.pred <- ifelse(pred.probs_3 > 0.5, "1", "0")
alasso4a_cv.pred

Yactual <- as.matrix(y.test)
confusion_matrix <- ftable(Yactual, alasso4a_cv.pred)
confusion_matrix
accuracy <- 100* (sum(diag(confusion_matrix)) / length(alasso4_cv.pred))
accuracy

‘‘‘



‘‘‘{r}
## ridge logistics regression

ridge <- glmnet( x=as.matrix(x.train),y= as.matrix(y.train),
                 ## Binary logistic regression
                 family = "binomial",

 ## ‘alpha = 1’ is the lasso penalty, and ‘alpha = 0’ the ridge penalty.
                 alpha = 0)

ridge
plot(ridge, xvar = "lambda")

## Perform ridge regression with 10-fold CV
ridge_cv <- cv.glmnet( x=as.matrix(x.train),y= as.matrix(y.train) ,

type.measure = "deviance",family = "binomial",nfold = 10,alpha = 0)

ridge_cv
## Penalty vs CV MSE plot
plot(ridge_cv)




###alternatively;
set.seed(999)
# Perform ridge regression
ridge3 <- glmnet( x=as.matrix(x.train),y= as.matrix(y.train),
                 ## Binary logistic regression
                 family = "binomial",
                 ## ‘alpha = 1’ is the lasso penalty,
                  and ‘alpha = 0’ the ridge penalty.
                 alpha = 0)
```

```
plot(ridge3, xvar = "lambda")



## Perform ridge regression with 5-fold CV
ridge3_cv <- cv.glmnet( x=as.matrix(x.train),y= as.matrix(y.train) ,

type.measure = "deviance",family = "binomial",
                      alpha = 0)
## Penalty vs CV MSE plot
plot(ridge3_cv)

## Extract coefficients at the error-minimizing lambda
ridge3_cv$lambda.min
## s: Value(s) of the penalty parameter ‘lambda’ at which
##    predictions are required. Default is the entire sequence used
##    to create the model.
(best_ridge_coef3 <- coef(ridge3_cv, s = ridge3_cv$lambda.min))

## The intercept estimates should be dropped.
best_ridge_coef3 <- as.numeric(best_ridge_coef3)[-1]






###alternatively
lambdas <- 10^seq(2, -3, by = -.1)
ridge_reg = glmnet(x=as.matrix(x.train),y= as.matrix(y.train), nlambda = 25,
 alpha = 0, family = 'gaussian', lambda = lambdas)

summary(ridge_reg)
cv_ridge <- cv.glmnet(x=as.matrix(x.train),y= as.matrix(y.train),

alpha = 0, lambda = lambdas)
optimal_lambda <- cv_ridge$lambda.min
optimal_lambda

# Compute R^2 from true and predicted values
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))


  # Model performance metrics
data.frame(
  RMSE = RMSE,
  Rsquare = R_square
)

}

# Prediction and evaluation on train data
predictions_train <- predict(ridge_reg, s = optimal_lambda, newx = x)
eval_results(y.train, predictions_train, traindat)
```

```
# Prediction and evaluation on test data
predictions_test <- predict(ridge_reg, s = optimal_lambda, newx = x.test)
eval_results(y.test, predictions_test, testdat)

```
```

```
```{r}
## Elastic Net Logistic regression model
elasticnet <- cv.glmnet(x=as.matrix(x.train),y= as.matrix(y.train),

family = "binomial", type.measure = "class", nfolds=10, alpha=0.5, nlambda = 100)

plot(elasticnet)

predict_validation <- predict(elasticnet,

newx = as.matrix(x.test), s = c(elasticnet$lambda.min), type = "class")
predict_validation

fit  <- cv.glmnet(x=as.matrix(x.train), y=as.matrix(y.train),

family="binomial", type.measure="class", alpha=0.1)
fit
```
```
```

~~~
```{r}

library(glmnet)
cv.lasso <- cv.glmnet(x=as.matrix(x.train),y= as.matrix(y.train),

alpha = 1, family = "binomial")
plot(cv.lasso)
cv.lasso$lambda.min
cv.lasso$lambda.1se
coef(cv.lasso, cv.lasso$lambda.min)
coef(cv.lasso, cv.lasso$lambda.1se)



# Find the best lambda using cross-validation
set.seed(123)
cv.lasso <- cv.glmnet(x=as.matrix(x.train),y= as.matrix(y.train),

alpha = 1, family = "binomial")
# Fit the final model on the training data
model <- glmnet(x=as.matrix(x.train),y= as.matrix(y.train),

alpha = 1, family = "binomial",lambda = cv.lasso$lambda.min)
# Display regression coefficients
coef(model)
~~~

```
# Make predictions on the test data
x.test <- model.matrix(y ~., test.data)[,-1]
probabilities <- model %>% predict(newx = x.test)
predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Model accuracy
observed.classes <- test.data$diabetes
mean(predicted.classes == observed.classes)
```

```{r}
##### SCAD
install.packages("ncvreg")
library(ncvreg)
cvfit.SCAD <- cv.ncvreg(X=as.matrix(x.train),y= as.matrix(y.train), nfolds=5,

family="binomial", penalty="SCAD", lambda.min=.01, nlambda=100, eps=.01, max.iter=1000)

## Warning in ncvreg(X = X, y = y, ...): Maximum number of iterations reached
# USING THE ARGUMENT penalty="MCP" TO CHOOSE AMONG DIFFERENT PENALTY FUNCTIONS
plot(cvfit.SCAD)



result.SCAD <- cvfit.SCAD$fit
beta.hat <- as.vector(result.SCAD$beta[-1, cvfit.SCAD$min])
cutoff <- 0
terms <- colnames(x=as.matrix(x.train))[abs(beta.hat) > cutoff]; terms
```

```
formula.SCAD <- as.formula(paste(c("y~ 1", terms), collapse=" + "))
fit.SCAD <- glm(formula.SCAD, data =dat, family="binomial")
summary(fit.SCAD)

BIC(fit.SCAD)
##

#fit.SCAD
#Predicted Probabilities Using JACKKNIFE
n <- NROW(dat)
p.scad.jk <- rep(0, n)
for (i in 1:n){
print(i)
fit.i <- glm(formula(fit.SCAD), data=dat[-i,], family = "binomial")
p.scad.jk[i] <- predict(fit.i, newdata=dat[i,], type="response")
}
roc.SCAD <- plot.roc(y= as.matrix(y.train),

p.scad.jk, main="(b) ROC for fit.SCAD", percent=TRUE,

 print.auc=TRUE, print.auc.cex=1.5, col="blue")
```
‘‘‘

```
‘‘‘{r}
#to fit a penalized logistic regression model,
library(ncvreg)
fit <- ncvreg(X=as.matrix(x.train),y= as.matrix(y.train), family=’binomial’)
summary(fit, lambda=0.05)
cvfit <- cv.ncvreg(X=as.matrix(x.train),y= as.matrix(y.train), family=’binomial’)
#par(mfrow=c(2,2))
plot(cvfit, type=’all’)
‘‘‘
###########
X <- leukemia$x
y <- leukemia$y
# This is the model fitting step.
r <- system.time(fit.glmnet <-
       glmnet(X,y,family = "binomial",lambda = lambda,alpha = alpha))
cat(sprintf("Model fitting took %0.2f seconds.\n",r["elapsed"]))


# This is the cross-validation step.
r <- system.time(out.cv.glmnet <-
       cv.glmnet(X,y,family = "binomial",type.measure = "class",
                 alpha = alpha,nfolds = nfolds,lambda = lambda))
lambda <- out.cv.glmnet$lambda
cat(sprintf("Cross-validation took %0.2f seconds.\n",r["elapsed"]))


# Choose the largest value of lambda that is within 1 standard error
# of the smallest misclassification error.
```

~~~
lambda.opt <- out.cv.glmnet$lambda.1se

cat("classification results with lambda = ",lambda.opt,":\n",sep="")
y.glmnet <- c(predict(fit.glmnet,X,s = lambda.opt,type = "class"))
print(table(true = factor(y),pred = factor(y.glmnet)))



```{r}
##SCAD REGRESSION
fit <- ncvreg(X= trainleukemia_x, y=trainleukemia_y,

family="binomial", penalty="SCAD")
summary(fit, lambda=0.5)
plot(fit, main=expression(paste("SCAD, ",gamma,"=",3)))
op <- par(mfrow=c(2,2))
fit <- ncvreg(X= trainleukemia_x, y=trainleukemia_y, family="binomial", alpha=0.5)
plot(fit, main=expression(paste(alpha,"=",0.5)))
par(op)



cvfit.SCAD <- cv.ncvreg(X=trainleukemia_x,y= trainleukemia_y, nfolds=5,

family="binomial", penalty="SCAD", lambda.min=.01, nlambda=100, eps=.01, max.iter=1000)

## Warning in ncvreg(X = X, y = y, ...): Maximum number of iterations reached
# USING THE ARGUMENT penalty="MCP" TO CHOOSE AMONG DIFFERENT PENALTY FUNCTIONS
plot(cvfit.SCAD)
~~~

```
### MCP REGRESSION

fit <- ncvreg(X= trainleukemia_x, y=trainleukemia_y, family="binomial", penalty="MCP")



cvfit.MCP <- cv.ncvreg(X=trainleukemia_x,y= trainleukemia_y, nfolds=5,


family="binomial", penalty="MCP", lambda.min=.01, nlambda=100, eps=.01, max.iter=1000)
```

References

Algamal, Z. and Lee M. (2015). High dimensional logistic regression model using adjusted elastic net penalty. *Journal of Statistics*, 34: 667–676.

Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D. and Levine, A.J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA*, 96(12): 6745–6750.

Alonso-González, C. and Moro-Sancho, I. (2012). Microarray gene expression classification with few genes: Criteria to combine attribute selection and classification methods. *Expert Systems with Applications*, 39: 7270–7280.

Bellman, R. (1957). Curse of dimensionality and on the theory of dynamic programming. *Proc. Natl. Acad. Sci. USA*, 38: 716–719.

Chen, W. and Angelia, S. P. (2013). Classification consistency analysis for bootstrapping gene selection. *Journal of Statistics*, 39: 7270–7280.

Fan, F. Y. and Barut, E. (2014). Adaptive robust variable selection. *The Annals of Statistics*, 42: 324–351.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Annals of Statistics*, 96: 1348–1360.

Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Statistica Sinica, Journal of Statistics*, 79: 1348–1360.

Golub, T. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science. *The Annals of Statistics*, Vol. 286: 531-537.

Gordon, G. and Jensen, R. (2002). Translation of Microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Research*, Vol. 62: 4963-4967

Ing, C. and Lai, T. (2011). A stepwise regression method and consistent model selection for high dimensional sparse linear models. *Statistica Sinica*, pages 1473-1513.

Little, R. and Rubin, D. (1987). Statistical Analysis with Missing Data. *J. Wiley.*

Liu, Q. and Sung Andrew H. (2012). Penalized logistic regression with the adaptive lasso for gene selection in high-dimensional cancer classification. *Journal of Statistics.*

Piao, P. and Ryu, K. (2012). An ensemble correlation-based gene selection for cancer classification with gene expression data. *Bioinformatics*, 28: 3306-3315.

Pyle, D. (1999). Data Preparation for Data Mining. *Morgan Kaufmann.*

Singh, D. (2002) Gene expression correlates of clinical prostate cancer behavior *Cancer Research*, 1: 203–209.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(22): 267-288.

Tibshirani, R. (2013). Introduction to statistical learning. *Journal of the Royal Statistical Society*, 1(1): 67-82.

Zhang, C. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2): 894-942.

Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *The Annals of Statistics*, 36: 1509-1533.

Zoul, H. and Zhang, H. (2008). On the adaptive elastic-net with a diverging number of parameters. *The Annals of Statistics*, 37(4): 1733–1751.

Curriculum Vitae

Derrick Kwesi Bonney was born on February 23, 1992. The third son of Joseph Kwame Bonney and Faustina Akrong, he graduated from Saint Anthony High School in 2008. He entered the University of Cape Coast in the Fall of 2013 where he obtained his bachelors degree in Actuarial Science. He started working at Enterprise Life Insurance as a Chief Underwriting Officer and served as an auditor for Ransbet Supplies. He also passed 10 papers in Accountancy at the Institute of Chartered Accountant (Ghana). While pursuing his Bachelor's degree in Actuarial Science, he worked as a Teaching Assistant and organized Classes for Undergraduate Studies. He received his Bachelor's degree in Actuarial Science in the Summer of 2017. In the Fall of 2018, he entered the Graduate School of The University of Texas at El Paso. While pursuing a Master's degree in Statistics, he worked as a Teaching and Research Assistant at the Department of Mathematical Sciences. In the Fall of 2019, he did an internship with New York Life Insurance Company at El Paso, Texas. He is an active member of International Association of Black Actuaries, Institute of Chartered Accountant and American Statistical Association. He has participated in many Conferences such as Joint Statistical Meeting (JSM, 2019) in Denver, Colorado, WNAR 2019 in Santa Fe, New Mexico, StatFest 2019 at UT health medical center, Houston, Texas. Joint Statistical Meeting (Diversity and Mentoring - 2019) in Denver, International Association of Black Actuaries (2019) in Houston,Texas. 55th Southern Regional Council on Statistics (2018) in Kentucky.

Permanent address: 520 Los Angeles Drive
El Paso, Texas 79902