VARIOUS CONSIDERATIONS ON PERFORMANCE MEASURES FOR A CLASSIFICATION OF ORDINAL DATA

A THESIS

Presented to the Department of Mathematics and Statistics

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Applied Statistics

Committee Members:

Hojin Moon, Ph.D. (Chair)
Yong Hee Kim-Park, Ph.D.
Kagba Suaray, Ph.D.

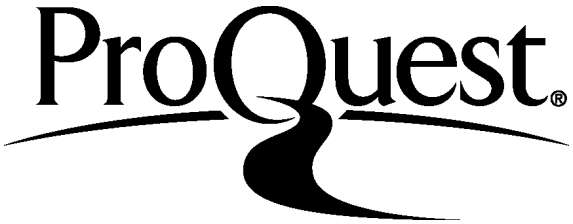
College Designee:

Tangan Gao, Ph.D.

By Denis Barasa Nyongesa

B.Ed., 2008, Moi University, Kenya

August 2016



ABSTRACT

VARIOUS CONSIDERATIONS ON PERFORMANCE MEASURES FOR A CLASSIFICATION OF ORDINAL DATA

By

Denis Barasa Nyongesa

August 2016

The technological advancement and the escalating interest in personalized medicine has resulted in increased ordinal classification problems. The most commonly used performance metrics for evaluating the effectiveness of a multi-class ordinal classifier include: predictive accuracy, Kendall's tau-b rank correlation, and the average mean absolute error (AMAE). These metrics are beneficial in the quest to classify multi-class ordinal data, but no single performance metric incorporates the misclassification cost. Recently, distance, which finds the optimal trade-off between the predictive accuracy and the misclassification cost was proposed as a cost-sensitive performance metric for ordinal data. This thesis proposes the criteria for variable selection and methods that accounts for minimum distance and improved accuracy, thereby providing a platform for a more comprehensive and comparative analysis of multiple ordinal classifiers. The strengths of our methodology are demonstrated through real data analysis of a colon cancer data set.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Dr. Hojin Moon, whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate his vast knowledge and skill in many areas. I would like to thank the other members of my committee, Dr. Yong Hee Kim-Park, and Dr. Kagba Suaray for the assistance they provided at all levels of the research project.

I would also like to thank my family for the support they provided me through my entire life and in particular, I must acknowledge my mum, Alice, without whose constant encouragement, I would not have finished this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Introduction

A gene is any discrete locus of heritable, genomic sequence which affects an organism's traits by being expressed as a functional product or by regulation of gene expression. The National Cancer Institute (NCI) defines gene expression profiles as the information about all messenger ribonucleic acids (RNA) that are made in various cell types. A gene expression profile may be used to find and diagnose a disease or condition and to see how well the body responds to treatment. Gene expression profiles may be used in precision medicine. A messenger RNA is a type of RNA found in cells. Messenger RNA (also called mRNA) molecules carry the genetic information needed to make proteins. They carry the information from the DNA in the nucleus of the cell to the cytoplasm where the proteins are made.

1.2. Overview of Cancer

Cancer has been recognized since early times, but treatment protocols and medications have lagged the initial observations of the disease by millennia (Xu, Cui, and Puett 2014). Although most cancers develop in the aging population, tragic childhood and teenage cancers are on the rise in the recent past. Epidemiological data show that, behind heart disease, cancer is the second leading cause of death worldwide, and many expect that in time, cancer will overtake heart disease as the leading cause of mortality (Xu, Cui, and Puett 2014).

The major advances in technology and contributions from medicine and biology has helped unravel many cancer complexities. Cancer has been considered by many investigators as a genetic disease, generally involving sequential random mutations and epigenetic changes. Cancer is now recognized as a very heterogeneous disease because even within the same type of

cancer, it may emerge that its origins can be attributable to a number of cases (Xu, Cui, and Puett 2014).

Cancer of the large bowel (colorectal cancer) is the third most common cancer in men and the second most common cancer in women worldwide. Despite recent advances in the screening, diagnosis, and treatment of colorectal cancer, an estimated 608,000 people die every year from this form of cancer—8% of all cancer deaths. The prognosis and treatment options for colorectal cancer depend on four pathological stages (I - IV), each of which has a different treatment option and five-year survival rate, so it is important that the stage is correctly identified.

1.2.1. Cancer Classification and Identification

Cancer is a family of diseases that share a common set of characteristics such as reprogrammed energy metabolism, uncontrolled cell growth, and tumor angiogenesis. Some cancer types developing in the same tissue may have distinct characteristics in terms of their growth patterns, malignance levels, survival rates and possibly even different underlying mechanisms. They may respond differently to the same drug treatment and hence have different mortality rates (Xu, Cui, and Puett 2014). Stewart and Kleihues (2003) stated that there are over 200 types of human cancers that have been identified and characterized, majority of which are determined based on the location, the originating cell type and cell morphology. It's now becoming increasingly evident that this type of classification, in large part subjective, is not adequate for the developing personalized treatment plans, which are becoming increasingly desirable and clearly represent the future of cancer medicine.

It is now feasible to classify cancers based on their molecular level information due to the rapid accumulation of high-throughput *omics* data for cancer, in particular, transcriptomic and

genomic data. This can be, for example, based on distinct expression patterns of certain genes or pathways shared only by samples of the same cancer type or combinations of mutations that tend to be selected to be more accurate (or co-occur) in certain cancer types. Such type-defining expression or mutation patterns of the genes are referred to as the *signature* of a cancer type.

1.2.2. Cancer Types, Grades and Stages

The earliest description of cancer can be traced back to 2500 BC by Egyptian physician Imhotep (Mukherjee 2010). Existing evidence suggests that the Egyptian physicians at the time could distinguish between benign and malignant tumors. The wide availability of the microscopes to physicians and surgeons in the nineteenth century led to the study of cancer as a scientific discipline. The early classification of cancer was primarily based on the cancer's location, such as lung cancer, skin cancer or blood cancer (leukemia). The oncologists, over time, began to realize that different cancers can develop from the same organ (Xu, Cui, and Puett 2014).

Yamagiwa and Ichikawa (1918), Japanese researchers, were the first to discover the multistage nature of cancer in the beginning of the twentieth century. For most of the cancers, the histological stage refers to the extent the cancer has spread, which typically is numbered from I through IV, with IV representing the most advanced stage. Since the treatment plan for cancer patients is often based on the stage of the disease, the stage is thus an important predictor of the patients' survival. Cancer stages are currently determined through pathological analysis of the biopsied specimens of the cancer tissue, including lymph nodes as well as analysis by imaging techniques with interpretation of the results by radiologists.

The pathologists, based on surgical specimens, also use cancer grade to represent the level of malignancy of a given cancer. The use of cancer grade is independent of the cancer's

type and stage. A popular grading system uses four grades: (1) G1 (highly differentiated), (2) G2 (Moderately differentiated), (3) G3 (Poorly differentiated), and (4) G4 (undifferentiated), with G4 representing the most malignant.

1.3. Ordinal Categorical Data

Ordered categorical data, or simply ordinal data, are commonplace in scientific disciplines where humans are used as measurement instruments. Ordinal classification is a multiclass problem where objects are classified into groups that have an inherent, natural ordering. In the recent years, the surge of personalized medicine and technological advancements in the genomics research has led to increased prevalence of ordinal data classification problems in clinical research.

The individual gene ranking methods perform gene selection through a univariate criterion function to provide a list of top ranked genes. However, the combination of the genes through individual gene-ranking may not produce a top ranked combination of genes because individual ranking tends to ignore redundancy and interaction among the genes. Such methods may result into low power when the co-regulation of multiple genes during tumor progression is not fully utilized (Zhang et al. 2012).

Several algorithms have been developed to classify ordinal data. Sanchez-Montero et al. (2011) proposed an algorithm that tries to build classifiers with simultaneously optimized accuracy, which is the rate of all correct predictions, and minimum sensitivity, which is the number of patterns correctly predicted to be in class *i* with respect to the total number of patterns in class *i*. They presented an efficient alternative to the Pareto based approach to train multi-class classifiers with a simultaneous improvement in accuracy and minimum sensitivity.

Kotsiantis and Pintelas (2014) proposed a cost-sensitive technique for ordinal data classification that takes into account the misclassification cost. They showed that the technique minimizes the distances between the actual and the predicted classes without causing harm but instead slightly improving the prediction accuracy. The technique basically choses the class with the minimum conditional risk. The method does not require any modification of the underlying learning algorithm and is applicable as long as the classifier produces probability estimates.

Andreas, Klaus and Beiβbarth (2013) proposed *hierarchical twoing (hi2),* a novel algorithm for classification of high-dimensional data into ordered categories. The *hi2* combines the power of well-understood binary classification with ordinal response prediction. It is a tree-like classification scheme that recursively partitions the data into two-class problems so that the information from the classes far away has a more direct impact on the local binary classification.

Many statistical approaches, such as machine learning algorithms for ordinal data classification (although rare), rely on specific distributional assumptions for modeling the class variable and they also assume a stochastic ordering of the input space (Ralf, Thore, and Obermayer 1999).

In chapter 2, we introduce the literature review which covers the theoretical frameworks of classification and regression trees, support vector machines, random forests, the lasso and multinomial logit regression modelling approaches. Chapter 3 is devoted to methodology, chapter 4 shows the results and discussion of the results in chapter 5. Chapter 6 presents the conclusions and recommendations for further research.

CHAPTER 2

LITERATURE REVIEW

2. 1. Theoretical Framework of Classification and Regression Trees (CART)

The classification and regression tree (CART) algorithm, introduced by Breiman et al. (1984) offers a nonparametric approach to classification and regression problems. CARTs are an approach to discovering relationships among a large number of independent (predictor) variables and a categorical or continuous trait. Classification trees are applied to categorical outcomes while regression trees apply to continuous outcomes. Both involve application of a recursive algorithm that aims to partition the individuals into groups in a way that minimizes the within-group heterogeneity (Foulkes 2009).

Consider a classification problem with k classes, p predictors, and n observations. The CART algorithm begins with a root node containing all observations. This node is split into two child nodes according to a splitting criterion determined based on one predictor such that the split minimizes an impurity measure.

The classification tree is aimed at predicting that each observation belongs to the class that occurs most in the training observations in the region to which it belongs. In interpretation of the classification tree, we are interested in class prediction corresponding to a particular terminal node region and the class proportions among the training observations that fall in that region. We use recursive binary splitting to grow a classification tree. The classification error rate is the proportion of the misclassified observations. It is the fraction of the training observations in the terminal node region that do not belong to the most common class: $E = 1 - \max_k \hat{p}_{mk}$; where E is the classification error rate, $\hat{p}_{mk}$ represents the proportion of the training observations in the m^{th} region that are from the k^{th} class, and $0 \leq \hat{p}_{mk} \leq 1$. However, it turns out that the

classification error is not sufficiently sensitive for tree growing as far as determining the node impurity measure is concerned, and so the two preferable sensitive measures are the Gini index and cross entropy (James et al. 2013). For classification problems, the most preferable impurity measure is the Gini impurity index, which is calculated as $G = \sum_{k=1}^{K} \hat{p}_{mk} (1 - \hat{p}_{mk})$, which measures a total variation across K classes for node *m*. We can see that the Gini index takes on a small value if all of the $\hat{p}_{mk}{}'s$ are close to zero or one. For this reason, the Gini index is referred to as a node purity – a small value indicates that a node contains predominantly observations from a single class (James et al. 2013). The nodes are recursively split by choosing the next split to maximize the reduction in impurity, defined as $\Delta i(\theta, m) = i(m) - [p_l i(l) + p_r i(r)]$, where θ is a split of node m into nodes l and r with a proportion p_l of observations in m going to l, and proportion p_r going to r (Ahn and Moon 2010). Splitting continues until a stopping criterion is reached. The default stopping criterion suggested by Breiman et al. (1984) is to terminate the algorithm when all nodes either contain a single class or have size $N(t) \leq 5$ for a node t, where $N(t)$ is the number of observations in t; the R implementation of CART in the package *rpart* uses a default stopping criterion of $N(t) = 20$. A cross-validation can be used to determine the optimal minimum node size. For prediction in classification problems, each terminal node (also known as leaf node) in the resulting tree is assigned an overall class equal to the majority class in the node. The predicted class of a new observation is the class of the terminal node that the new observation belongs to.

One potential problem of the CART algorithm is overfitting the training data, which occurs when trees are grown too deeply, resulting in decision boundaries that closely track the classes of the training sample. A common approach to avoiding this problem is the application of cost-complexity pruning. For a tree T, let $R(T)$ be an estimate of the tree misclassification

rate, and $|\tilde{T}|$ be the number of terminal nodes in the tree. For a predefined complexity parameter α (where α is a nonnegative tuning parameter), the cost-complexity measure $R_\alpha(T)$ is defined as

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|.$$

Starting with a deeply grown tree, cost-complexity pruning constructs a nested sequence of subtrees by successively deleting branches to minimize $R_\alpha(T)$ (Tibshirani, Friedman, and Hastie 2009). The resulting tree with minimum $R_\alpha(T)$ will produce coarser decision boundaries (due to fewer terminal nodes) and thus be less prone to overfitting. The choice of the value of α is determined by a cross-validation.

A related problem for CART is high-variance in the tree structure with respect to the changes in the sample data (Larocque, Ben-Ameur, and Bou-Hamad 2011). The decision boundaries produced by the CART algorithm are rectilinear boundaries parallel to the variable axes. Small variations in the data can produce changes in the tree structure by causing splits with different predictors or introducing additional splits. Changes in the tree structure are likely to result in large shifts in the decision boundaries, with the result that small changes in the training data can result in significantly different predictions. This high-variance is of particular concern when working with noisy data or data with many predictors that are not strongly correlated with the response variable, as is the case with genomic (or microarray) data.

For instance, consider $y = (y_1, y_2, \ldots, y_n)$ and p predictor variables given by $x_1, x_2, \ldots, x_p$ where $x_j = (x_{1j}, x_{2j}, \ldots, x_{nj})^T$ and $i = 1,2, \ldots, n$ indicates individual, in discovering the relationship between $X = [X_1, X_2, \ldots, X_p]$ and y. We construct a tree by determining a variable x_j that is the 'most predictive' of the trait y. The individuals in the sample are then divided into two groups based on their corresponding value of x_j (Foulkes 2009). Let the set of all individuals be denoted by Ω, and let the most predictive variable be given by

$x_{(1)}$. Individuals are first divided into Ω_1 and Ω_2 based on the value of $x_{(1)}$. Define $\Omega_1 = \{i: x_{i(1)} = 0\}$ and $\Omega_2 = \{i: x_{i(1)} = 1\}$ for $i = 1, \ldots, n$, representing the individual. Next, identify the 'most predictive' variable x_j of y_j within each of the groups of individuals given by Ω_1 and Ω_2. Suppose this variable is $x_{(2)}$ for Ω_1 and $x_{(3)}$for Ω_2. Further subgroups are then defined based on the values of $x_{i(2)}$ and $x_{i(3)}$ The partitioning procedure is then repeated recursively until a stopping rule is achieved.

2. 2. Theoretical Framework of Support Vector Machine

Support vector machines (SVM) are a group of supervised learning methods that can be applied to classification or regression (Ovidiu 2007). SVM models were originally defined for the classification of linearly separable classes of objects. It is easy to find a line (hyperplane) that separates two-dimensional objects that belong to two classes perfectly (i.e., class +1 and class -1).

SVM can also be used to separate classes that cannot be separated with a linear classifier. In such cases, the coordinates of the objects are mapped into a feature space using nonlinear functions called a kernel function Φ. The feature space is a high-dimensional space in which the two classes can be separated with a linear classifier (Ovidiu 2007). The nonlinear mapping induced by the feature functions is computed with special nonlinear functions (kernels). Support Vector Machine (SVM) is part of a kernel-based methods which are used for pattern classification and regression. A classifier takes an input pattern called feature vector, and determines to which class it belongs to. Kernels have the advantage of operating in the input space, where the solution of the classification problem is a weighted sum of kernel functions evaluated at the support vectors.

2.2.1 The Hyperplane and the Separating Hyperplane

In general, a hyperplane, in a p -dimensional space, is a flat *affine* subspace of dimension $p-1$. For instance, in two dimensions, a hyperplane is a flat one-dimensional subspace (a line) while it is a flat two-dimensional subspace (or a plane) in three dimensions. It can be hard to visualize a hyperplane in $p > 3$dimensions, but the notion of a $(p-1)$ - dimensional flat subspace still applies. Mathematically, the hyperplane, in two dimensions, is defined by the equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 \qquad (2.1)$$

for the parameters β_0, β_1 and β_2 . Saying that (2.1) above 'defines' the hyperplane means that any $X = (X_1, X_2)^T$ for which (2.1) holds is a point on the hyperplane. The extension of (2.1) to a p - dimensional setting gives a p - dimensional hyperplane, which is defined by the equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0 \qquad (2.2)$$

for the parameters β_0, β_1, $\beta_2, \dots, \beta_p$. If a point $X = (X_1, X_2, \dots, X_p)^T$ satisfies (2.2) above, then the point X lies on the hyperplane (James et al. 2013).

If *X* does not satisfy equation (2.2), but rather,

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0 \quad or \quad \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0 \qquad (2.3)\ \&\ (2.4)$$

Then this indicates that X lies on the either side of the hyperplane and so the hyperplane divides the p - dimensional space into two halves. Calculating the sign of one of the sides of the hyperplane helps to determine the side of the hyperplane a point lies.

Consider an $n \times p$ data matrix **X** that consists of n training observations in a

p-dimensional space, $x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$, (2.5)

such that these observations fall into two classes, that is, $y_1, \ldots, y_n \in \{-1, +1\}$ where -1 represents one class and +1 represents the other. We want to develop a classifier based on the training data that will classify the test observation $x^* = (x_1^* \ldots x_p^*)^T$ using its feature measurements. The separating hyperplane has the property that

$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0$ for $y_i = +1$ and

$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0$ for $y_i = -1$,

Also, $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0$ for i=1,…, n.

The test observation $x^*$is assigned based on the sign of $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$. A negative $f(x^*)$ is assigned to the class -1 whereas a positive one is assigned to the class +1. The farther x^* lies from the hyperplane the better because it is indicative of an $f(x^*)$ far from zero hence confident of the class assignment of x^*. Likewise, we're less confident about the class assignment of x^* if it is located near the hyperplane, that is, $f(x^*)$ close to zero (James et al. 2013).

2.2.2 The Maximal Margin Classifier

An infinite number of hyperplanes can be obtained by shifting (up, or down, or rotate) a given separating hyperplane a little bit without coming into contact with any of the observations. The maximal margin (or optimal separating) hyperplane is the separating hyperplane that is farthest from the training observations. If the perpendicular distance from the training observations to the separating hyperplane is calculated, the smallest (minimal) distance from the hyperplane to the observations is known as the *margin*. The hyperplane with the largest minimal distance from the observations to the training observations is the *maximal margin hyperplane*. Each of the test observations can then be classified based on which side of the maximal margin hyperplane it lies. This is known as the *maximal margin classifier*. In other words, the maximal

margin hyperplane represents the mid-line of the widest "slab" that we can insert between the two classes being classified. If a classifier with a large margin on the training data also has a large margin on the test data, the test observations will then be correctly classified. The drawback with the maximal margin is the fact that it can lead to overfitting. The maximal margin hyperplane directly depends on the support vectors (training observations that "support" the maximal margin hyperplane) but not on other observations. The support vectors lie on the boundary of the margin.

SVMs uses an implicit mapping Φ of data into a high dimensional feature space defined by a kernel feature, that is, a function returning the inner product $[\Phi(x), \Phi(x')]$ between the images of two data points $\boldsymbol{X}$ and $\boldsymbol{X}'$ in the feature space. The learning then takes place in the feature space, and the data points only appear inside the dot products with other points, the situation called a "kernel trick" (Scholkopf and Smola 2002). More precisely, if a projection $\Phi: X \rightarrow H$ is used, the dot product $[\Phi(x), \Phi(x')]$ can be represented by kernel function k, $k(x, x') = [\Phi(x), \Phi(x')]$, which is computationally simpler than explicitly projecting x and x' data points into the feature space *H* (Karatzoglou, Meyer, and Hornik 2006).

2.2.3 Multi-Class SVM Classification

SVM classification was originally defined for two classes problems. This is a limitation in some cases when three or more classes of patterns are present in the training set. Although the number of proposals for extending SVMs to k-class case has been made, many multiclass SVM classification approaches decompose the training set into several two-class problems (Ovidiu 2007; Gareth 2013).

The *one-versus-one (or all-pairs) approach* trains a two class SVM model for any two classes from the training set, which for a k-class problem results in $\frac{k(k-1)}{2}$ or $\binom{k}{2}$ SVM models.

In the prediction phase, a voting procedure assigns the class of the prediction pattern to be the class with the maximum number of votes, that is, the final classification is performed by assigning the test observation to the class to which it was most frequently assigned in the $\binom{k}{2}$ pairwise classification. Debnath, Takahide, and Takahashi (2004) proposed an optimized one-versus-one multiclass SVM in which only a minimum number of SVM classifiers are trained for each class.

The *one-versus-all procedure* requires a much smaller number of models, namely for a k-class problem, only k SVM classifiers are needed, each time comparing one of the k classes to $k-1$ classes. The i^{th} SVM classifier is trained with all patterns from the i^{th} class labeled +1, and all other patterns labeled -1.

If x^* denotes the test observation, the observation is assigned to the class for which $\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \cdots + \beta_{pk}x_p^*$ is largest, as this amounts to the high level of confidence that the test observation belongs to the k^{th} class rather than to any of the other classes (Gareth 2013). Although it is easier to implement than the one-versus-one approach, the training sets may be imbalanced due to the large number of -1 patterns. For this thesis, the one-verses-all approach will be used.

Hsu and Lin (2002) did a comparative evaluation of one-versus-one and one-versus-all methods for 10 classification problems. They found out that one-versus-all is less suitable. However, not all literature reports agree with this finding. Based on a critical review of the existing literature on multiclass SVM and experiments with many datasets, Rifkin and Klautau (2004) concluded that the one-versus-all SVM classification is as accurate as any other multiclass approach. Angulo, Parra, and Catala (2003) proposed the K-SVCR (K-class support vector classification-regression) for k-class classification. This algorithm has ternary outputs,

{-1, 0, +1}, and in the learning phase evaluates all patterns in a one-versus-one-versus-rest procedure by using a mixed classification and regression SVM. The prediction phase implements a voting scheme that makes the algorithm fault-tolerant. According to Statnikov et al. (2005) and Tao, Zhang, and Ogihara (2004), multiclass SVM classification is particularly relevant for the classification of microarray gene expression data, with particular importance for disease recognition and classification.

2. 3. Theoretical Framework of Random Forest

Random forests (RFs) represents an extension of CART that involves generating an ensemble of classification or regression trees. This set of trees is constructed in a manner that addresses some of the limitations of CART. Through repeated sampling of sets of predictor variables at the tree splitting stage, RFs offer a natural approach to handling collinearity (Foulkes 2009). Tibshirani, Friedman, and Hastie (2009) and Larocque, Ben-Ameur, and Bou-Hamad (2011) showed that ensemble methods such as random forests, bootstrap-aggregating and boosting have been shown to effectively reduce the variance of CART and improve the overall predictive accuracy.

The bagging approach (or bootstrap-aggregating) produces an ensemble of trees, each grown from a bootstrapped sample of the original training data. From this ensemble, the predicted class of a new observation is determined by majority vote of the predictions of the trees in the ensemble. Random forests are a modification of the bagging method. Random forests improve the performance through de-correlating the trees in the ensemble (Breiman 2001).

Unlike in the CART setting, RF do not yield a clear structure for the final model association, that is, RF does not generate a final tree that can be interpreted as a model for the association. Instead, a measure of variable importance resulting from the application of the

approach provides us with a general measure of contribution of each potential predictor variable to the observed variability in the response variable under investigation (Foulkes 2009).

RF fits many classification trees to a data set, and then combines the predictions from all the trees. The algorithm begins with the selection of many (e.g., 1000 trees, the default in *R* is 500 trees) bootstrap samples from the data. In a typical bootstrap sample, approximately 63% of the original observations occur at least once. Observations in the original data set that do not occur in a bootstrap sample are called the out of-bag observations. A classification tree is then fit to each bootstrap sample, but at each node, only a small number of randomly selected variables (e.g., the square root of the number of variables as a default for classification) are available for the binary partitioning.

Each fitted tree is used to predict the out-of-bag observations. The predicted class of an observation is calculated by a majority vote of the out-of-bag predictions for that observation, with ties split randomly. The accuracies and error rates are then computed for each observation using the out-of-bag predictions, and then averaged over all observations. Because the out-of-bag observations were not used in the fitting of the trees, the out-of-bag estimates are essentially similar to the cross-validated accuracy estimates. Probabilities of membership in the different classes are estimated by the proportions of out of-bag predictions in each class.

2.3.1. Variable Importance

Most statistical procedures for regression and classification measure variable importance indirectly by selecting variables using some criteria such as statistical significance and Akaike's Information Criterion. The approach taken in RF is completely different. For each tree in the forest, there is a misclassification rate for the out-of-bag observations. To assess the importance of a specific predictor variable, the values of the variable are randomly permuted for the out-of-

bag observations, and then the modified out-of-bag data are passed down the tree to get new predictions. The difference between the misclassification rate for the modified and original out-of-bag data, divided by the standard error, is a measure of the importance of the variable (Cutler et al. 2007).

Let $X = (x_1, \dots, x_p)$ be the set of p potential predictor variables, where $x_p = (x_{1p}, \dots, x_{np})^T$, where n is the number of individuals in our sample. Let y be the response variable. Then the measure of node impurity (most common measure), given by the mean square error (MSE) for a node is; $I(\Omega) = \frac{1}{n_\Omega} \sum_{i \in \Omega} (y_i - \bar{y})^2$, where n_Ω is the number of individuals in the node Ω, and $\bar{y}$ is the mean of the corresponding elements of y. The best split is the one that maximizes the reduction in the node impurity given by, $\emptyset = I(\Omega) - I(\Omega_L) - I(\Omega_R)$, where $I(\Omega_L)$= left child node of Ω and $I(\Omega_R)$= right child node of Ω (Foulkes 2009).

Unlike the bagging method which grows each tree by considering all p predictors from the data set with ensembles that are highly correlated, the random forest method considers m predictors, $m < p$, from the total set of predictors. By randomly selecting a subset of predictors, the correlation of the trees in an ensemble is reduced, leading to a greater reduction in variance for the random forest model compared to simple bagging.

Breiman (2001) proved that random forests do not overfit the data, even for a very large number of trees, an advantage over CART. Random forest decision boundaries tend to be axis-oriented due to the nature of the tree decision boundaries, but the ensemble voting allows for much more dynamic boundaries than sharp rectilinear edges.

The main parameters for the random forests method are the number of trees to grow, $ntree$, and the number of predictors to try per tree, $mtry$. These values can be chosen to minimize the estimated classification error using a cross-validation. The R package used to

implement random forests method is *random Forest*; for classification problems the default parameters are $ntree = 500,$ and $mtry = \sqrt{\text{p}}.$ These default values have been shown to have consistently good results across a variety of classification problems (Ahn and Moon 2010).

2. 4. Theoretical Framework of Lasso

The lasso is a regularization technique for estimating or fitting a generalized linear model via penalized maximum likelihood, that is, the lasso includes a penalty that constrains the size of the estimated coefficients. As the penalty term increases, the lasso technique tends to shrink some coefficients and set others to zero, thus making the lasso an alternative to stepwise regression and other model selection and dimensionality reduction techniques. Therefore, the lasso can deal with all shapes of data, including very large sparse data matrices. Although the lasso is said to generate the estimates that are biased to be small (shrinkage estimator), these estimates can have smaller error than the ordinary maximum likelihood estimator when applied to new data (Tibshirani 1996).

Consider the data $(X^i, y_i), i = 1,2, \dots, N$, where, $X^i = (x_{i1}, \dots, x_{ip})^T$ are predictor variables and y_i are the responses, and letting $\hat{\beta} = (\hat{\beta}_1, \dots, \hat{\beta}_p)^T$, then the lasso estimate $(\hat{\alpha}, \hat{\beta})$ is defined by: $(\hat{\alpha}, \hat{\beta}) = \arg\min\{\sum_{i=1}^{N}(y_i - \alpha - \sum_j \beta_j x_{ij})^2\}$ subject to $\sum_j |\beta_j| \leq t$, $t \geq 0$; where t is the tuning parameter and $\hat{\alpha} = \bar{y}$, for all t. And it can be assumed without loss of generality that $\bar{y} = 0$. Therefore, α can as well be omitted. $\hat{t} = \sum(|\hat{\beta}_j^0| - \hat{\gamma})$; $\hat{\beta}_j^0$ is the least squares estimates and γ is determined by the condition $t = \sum |\hat{\beta}_j|$.

$\hat{\gamma} = argmin_{\gamma \geq 0}\{\hat{\tau}^2[p - 2\#\left(j; \left|\frac{\hat{\beta}_j^0}{\hat{\tau}}\right| < \gamma\right) + \sum_{j=1}^{p} \max\left(\left|\frac{\hat{\beta}_j^0}{\hat{\tau}}\right|, \gamma\right)^2]\}$; where p is the number of variables and $\hat{\tau}$ is the standard error of $\hat{\beta}_j^0$ given as $\hat{\tau} = \hat{\sigma}/\sqrt{N}$; $\hat{\sigma}^2 = \frac{\sum(y_i - \hat{y}_i)^2}{N-p}$ (Tibshirani 1996). The selection of the best subset of size k reduces to choosing the largest k coefficients in absolute value and setting the rest to zero. Equivalently, $\hat{\beta}_j = \hat{\beta}_j^0$ if $|\hat{\beta}_j^0| > \lambda$ for some choice of λ and

$\hat{\beta}_j = 0$ otherwise. Assumptions:

- the observations are independent or $y_i's$ are conditionally independent given $x_{ij}'s$, for example, if we consider the model ($x_{ij}\, y_1$, $x_{ij}\, y_2$), which means that y_1 and y_2 are conditionally independent given x_{ij} . Mathematically, the model ($x_{ij}\, y_1$, $x_{ij}\, y_2$) means that the conditional probability of y_1 and y_2 given x_{ij} equals the product of conditional probabilities of y_1 given $x_{ij,}$ $\{P(y_1|x_{ij})\}$ and y_2 given x_{ij} , $\{P(y_2|x_{ij})\}$.
- x_{ij} are standardized so that $\sum_i x_{ij}^2/_N = 1$, $\sum_i x_{ij}/_N = 0$.

The parameter $t \geq 0$ controls the amount of shrinkage that is applied to the estimates. If $\hat{\beta}_j^0$ is the full least square estimates, then values of $t < \sum |\hat{\beta}_j^0|$ will cause the shrinkage of solutions towards zero, and some coefficients may be exactly equal to zero (Tibshirani 1996).

Alternatively, the lasso solution is to minimizing the optimization problem

$$\sum_{i=1}^{n}(Y_i - \beta^T X_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|;$$

where $\beta = (\beta_1, \ldots, \beta_p)$ and $\lambda \geq 0$ is a penalty term. Thus, the constraint that is utilized is an L_1 constraint. Another way is minimizing $\sum_{i=1}^{n}(Y_i - \beta^T X_i)^2$, subject to the constraint that $\sum_{j=1}^{p} |\beta_j| \leq t$. Note that in the absence of the constraint, the solution is given by the ordinary least squares (OLS) estimator. If the usual OLS estimator satisfies the constraint, then the LASSO and OLS estimates of β coincide. However, for smaller values of t, some of the components of β are estimated to be zero. While Tibshirani considered estimating coefficients in regression models using LASSO, our interest is in using gene expression data to classify tumors (Ghosh and Chinnaiyan 2005).

2. 5. Theoretical Framework of Logistic Regression

Consider a population with two groups. We can think of logistic regression approach to classification by focusing on the probability that an observation belongs to each of the two groups,

given a particular characteristic vector (Green et al. 1998). Green et al. (1998) used the Bayes Theorem to express the probabilities of group 1 and group 2 as:

$P(\pi_1|x) = \frac{P(x|\pi_1)}{P(x|\pi_1)+P(x|\pi_2)}$ and $P(\pi_2|x) = \frac{P(x|\pi_2)}{P(x|\pi_1)+P(x|\pi_2)}$.

Bayes' Theorem: Assume that $B_1, \dots, B_k$, is a partition of the sample space, S, such that $P(B_i > 0), \text{for } i = 1,2, \dots, k$. Then $P\left(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_{i=1}^{k} P(A|B_i)P(B_i)}\right)$ (Wackerly, Mendenhall, and Scheaffer 2008). Assuming that the set of independent variables is distributed multivariate normally with a common variance-covariance matrix, Green et al. (1998) found that the odds ratio (ratio of the two probabilities) is equivalent to;

$\frac{P(\pi_1|x)}{P(\pi_2|x)} = \frac{P(x|\pi_1)}{P(x|\pi_2)} = \exp(\alpha + \beta' x)$; where $\alpha = -\left(\frac{1}{2}\right)(\mu_1 - \mu_2)'\Sigma^{-1}(\mu_1 - \mu_2)$,

$\beta = \Sigma^{-1}(\mu_1 - \mu_2)$, $\mu_1$is the mean of group 1 (π_1), μ_2 is the mean of group 2 (π_2) and Σ is the common variance-covariance matrix. The odds ratio thus gives rise to: $P(\pi_1|x) = \frac{\exp(\alpha+\beta' x)}{1+\exp(\alpha+\beta' x)}$ and $P(\pi_2|x) = \frac{1}{1+\exp(\alpha+\beta' x)}$. If the population parameter values, α and β, are known, we can calculate the probabilities that a new observation vector belongs to either group 1 or group 2 respectively using the above expressions. However, in reality, the population parameters α and β are in most cases unknown. Therefore, the population parameters α and β are replaced with the estimates from the training sample in order to calculate the probabilities. The new observation vector is classified as belonging to group 1 if the probability of it belonging to group 1 is greater than a particular value determined by the analyst and to group 2 otherwise.

2. 5. 1. The Multinomial Logit Regression (MLR) Model

The multinomial logit model entails generalization of the logistic regression approach to three or more groups. Its derivation is similar to that of logistic regression. The colon cancer data set (GSE17536), downloaded from the Gene Expression Omnibus (GEO) database (Smith et al. 2010), has four groups or stages, that is, I, II, III and IV, and thus multinomial regression is appropriate. Extension of the logistic regression classification approach described above gives rise to the following expressions that can be used to calculate the probabilities of an observation belonging to each of the four groups.

$$P(\pi_1|x) = \frac{\exp(\alpha_1+\beta_1'x)}{1+\exp(\alpha_1+\beta_1'x)+\exp(\alpha_2+\beta_2'x)+\exp(\alpha_3+\beta_3'x)}, \qquad (2.5.2.1)$$

$$P(\pi_2|x) = \frac{\exp(\alpha_2+\beta_2'x)}{1+\exp(\alpha_1+\beta_1'x)+\exp(\alpha_2+\beta_2'x)+\exp(\alpha_3+\beta_3'x)}, \qquad (2.5.2.2)$$

$$P(\pi_3|x) = \frac{\exp(\alpha_3+\beta_3'x)}{1+\exp(\alpha_1+\beta_1'x)+\exp(\alpha_2+\beta_2'x)+\exp(\alpha_3+\beta_3'x)}, \text{ and} \qquad (2.5.2.3)$$

$$P(\pi_4|x) = \frac{1}{1+\exp(\alpha_1+\beta_1'x)+\exp(\alpha_2+\beta_2'x)+\exp(\alpha_3+\beta_3'x)}. \qquad (2.5.2.4)$$

If the population parameters α_i, β_i , where $i = 1,2,3,4$ values are known, we can calculate the probabilities that the new observation vector belongs to cancer stages I, II, III, or IV. Of course, in reality, these population parameters are unknown and therefore must be replaced by the training sample estimates.

The parameters can be estimated by (conditional) maximum likelihood logistic estimation procedure using an iterative algorithm. The Newton-Raphson method is used to estimate the parameters for which the observed data would have the highest probability of occurrence (Green et al. 1998 : Czepiel 2002). The expression (2.5.2.1) calculates the probability that a new observation vector belongs to stage I, expression (2.5.2.2) calculates the probability that a new observation vector belongs to cancer stage II and so on.

According to Green et al. (1998), we classify the new observation vector to the group or stage with the highest calculated probability.

CHAPTER 3

METHODS

3.1. Multi-Class Prediction Performance Metrics

Although there are many performance metrics to evaluate the efficiency of a multi-class classifier, there are limited performance metrics suitable for ordinal data. The commonly used ordinal data performance metrics include assessing the overall accuracy, the rank association and the misclassification error that accounts for the inherent ordering between classes. Each of the metric mentioned above attempts to measure how well the predicted labels for the N samples $\{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_N\}$ correspond to the true labels $\{y_1, y_2, \dots, y_N\}$(Chang, George, and Tzu-Pin 2016).

3.1.1 Accuracy

Accuracy is the most common statistic for reporting the performance of a multiclass classifier. Accuracy can be defined as the number of correct predictions across all the classes, say k, divided by the number of the samples, n. The accuracy measures the proportion of the correct classification. Accuracy is given by Chang, George, and Tzu-Pin (2016) as:

$Acc = \frac{1}{N} \sum I_{\{y_i - \hat{y}_i\}}$, where $0 \leq \text{Acc} \leq 1$.

While accuracy is simple in nature, assessing performance using accuracy alone has long been known to be prone to erroneous interpretation. This is because accuracy does not account for the degree of class imbalance that may be present in a given dataset, which means it can only be correctly interpreted in relation to a dataset-dependent baseline accuracy (Carillo, Brodersen, and Castellanos 2014). Accuracy alone as a performance metric also ignores the cost of misclassification.

3.1.2 Rank Order Correlation

The measures of ordinal association that consider whether the variable Y tends to increase as X increases includes the Gamma, Kendall's tau-b, Stuart's tau-c, and Somers' D (Signorell 2015). These measures are appropriate for ordinal variables, and they classify pairs of observations as concordant or discordant. A pair is concordant if the observation with the larger value of X also has the larger value of Y, that is, a pair of observation is in the same direction. A pair is discordant if the observation with the larger value of X has the smaller value of Y (Agresti 2002). Consider two sets of paired observations, (X_i, Y_i) and (X_j, Y_j). A pair is concordant if $X_i > X_j$ and $Y_i > Y_j$ or $X_i < X_j$ and $Y_i < Y_j$. On the other hand, a pair of observations is said to be discordant when $X_i > X_j$ and $Y_i < Y_j$ or $X_i < X_j$ and $Y_i > Y_j$.

3.1.2.1. Kendall's Tau-b. The Kendall's rank correlation can be used to measure the association between the true class,y, and the predicted class, $\hat{y}$. Kendall's coefficient is the most widely used rank order correlation statistic, $\tau_b \, \epsilon \, [-1,1]$ (Kendall 1962). Kendall's Tau-b is preferred over Kendall's Tau-a and Kendall's Tau-c because it incorporates ties in its statistic. Kendall's tau is given as:

$$\tau_b = \frac{C-D}{\sqrt{(C+D-T_t)(C+D-T_p)}},$$

Where C=the number of concordant pairs, D= the number of discordant pairs, T_t = the number of tied pairs in the true class membership, and T_p = the number of tied pairs in the predicted class membership. Kendall's tau assesses the order relation between the true and predicted class labels and thus is not affected by values chosen to represent class labels, which are often arbitrary. A positive value indicates that both the predicted and reference (or true) classes increase together. A negative value indicates that as the predicted classes increase, the reference decreases

and vice versa. The sign, positive or negative, is important because it portrays the relationship between the predicted and the reference classes.

3.1.3. The Cost Matrix

Given ordered classes, we are not only interested in maximizing the classification accuracy, but also in minimizing the distances between the actual and the predicted classes. Chang, George, and Tzu-Pin (2016) specifically devised ordinal data classification algorithm that incorporates the cost of misclassification error. Jayasree and Gavya (2014) defined an imbalanced dataset as one whose classification categories are not approximately equally represented.

Let $C_{M\ X\ M}$ denote the cost matrix associated with a single experiment consisting of predicting class membership for *N* test objects into one of the *M* classes. The rows and columns of the matrix, $C_{M\ X\ M}$, are the predicted and the true (or reference) class memberships respectively. The entry $c_{i,j}$ denotes the cost of misclassifying a class j object into class i. The misclassification cost is determined by the product of two factors – first, the inverse probability of misclassifying an object into class i given that the object has been misclassified and second, the absolute prediction error. All the main diagonal entries of the cost matrix are zero, that is, $c_{i,i} = 0$, because no cost is associated with correctly classifying a test sample (Chang, George, and Tzu-Pin 2016). Chang, George, and Tzu-Pin (2016) proposed the following formula that computes the off-diagonal entries of the cost matrix.

$$c_{i,j} = \frac{\sum_{i \neq j}^{M} n_i}{n_i} \ |j - i|, \text{for } i \neq j, i = 1, \dots, M, \text{and } j = 1, \dots, M,$$

where $n_i$is the number of training samples in class i.

The first component of the misclassification cost formula above ensures that the cost is weighted by the size of the class. This is important to ensure that the prediction error for classes

that are rare is not masked by the prediction error for the majority class, thus a good algorithm to handle class imbalances in the dataset too. The second component is a linear absolute loss function value to measure the distance between the predicted and true class labels. The higher the $c_{i,j}$ value of the cost matrix, the higher the cost of misclassifying a sample in class j into class i.

3.2. Total Misclassification Cost (TC) of a Classifier

The confusion matrix is a table containing the predicted and actual classifications obtained using a classification system. The rows of the confusion matrix represent the predicted and the columns represents the true (actual) class membership. Let $F_{M\times M}$ represent the confusion matrix of a given single classifier for an *M*-class classification problem. The entry $f_{i,j}$ denotes the total number of class *j* objects (or samples) incorrectly classified into class i. The entry $f_{i,i}$, that is, *i=j,* represents the number of class *i* samples correctly classified into class *i* (Chang, George, and Tzu-Pin 2016).

$$TC = \sum_{i=1}^{M}\sum_{j=1}^{M} c_{i,j} * f_{i,j} \; = trace\,(CF^T)$$

To estimate the maximum total misclassification cost of a classifier, optimization was used. Maximum TC was obtained by fixing the cost matrix *C* and letting the confusion matrix to be a random variable F_x. The maximum TC was obtained as follows;

$$maxTC = \sum_{i=1}^{M}\sum_{j=1}^{M} c_{i,j} * fx_{i,j}$$

The constrOptim function in R was used to determine the unique solution of the maximum total misclassification cost *(http://www.inside-r.org/r-doc/stats/constrOptim*).

3.3. Leave One Out Cross-Validation (LOOCV)

The LOOCV is a simple cross-validation technique where each learning set is created by taking all the samples except one, the test set being the sample left out. Thus, for *n* samples, we have *n* different training sets and *n* different test sets. This cross-validation procedure does not waste much data as only one sample is removed from the training set to make the test set. There are some caveats to be weighed by potential users of LOOCV for model selection. When compared with k-fold cross validation, one builds n models from n samples instead of k models, where $n > k$. Moreover, each is trained on $n - 1$ samples rather than $\frac{(k-1)n}{k}$. In both ways, assuming k is not too large and $k < n$, LOOCV is more computationally expensive than k-fold cross validation. Although LOOCV results in less bias, it often results in high variance as an estimator for the test error.

Since each model is build using $n - 1$ of the n samples, models constructed from folds ($n - 1$ samples) are virtually identical to each other and to the model built from the entire training set (scikit-learn developers 2010-2014).

3.4. Patients and Data

The colon cancer data set (GSE17536) downloaded from the Gene Expression Omnibus (GEO) database (Smith et al. 2010) comprises of 4 cancer stages as categorized by the American Joint Committee on Cancer (AJCC). The 177 samples (patients) from the Moffitt Cancer Center (MCC), with the AJCC stages I-1V resulted in 24, 57, 57, and 39 patients respectively. The minimum, median, mean and maximum age of the patients was 26 years, 66 years, 65.48 years and 92 years respectively. 45.76% of the patients were women and 54.24% were men. Among the 177 colon cancer patients, 9 were Black, 151 were Caucasian, 1 was Hispanic and 16 were

others (not Caucasian, Black, or Hispanic). Below are summary tables for the patients' ethnicity and gender by the AJCC stages.

TABLE 1. The Cancer Stages and Patients by Ethnicity

Ethinicity	Colon Cancer Stage				Total
	1	2	3	4	
Black	0	3	3	3	9
Caucasian	20	48	49	34	151
Hispanic	0	1	0	0	1
Other (Not Black, Caucasian, or Hispanic)	4	5	5	2	16
Total	24	57	57	39	177

TABLE 2. The Cancer Grades and Patients by Gender

Grade	Gender		Total
	Female	Male	
Well differentiated (WD)	8	8	16
Moderately differentiated (MD)	59	75	134
Poorly differentiated (PD)	14	13	27
Total	81	96	177

3.5. Data Preprocessing and Screening

The raw data for this study was obtained from the Gene Expression Omnibus (National Center for Biotechnology Information) database. The RNA molecule was extracted using the Qiagen RNeasy kit. The GeneChip Scanner was used for scanning and

generating the raw CEL files. The bioconductor's "affy" package in R was used for Robust Microchip Analysis (RMA) normalization and raw data processing using default settings for background correction and normalization. All analyses were performed using R software. The final data values used for analysis are the RMA normalized signal intensity.

3.6. Methodology

The colon cancer data set consisted of 54,675 genes for each of the 177 samples (patients). The ANOVA F-test was used to determine the differentially expressed genes (variables) and reduce the variables to 1670 genes (p-value <= 0.01) on which the random forests variable importance was used to determine the 99 top-ranked genes, by the score of the mean decrease accuracy, for further analysis. The classification models were built using a reduced dataset consisting of the 99 top-ranked genes.

Each prediction model was trained and tested (1) on the same Moffitt Cancer Center (MCC) samples (2) on the MCC samples by the LOOCV technique within the Support Vector Machine (SVM), decision trees (rpart), least absolute shrinkage and selection operator (lasso), Multinomial Logistic Regression (MLR) and Random Forests (RF). Then the accuracy, Kendall's tau-b and the cost matrix were determined. We then present the criteria for selecting informative variables (genes), classification methods, and the performance metric, distance, d, proposed by Chang, George, and Tzu-Pin (2016) for evaluating multi-class classifiers for any given ordinal dataset. The metric, distance, which incorporates the misclassification cost and the class weight, guided by a tradeoff between the accuracy and the misclassification cost is then compared on the two approaches, LOOCV and training and testing on the same data set.

The colon cancer dataset is used to present a comparative analysis of the variable selection criteria, selected classification prediction models, distance (the metric proposed by Chang, George, and Tzu-Pin (2016)) against the accuracy, and the Kendall's tau-b.

CHAPTER 4

PRESENTATION OF THE RESULTS

4.1. Introduction

The univariate analysis of variance (ANOVA) F-test was used to determine the differentially expressed genes. With *p-value* = 0.01, 1670 out of 54,675 genes were obtained by ANOVA F-test, from which top 99 genes were determined by considering the scores of Mean Decrease in Accuracy by LOOCV, which was a method measuring variable importance in the Random Forests. The ANOVA F-test makes the following assumptions:

- the samples are independent, and
- the gene expression is normally distributed with same variance (Dziuda 2010). Assuming equality of variance ensures that the estimates we get will be optimal.

All the final models were built using a reduced dataset consisting of 99 out of the 1670 top-ranked genes. Two approaches of the classification were used to obtain the prediction models. One was trained and validated on the same Moffitt Cancer Center (MCC) samples. While the second employed leave one out cross validation (LOOCV) approach.

The confusion matrix is a table containing the predicted and actual classifications obtained using a classification system. The true positives (TP) for a given stage is the number of the correctly classified patients. The false positives (FP) for a given stage is the sum of values in the corresponding row (excluding the TP). The false negatives (FN) for a stage is the sum of the values in the corresponding column (excluding the TP). The total number of true negatives (TN) for a certain stage is the sum of all rows and columns excluding that stage's row and column. The components of the confusion matrix and how they are obtained are as follows;

- Sensitivity = TP / (TP + FN)

- Specificity = TN / (TN + FP)
- Prevalence = (TP + FN) / (TP+FP+TN+FN)
- PPV = (sensitivity * Prevalence)/((sensitivity*Prevalence) + ((1-specificity) * (1-Prevalence)))
- NPV = (specificity * (1-Prevalence))/(((1-sensitivity) * Prevalence) + ((specificity)*(1-Prevalence)))
- Detection Rate = TP / (TP+FP+TN+FN)
- Detection Prevalence = (TP + FP) / (TP+FP+TN+FN)
- Balanced Accuracy = (Sensitivity + Specificity) / 2

Note: PPV = Positive Predictive Value and NPV= Negative Predictive Value

4.2. The Cost Matrix

The rows and columns of the cost matrix are the predicted and the true (or actual) class memberships respectively. The cost matrix for the colon cancer data used is shown below. The entry $c_{i,j}$ denotes the cost of misclassifying a class j object into class i. For example, the cost of misclassifying a stage IV patient into stage I is 17.25 while the cost of misclassifying a stage I patient into stage II is 2.68. No cost is associated with correctly classifying a test sample, that explains why the entries in the main diagonal are equal to zero. Below is the cost matrix for the colon cancer data set (with 24, 57, 57, and 39 patients in stages I – IV respectively).

$$\begin{pmatrix} 00.00 & 05.00 & 10.00 & 17.25 \\ 02.68 & 00.00 & 02.11 & 04.84 \\ 05.37 & 02.11 & 00.00 & 02.42 \\ 11.78 & 06.15 & 03.08 & 00.00 \end{pmatrix}$$

The higher the $c_{i,j}$ value of the cost matrix, the higher the cost of misclassifying a sample in class j into class i.

4.3. Classification and Regression Trees (CART) Methodology

The classification trees were built using the 'rpart' package version 4.1-4 in R (Therneau, Atkinson, and Ripley 2015). See Table 3 (iii) below for the confusion matrix that was obtained by training and testing (or validating) on the same data set. The 'rpart' ordinal classification algorithm was designed for high-dimensional data, that has large number of predictor variables (p) compared to the smaller sample size (n).

TABLE 3. Confusion Matrices of Training and Testing the Models on the same Data Sets

(i) SVM: $\begin{vmatrix} 24 & 0 & 0 & 0 \\ 0 & 57 & 0 & 0 \\ 0 & 0 & 57 & 0 \\ 0 & 0 & 0 & 39 \end{vmatrix}$ (ii) RF: $\begin{vmatrix} 24 & 0 & 0 & 0 \\ 0 & 57 & 0 & 0 \\ 0 & 0 & 57 & 0 \\ 0 & 0 & 0 & 39 \end{vmatrix}$ (iii) CART: $\begin{vmatrix} 16 & 0 & 0 & 0 \\ 4 & 53 & 4 & 2 \\ 1 & 2 & 52 & 4 \\ 3 & 2 & 1 & 33 \end{vmatrix}$

(iv) LASSO: $\begin{vmatrix} 24 & 0 & 0 & 0 \\ 0 & 57 & 0 & 0 \\ 0 & 0 & 57 & 0 \\ 0 & 0 & 0 & 39 \end{vmatrix}$ (v) MLR: $\begin{vmatrix} 24 & 0 & 0 & 0 \\ 0 & 57 & 0 & 0 \\ 0 & 0 & 57 & 0 \\ 0 & 0 & 0 & 39 \end{vmatrix}$

NOTE: The rows for the confusion matrices represent the predicted samples while the columns represent the reference (or true) samples.

The CART classification algorithm had an estimated accuracy of 87.01% with a 95% confidence interval of (81.14%, 91.58%). Of the 177 patients correctly classified by the model, 16 were classified as stage I, 53 were classified as stage II, 52 were classified as stage III, and 33 were classified as stage IV. The number of the samples or patients misclassified by the CART training model were 23. As a consequence, the model has relatively high positive predictive values (PPV) for each stage at 100%, 84.13%, 88.14% and 84.62% for the colon cancer stages I, II, III, and IV respectively.

Table 4 below indicates high values for sensitivity, specificity, positive predicted value, negative predicted value and the balanced accuracy while lower values for prevalence, detection rate and detection prevalence.

TABLE 4. Stage Statistics for the CART by the Training Approach

Statistic	Class1	Class2	Class3	Class4
Sensitivity	0.6667	0.9298	0.9123	0.8462
Specificity	1.0000	0.9167	0.9417	0.9565
Positive Predictive Value	1.0000	0.8413	0.8814	0.8462
Negative Predictive Value	0.9503	0.9649	0.9576	0.9565
Prevalence	0.1356	0.3220	0.3220	0.2203
Detection Rate	0.0904	0.2994	0.2938	0.1864
Detection Prevalence	0.0904	0.3559	0.3333	0.2203
Balanced Accuracy	0.8333	0.9232	0.9270	0.9013

The LOOCV approach within CART resulted in the confusion matrix in Table 5 (iii). The estimated performance of CART using the LOOCV classification algorithm approach had accuracy of 47.46% with a 95% confidence interval of (39.92%, 55.09%).

TABLE 5. Confusion Matrices of LOOCV Approach for each of the 5 Models

(i) SVM: $\begin{vmatrix} 6 & 0 & 1 & 0 \\ 10 & 43 & 10 & 11 \\ 8 & 9 & 39 & 8 \\ 0 & 5 & 7 & 20 \end{vmatrix}$ (ii) RF: $\begin{vmatrix} 7 & 0 & 0 & 0 \\ 8 & 41 & 8 & 11 \\ 8 & 0 & 43 & 8 \\ 1 & 6 & 6 & 20 \end{vmatrix}$ (iii) CART: $\begin{vmatrix} 16 & 5 & 5 & 2 \\ 5 & 29 & 12 & 9 \\ 3 & 12 & 26 & 15 \\ 0 & 11 & 14 & 13 \end{vmatrix}$

(iv) LASSO: $\begin{vmatrix} 14 & 3 & 5 & 1 \\ 3 & 40 & 8 & 5 \\ 6 & 10 & 38 & 10 \\ 1 & 4 & 6 & 23 \end{vmatrix}$ (v)MLR: $\begin{vmatrix} 10 & 6 & 7 & 1 \\ 5 & 28 & 11 & 7 \\ 7 & 14 & 30 & 11 \\ 2 & 9 & 9 & 20 \end{vmatrix}$

NOTE: The rows for the confusion matrices represent the predicted samples while the columns represent the reference (or true) samples.

Of the 177 patients classified by the model, 13 were correctly classified as stage I, 29 as stage II, 26 as stage III, and 13 as stage IV. A total of 93 patients produced misclassified results. It can be noticed that the estimated accuracy obtained by the LOOCV approach is much lower compared to that obtained by directly training and testing on the same data set.

TABLE 6. Stage Statistics for the CART by LOOCV Approach

Statistic	Class1	Class2	Class3	Class4
Sensitivity	0.6667	0.5088	0.4561	0.33333
Specificity	0.9216	0.7833	0.7500	0.81884
Positive Predictive Value	0.5714	0.5273	0.4643	0.34211
Negative Predictive Value	0.9463	0.7705	0.7438	0.81295
Prevalence	0.1356	0.3220	0.3220	0.22034
Detection Rate	0.0904	0.1638	0.1469	0.07345
Detection Prevalence	0.1582	0.3107	0.3164	0.21469
Balanced Accuracy	0.7941	0.6461	0.6031	0.57609

The sensitivity, detection rates and detection prevalence for all the four classes using LOOCV approach were very low (see Table 6 above). Sensitivity tells us how often the test will be positive (true positive rate) if the patient indeed had the disease while specificity tells how often the test will be negative (true negative rate) if a person does not have the disease. The positive predictive values present the probability that the patient actually has the disease if the test results are positive while the negative predictive values indicates absence of the disease following negative test results.

4.3. Support Vector Machine (SVM) Methodology

The *one-verses-all* multi-class SVM was used. The radial kernel, which is a non-linear kernel was used. Four SVM classifiers were developed for each cancer stage. To build the classifier, each one of the four classes was compared to the other three classes in building the classifiers. The test sample is then assigned to the class whose classifier gives the highest value as this indicates with confidence that the patient belongs to that stage. SVM includes a penalty

parameter that allows a certain degree of misclassification, which is particularly important for non-separable training datasets. The tuning parameters (penalty parameter), cost = 2 and gamma = 0.015625, were determined from the data by LOOCV. The penalty parameter controls the trade-off between allowing training errors and forcing rigid separation margins. Increasing the value of the penalty parameter increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well, particularly when applied to the validation sets. The support vectors (the data points that lie closest to the decision surface or hyperplane) are usually difficult to classify. They carry all the relevant information about the classification problem. The support vectors have a direct bearing on the optimum location of the decision surface. The optimal hyperplane, in terms of classification performance is the one with the maximal margin of separation between the two classes, for example, {-1,1}.

Training and testing the SVM model on the same dataset resulted in the confusion matrix in Table 3 (i). The estimated re-substitution accuracy was 100% with a 95% confidence interval of (97.94%, 100%). All the patients were correctly classified. No patient was misclassified. The model had sensitivity, specificity, positive predictive values (PPV) and negative predictive values (NPV) for each colorectal cancer stage at 100% as depicted in Table 7.

TABLE 7. Colon Cancer Stage Statistics for SVM, Random Forest, Lasso and Logistic Regression Trained and Tested on the same Data Set

Statistic	Class1	Class2	Class3	Class4
Sensitivity	1.0000	1.000	1.000	1.0000
Specificity	1.0000	1.000	1.000	1.0000
Positive Predictive Value	1.0000	1.000	1.000	1.0000
Negative Predictive Value	1.0000	1.000	1.000	1.0000
Prevalence	0.1356	0.322	0.322	0.2203
Detection Rate	0.1356	0.322	0.322	0.2203
Detection Prevalence	0.1356	0.322	0.322	0.2203
Balanced Accuracy	1.0000	1.000	1.000	1.0000

The LOOCV was also used within SVM on the 99 top-ranked genes. The confusion matrix from the LOOCV procedure can be found in Table 5 (i). The estimated performance of the SVM classification algorithm had accuracy 61.02% with a 95% confidence interval of (53.41%, 68.24%). Of the 177 patients classified by the model, 6 were correctly classified as stage I, 43 as stage II, 39 as stage III, and 20 as stage IV. A total of 69 patients produced misclassified results.

TABLE 8. Stage Statistics for SVM by LOOCV Approach

Statistic	Stage I	Stage II	Stage III	Stage IV
Sensitivity	0.2500	0.7544	0.6842	0.5128
Specificity	0.9935	0.7417	0.7917	0.9130
Positive Predictive Value	0.8571	0.5811	0.6094	0.6250
Negative Predictive Value	0.8941	0.8641	0.8407	0.8690
Prevalence	0.1356	0.3220	0.3220	0.2203
Detection Rate	0.0339	0.2429	0.2203	0.1130
Detection Prevalence	0.0396	0.4181	0.3616	0.1808
Balanced Accuracy	0.6217	0.7480	0.7379	0.7129

The results in Table 8 above show that the model exhibits above average accuracy in classifying the patients into the four different colon cancer stages. The specificity and negative predictive values seems appropriate. The sensitivity, particularly for cancer stage I is very low. The model has positive predictive values for each stage at 85.71%, 58.11%, 60.94% and 62.5% for the colon cancer stages I, II, III, and IV respectively.

4.4. Random Forests Methodology

The random forests training and testing on the same dataset yielded the confusion matrix shown in Table 3 (ii). The overall estimated re-substitution accuracy for this approach was 100% with a 95% confidence interval of (97.94%,100%). All the patients were correctly classified. Positive predictive values, negative predictive values, specificity and sensitivity for each of the four stages were 100%.

The LOOCV approach resulted into 177 models, each built from $n - 1$ samples and validated on the one sample held out. Each of the 177 models were used the default *mtry* value. The number of trees used was 1000. This is important because sensitivity increases with the increase in *ntree* (results not included). Table 5 (ii) presents the confusion matrix obtained by LOOCV approach.

The overall estimated accuracy for the LOOCV approach in random forest was 62.71% with a 95% confidence interval of (55.14%, 69.85%). The correctly classified subjects were 7, 41, 43 and 20 in stages I, II, III, and IV respectively. A total of 66 patients were misclassified. The positive predictive values (the probability that subjects were accurately classified in their true stages) were 100%, 60.29%, 62.32%, and 60.61% for the stages I, II, III, and IV respectively (see Table 9 below).

TABLE 9. Stage Statistics for Random Forest by LOOCV Approach

Statistic	Stage I	Stage II	Stage III	Stage IV
Sensitivity	0.2917	0.7193	0.7544	0.5128
Specificity	1.0000	0.7750	0.7833	0.9058
Positive Predictive Value	1.0000	0.6029	0.6232	0.6061
Negative Predictive Value	0.9000	0.8532	0.8704	0.8681
Prevalence	0.1356	0.3220	0.3220	0.2203
Detection Rate	0.0396	0.2316	0.2429	0.1130
Detection Prevalence	0.0396	0.3842	0.3898	0.1864
Balanced Accuracy	0.6458	0.7471	0.7689	0.7093

4.5. Lasso Methodology

The lasso training and testing on the same dataset yielded the confusion matrix shown in Table 3 (iv). The re-substitution accuracy of training and testing the Lasso on the same data set was 100% with a 95% confidence interval of (97.94%, 100%). All the patients were correctly classified into their respective stages. No misclassified patients were obtained.

The lasso LOOCV approach attained a classification accuracy of 64.97% with a 95% confidence interval of (57.46%, 71.98%). The patients or samples that were correctly classified by the model were 14, 40, 38, and 23 for the colon cancer stages I, II, III, and IV respectively.

TABLE 10. Colon Cancer Stage Statistics for the Lasso by LOOCV Approach

Statistic	Stage I	Stage II	Stage III	Stage IV
Sensitivity	0.5833	0.7018	0.6667	0.5897
Specificity	0.9412	0.8667	0.7833	0.9203
Positive Predictive Value	0.6087	0.7143	0.5937	0.6765
Negative Predictive Value	0.9351	0.8595	0.8319	0.8881
Prevalence	0.1356	0.3220	0.3220	0.2203
Detection Rate	0.0791	0.2260	0.2147	0.1299
Detection Prevalence	0.1299	0.3164	0.3616	0.1921
Balanced Accuracy	0.7623	0.7842	0.7250	0.7550

It is evident from Table 10 above that the model attains higher specificity and negative predicted values and slightly above average values for sensitivity and positive predicted values.

4.6. Application of Logistic Regression Methodology

The multinomial logit regression (MLR) model was also used on the top 99 genes selected by random forests variable importance using LOOCV. The 'nnet' R package was used. Table 3 (v) shows the confusion matrix obtained after training and validating the MLR on the same colon cancer data set. The re-substitution accuracy of training and testing the MLR on the same data set was 100% with a 95% confidence interval of (97.94%, 100%). All the patients were correctly classified into their respective stages. From Table 4, there are no false positives and no false negatives. This is shown by 100% specificity and sensitivity.

The LOOCV approach was also used. The accuracy of training and testing the MLR by the LOOCV approach was 49.72% with a 95% confidence interval of (42.13%, 57.32%). Of all the 177 sample patients, 88 (sum of the main diagonal) were correctly classified into their

respective stages. A total of 89 patients or samples were misclassified. This accuracy is slightly below average, even though the classification approach was intensive and inclusive.

TABLE 11. Stage Statistics for the Logistic Regression by LOOCV Approach

Statistic	Stage I	Stage II	Stage III	Stage IV
Sensitivity	0.4167	0.4912	0.5263	0.5128
Specificity	0.9085	0.8083	0.7333	0.8551
Positive Predictive Value (PPV)	0.4167	0.5490	0.4839	0.5000
Negative Predictive Value (NPV)	0.9085	0.7698	0.7652	0.8613
Prevalence	0.1356	0.3220	0.3220	0.2203
Detection Rate	0.0565	0.1582	0.1695	0.1130
Detection Prevalence	0.1356	0.2881	0.3503	0.2260
Balanced Accuracy	0.6626	0.6498	0.6298	0.6839

The sensitivity and the PPV are low while the specificity and NPV are high for all the four cancer stages. This indicates that the probability of false positives is low (high specificity) while the probability of false negatives is moderate (low sensitivity).

TABLE 12. Kendall's Rau-b rank Correlation Estimates and Confidence Intervals

METHOD	LOOCV	95% C.I.	TRAINING	95% C.I.
SVM	0.4514	(0.3291, 0.5737)	1.0000	(0.9962, 1.0000)
RF	0.4390	(0.3098, 0.5682)	1.0000	(0.9962, 1.0000)
CART	0.3684	(0.2498, 0.4871)	0.7946	(0.6956, 0.8936)
LASSO	0.5233	(0.4059, 0.6407)	1.0000	(0.9962, 1.0000)
LOGISTIC REGRESSION	0.3423	(0.2200, 0.4647)	1.0000	(0.9962, 1.0000)

The Kendall's tau-b rank correlation, which is a non-parametric measure of correlation, assesses the statistical associations between two variables based on the ranks of the data. In our case, the variables are the predicted and the actual patient classes. The positive correlation coefficients signify that the ranks of both the variables are increasing. On the other

hand, the negative correlation coefficients signify that as the rank of one variable is increased, the rank of the other variable is decreased. The strength of the rank correlation, positively and negatively, increases as the rank correlation coefficient approaches +1 or -1 respectively.

The estimated Kendall's tau-b rank correlation coefficient of the CART model was 0.3684 with a 95% confidence interval (0.2498, 0.4871) for the LOOCV approach and an estimate of 0.7946 with a 95% confidence interval (0.9962, 1.000) for training and testing on the same data set approach. For the LOOCV approach, the lasso exhibited a moderate positive correlation, slightly higher than the SVM and the random forests in that order. The LOOCV rank correlation coefficients for both CART and MLR signify weak positive associations. The Kendall's tau-b values for all models, both the training and LOOCV approaches, are positive, meaning that the ranks of both variables are increasing. Except the CART training model (which also showed a strong positive rank correlation), all the training models had a perfect positive rank correlation. For the Kendall's tau-b rank correlation coefficients for all the models, see Table 12 above.

4.7. Variable Selection and Importance

Not all the variables in a given data set have the same importance to the model under construction. Some variables are more important, or contribute more to the model than others. Table 13 in appendix A lists 99 top-ranked genes out of the 54,675 genes. The selected genes were further used in model development. These genes were selected by the random forest variable importance using the LOOCV approach. Figures 1-12 in appendix A clearly shows the different measures used by different models to assess the importance of the variables as well as ordering them from the least important to the most important. The most important genes or

variables have a higher score and appear on the y-axes of the Figures and the importance decreases downwards in that order.

CHAPTER 5

DISCUSSION OF RESULTS

The comparative analysis of the five classifiers shows that the lasso dominates them all. As per the training approach, the lasso, SVM, random forests, and the logistic regression attained similar results, the CART, although better, didn't yield similar results to the rest. Based on the LOOCV approach, a comprehensive comparative analysis can be made as each classifier obtained distinct results. Table 14 below displays the summary of the performance metrics for the ordinal data.

TABLE 14. Summary Results for all the Performance Metrics and the Prediction Models

Performance	SVM		RANDOM FOREST		RPART		LASSO		LOGISTIC REGRESSION	
Metrics	LOOCV	TRAIN	LOOCV	TRAIN	LOOCV	TRAIN	LOOCV	TRAIN	LOOCV	TRAIN
Distance, d	0.3981	0.0824	0.3755	0.0824	0.5263	0.1481	0.3543	0.0824	0.5037	0.0824
Accuracy, Acc	0.6045	1.0000	0.6271	1.0000	0.4746	0.8701	0.6497	1.0000	0.4972	1.0000
Max Total MC, mc	0.0457	0.0824	0.0442	0.0824	0.031	0.0711	0.0534	0.0824	0.0301	0.0824
Kendall's Tau-b	0.4466	1.0000	0.439	1.0000	0.3684	0.7946	0.5233	1.0000	0.3423	1.0000

Considering the LOOCV approach, the lasso not only had a high accuracy, but also had the lowest distance and high Kendall's rank correlation. The lasso also attains the minimum distance (the smaller the distance, the better). The random forests come next in performance with regard to classification accuracy and distance (which is given as, $d = \sqrt{((1-acc)^2 + mc^2)}$) (Chang, George, and Tzu-Pin 2016). The SVM completes the three best classifiers, among the five classifiers presented on this thesis. The SVM has slightly lower distance and accuracy as compared to the lasso and the random forests.

The multinomial logistic regression and the CART have below average (less than 50%) LOOCV classification accuracies. On the contrast, these two classifiers attained the minimum maximum total misclassification costs (minimum *mc*) as shown in Table 13 above. This outcome may in turn demonstrate that as the classification accuracies increase, the maximum total misclassification costs seems to decrease and vice versa.

The 95% confidence intervals for the classification accuracies for LOOCV approaches for the three superior classifiers overlap for the most part (see Table 15 below). This clearly demonstrates that the three methods may collectively be said to attain more or less the same results, but individually distinct.

Not only does the multinomial logistic and the CART had lower classification accuracies, but also had the lowest Kendall's tau rank correlation. This might demonstrate that Kendall's tau rank correlation increases with increase in accuracy.

TABLE 15. Estimated Classification Accuracies and Confidence Intervals

METHOD	LOOCV	95% C.I.	TRAINING	95% C.I.
SVM	61.02%	(53.41%, 68.24%)	100%	(97.94%, 100%)
RF	62.71%	(55.14%, 69.85%)	100%	(97.94%, 100%)
CART	47.46%	(39.92%, 55.09%)	87.01%	(81.14%, 91.58%)
LASSO	64.97%	(57.46%, 71.98%)	100%	(97.94%, 100%)
LOGISTIC REGRESSION	49.72%	(42.13%, 57.32%)	100%	(97.94%, 100%)

On the other hand, the training and testing on the same data set approach resulted into the same results regarding classification accuracies for four classifiers, namely; SVM, random forests, the lasso and multinomial logistic regression. These classifiers all attained 100% classification accuracy

with a 95% confidence interval (97.94%, 100%). The lone classifier in this category that performed otherwise was the CART. It attained a classification accuracy of 87.01% with 95% confidence interval of (81.14%, 91.58%). This performance manifested itself in the Kendall's tau rank correlation too. These results show the model exhibits significantly greater accuracy in identifying patients who should be placed in the four different colon cancer stages.

The models developed by training and testing on the same data set will result into low accuracy when applied to another dataset. The performance attained during model building may not be achieved on a different data. The LOOCV is expected to perform either at the same level or better on a new data set. Models attained this way are believed to be parsimonious. They have very low bias but robust variation.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

Previous studies have come up with performance metrics for ordinal data, such as, predictive accuracy, the average mean absolute error (AMAE) and Kendall's tau, and drew pros and cons of each method in different settings. Each performance metric has its own strength and weakness, and so is the most commonly used ordinal data performance metric, the accuracy. The predictive accuracy's performance on imbalanced data sets for example, may be misleading. The classifier can achieve high accuracy for a class with the largest proportion of the sample due to classifying each sample into the majority class (Chang, George, and Tzu-Pin 2016). Therefore, not a single metric should be considered as being sufficient enough.

In practice, the cost of misclassifying a stage I patient into stage IV and vice versa or the cost of misclassifying a stage III patient as normal (has no colon cancer) cannot be the same. Such occurrences can be minimized, if not eliminated, by incorporating the cost of misclassification in determining the classifiers to be used in classification, without solemnly relying on the predictive accuracy which also has its own drawbacks. This therefore calls for a classifier that incorporates predictive accuracy and the misclassification cost, or more. Chang, George, and Tzu-Pin (2016) proposed distance, which is a trade-off between accuracy and the misclassification cost.

In this thesis, we looked further to select the most informative or important variables, models and classifiers and assess the performance of the proposed classifier, distance, along-side other performance metrics like the Kendall's rank correlation using two approaches on five possible classifiers. These approaches included training and testing the models on the same data sets and by using the leave one out cross validation (LOOCV) approach.

The models considered for investigating and assessing the performance metrics were; support vector machines (SVM), random forests (RF), classification and regression trees (CART), the least absolute shrinkage and selection operator (lasso) and the multinomial logistic regression (MLR).

The results from this research are a step toward unravelling the most informative genes that can provide the much needed information as far as cancer classification is concerned. They also provide important details to the classification algorithms that perform better with ordinal data while incorporating the misclassification cost. These classification algorithms could allow health care practitioners to more selectively classify patients and thereby reduce deaths due to placement on a treatment regimen that doesn't suit a particular colon cancer stage.

We recommend that developing a classifier with the inbuilt proposed metric, distance, will incorporate the trade-off within the calculations that results into the predicted stages. This will definitely make the comparison with other performance metrics simple. Further research can be done using other cancer types data to find ways in which these results can be generalized.

APPENDICES

APPENDIX A

THE 99 TOP-RANKED GENES AND PLOTS FROM THE CLASSIFICATION MODELS

TABLE 13. The 99 Top-Ranked Important Genes Obtained by Random Forest Using LOOCV

208772_at	209499_x_at	224020_at
227461_at	39966_at	224478_s_at
202909_at	227188_at	227718_at
1553542_at	236065_at	1569157_s_at
217412_at	1563182_at	203797_at
222751_at	208773_s_at	214049_x_at
226017_at	210808_s_at	215359_x_at
218919_at	213528_at	241892_at
204511_at	34697_at	1557359_at
235698_at	216705_s_at	1558882_at
204273_at	225270_at	214705_at
217231_s_at	204937_s_at	228202_at
236069_at	209510_at	1552427_at
225835_at	216682_s_at	1554183_s_at
203584_at	241364_at	209438_at
237268_at	1570639_at	221346_at
206945_at	219801_at	224967_at
224211_at	222484_s_at	227233_at
209757_s_at	212074_at	1556327_a_at
1561512_at	221109_at	1560405_at
201849_at	225955_at	204823_at
213820_s_at	229953_x_at	224083_s_at
210945_at	232216_at	224452_s_at
201945_at	235209_at	242585_at
226364_at	224413_s_at	1552445_a_at
203236_s_at	225726_s_at	211495_x_at
225265_at	241658_at	225550_at
204274_at	1556296_at	230650_at
204404_at	1569676_at	232397_at
223231_at	201985_at	235636_at
234113_at	217618_x_at	236675_at
235210_s_at	219583_s_at	237172_at
242931_at	220187_at	238397_at

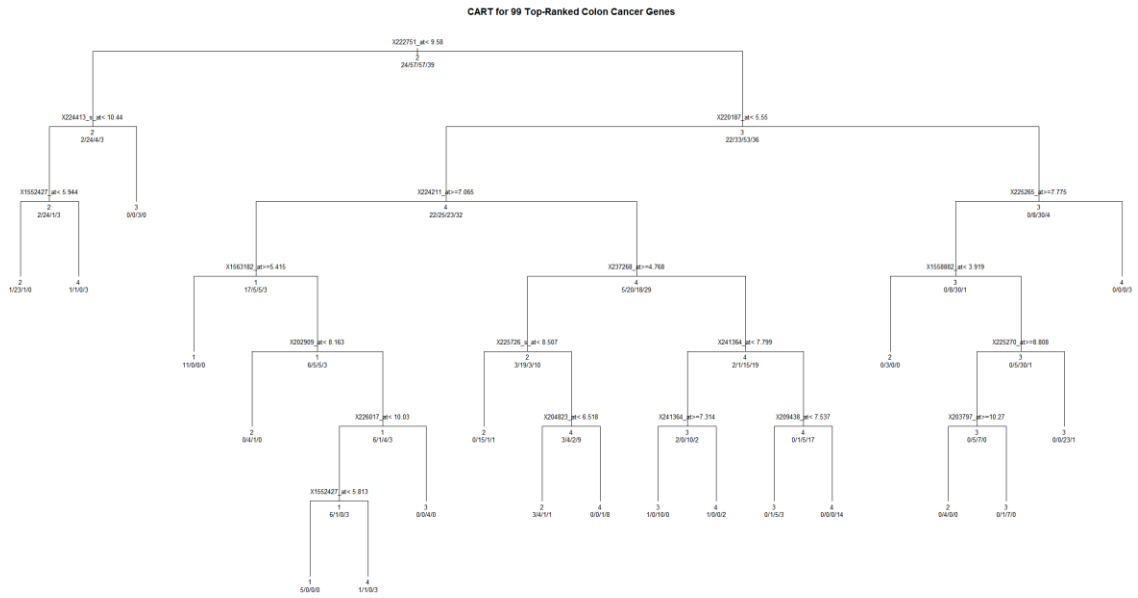


FIGURE 1. Decision tree for CART training approach.

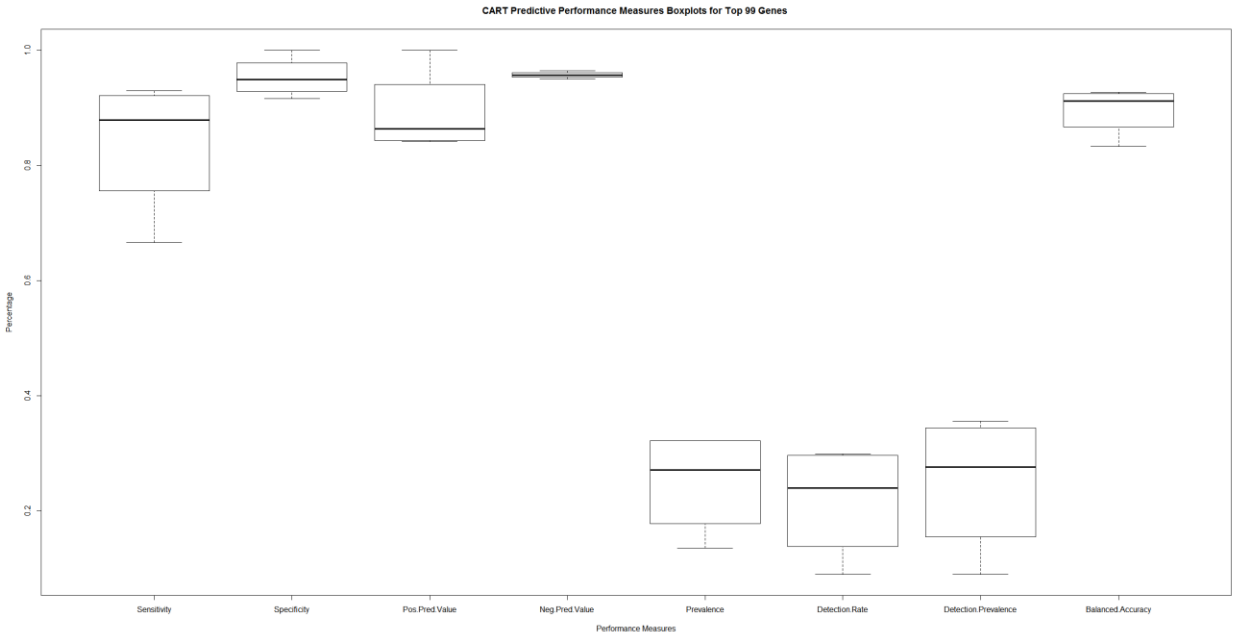


FIGURE 2. Boxplot of the performance measures for the CART training approach.

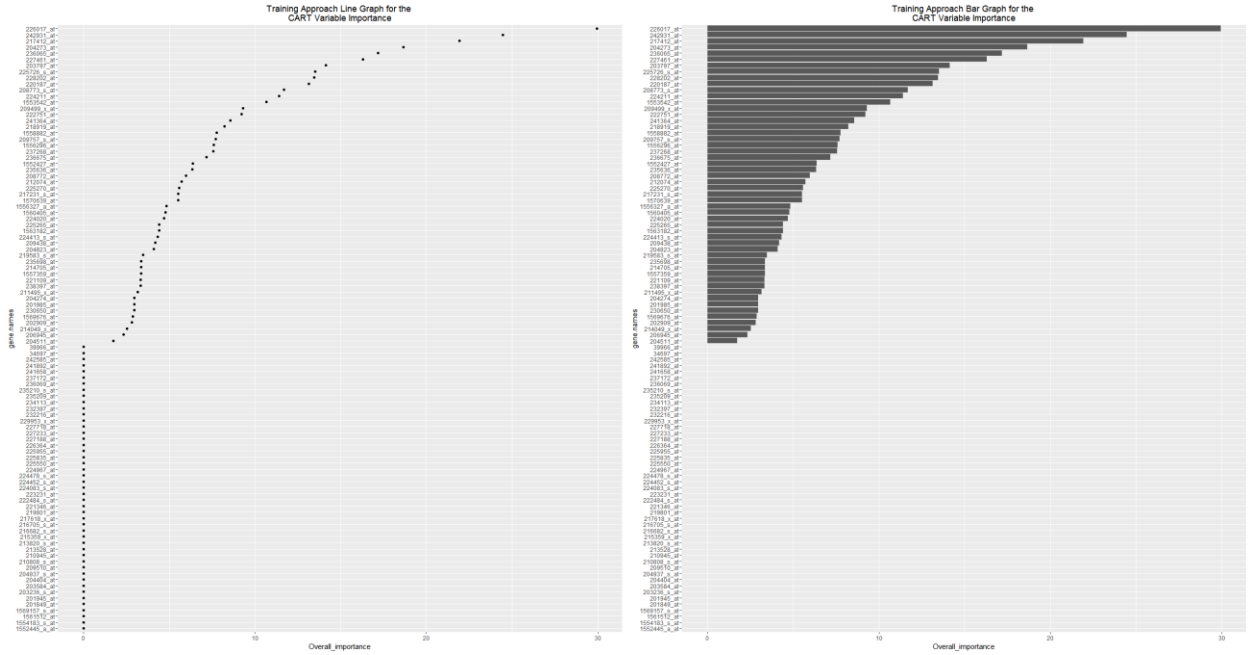


FIGURE 3. Variable importance plots for the CART training approach.

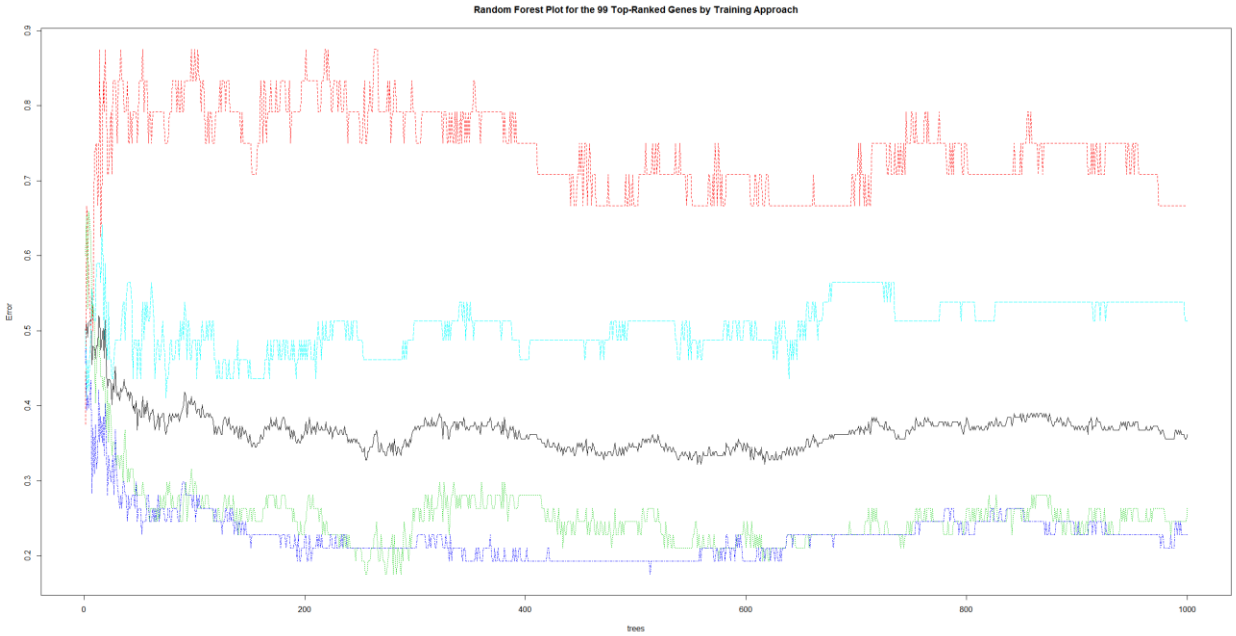


FIGURE 4. The random forest model plot for the training approach.

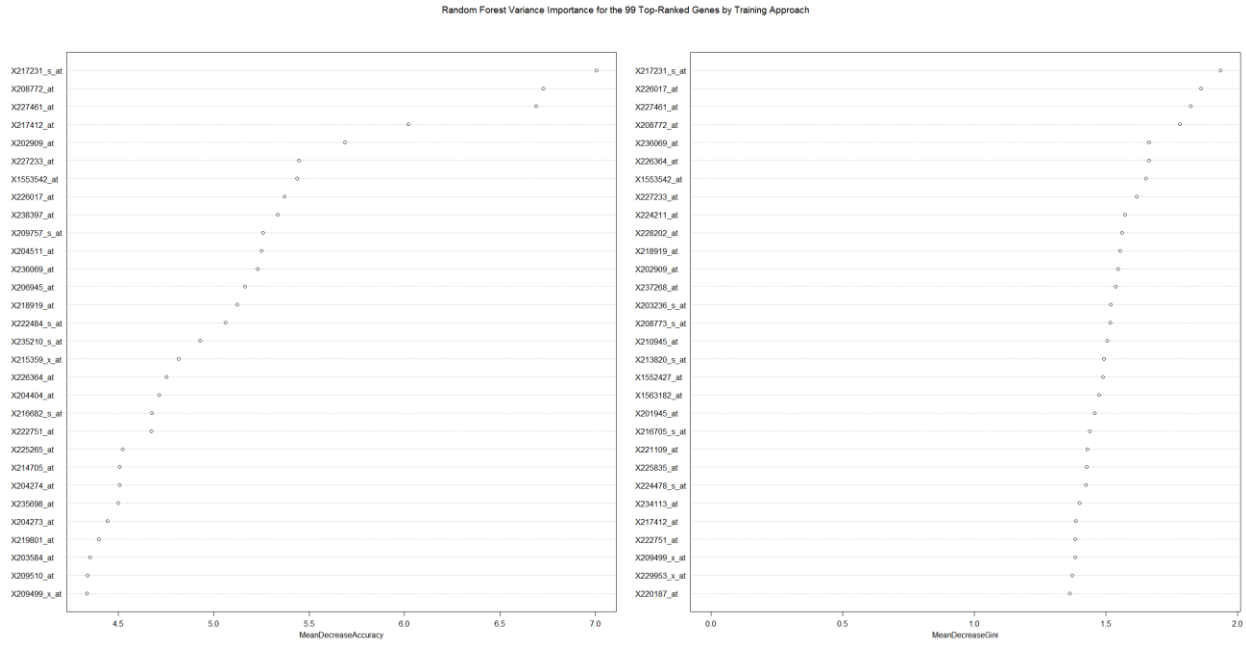


FIGURE 5. Random forest variable importance plots for the training approach.

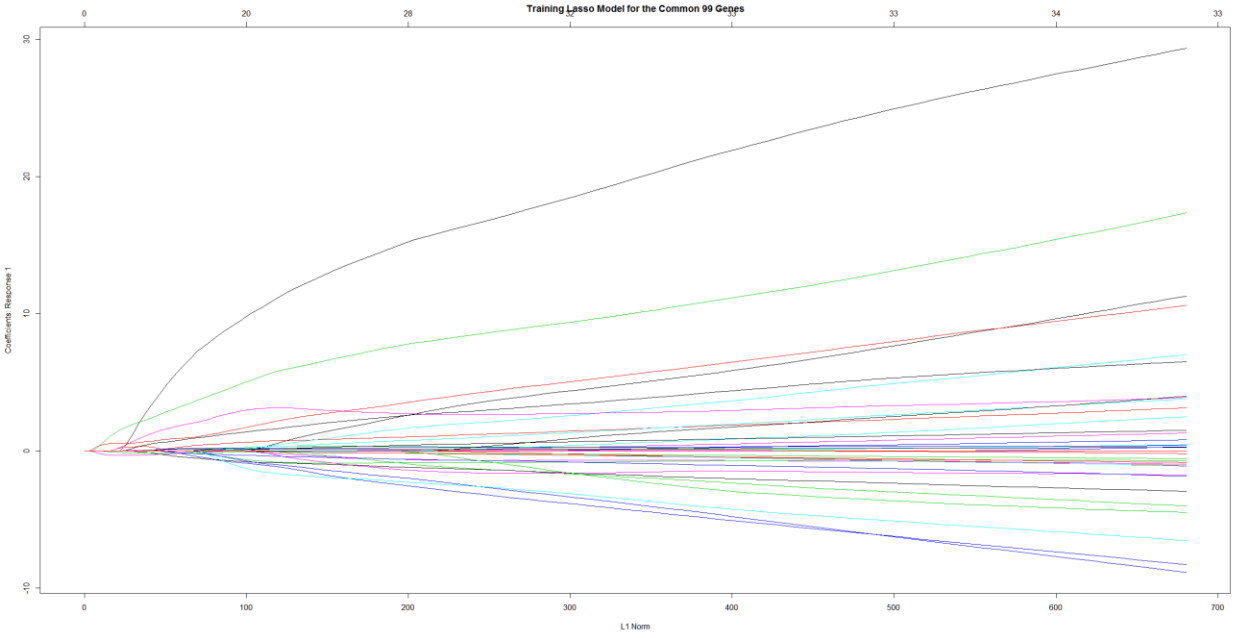


FIGURE 6. Colon cancer stage I lasso model coefficients for the training approach.

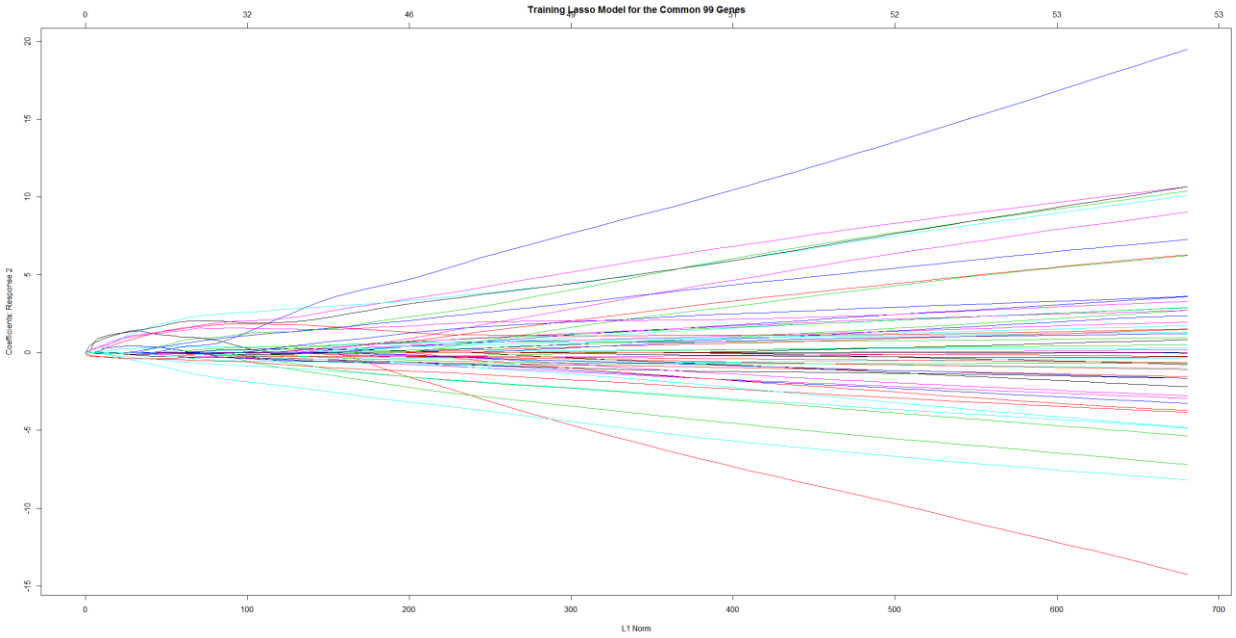


FIGURE 7. Colon cancer stage II lasso model coefficients for the training approach.

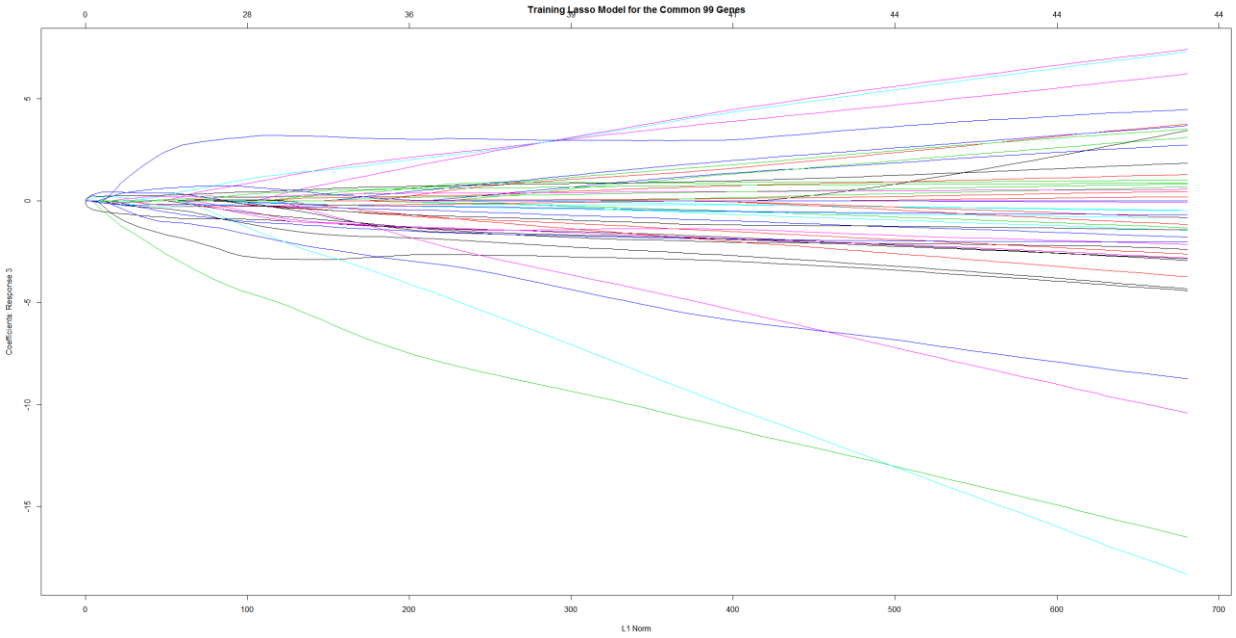


FIGURE 8. Colon cancer stage III lasso model coefficients for the training approach.

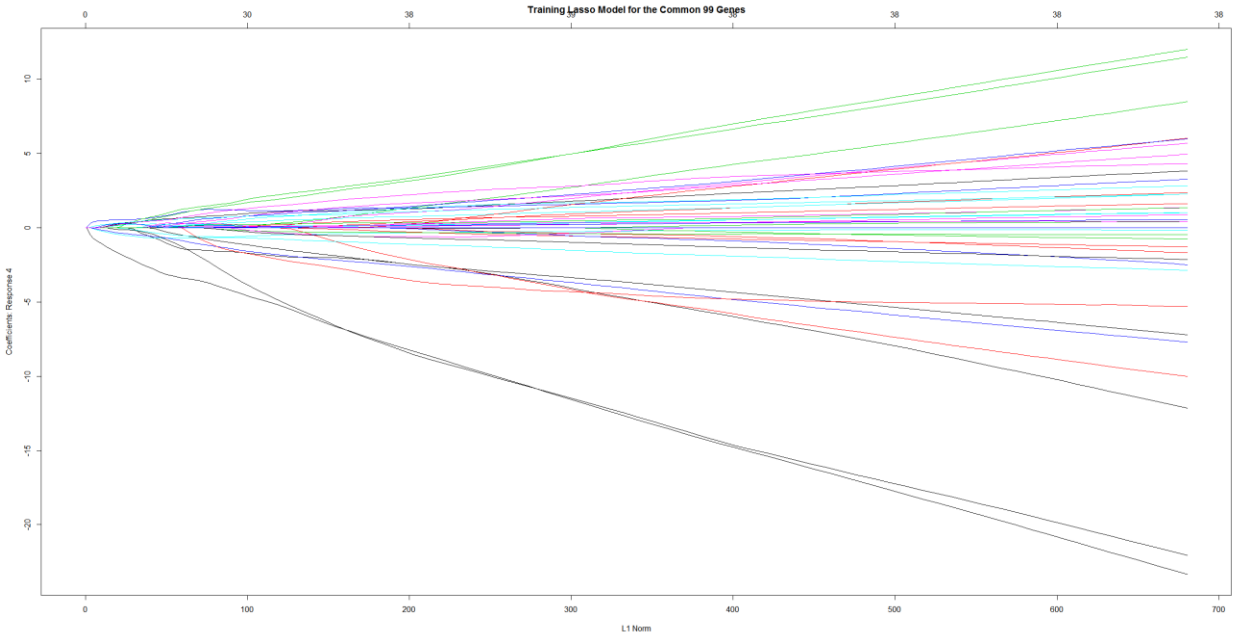


FIGURE 9. Colon cancer stage IV lasso model coefficients for the training approach.

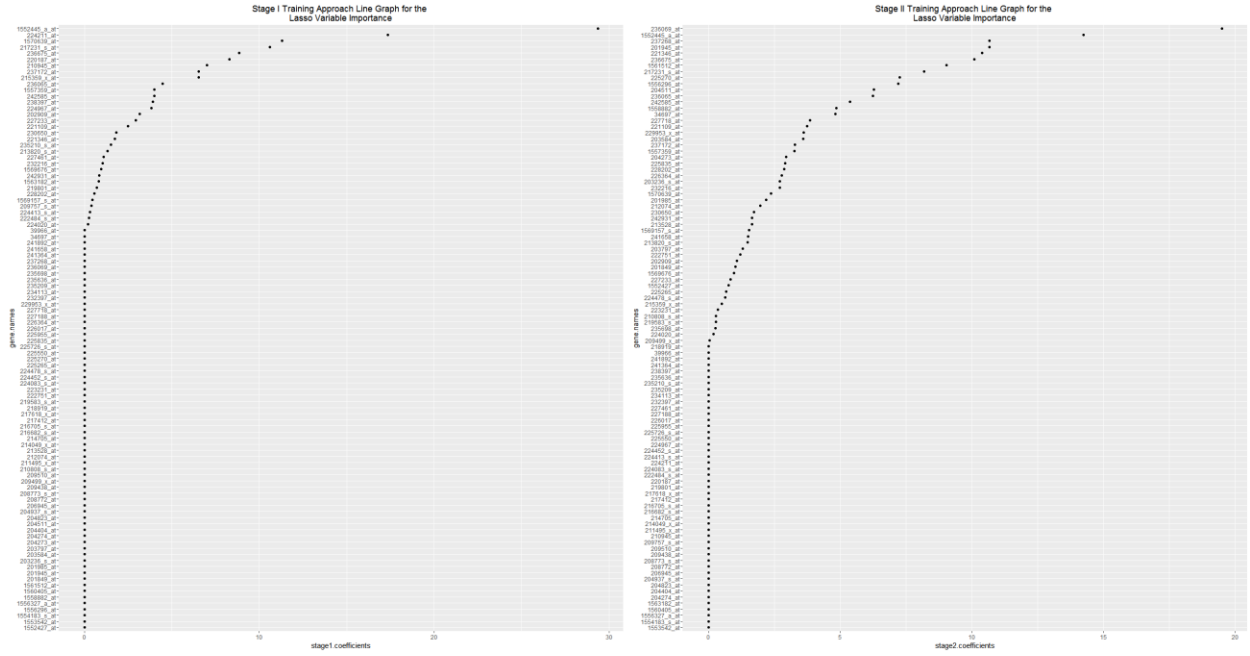


FIGURE 10. Variable importance plots for the lasso stages I and II training approach.

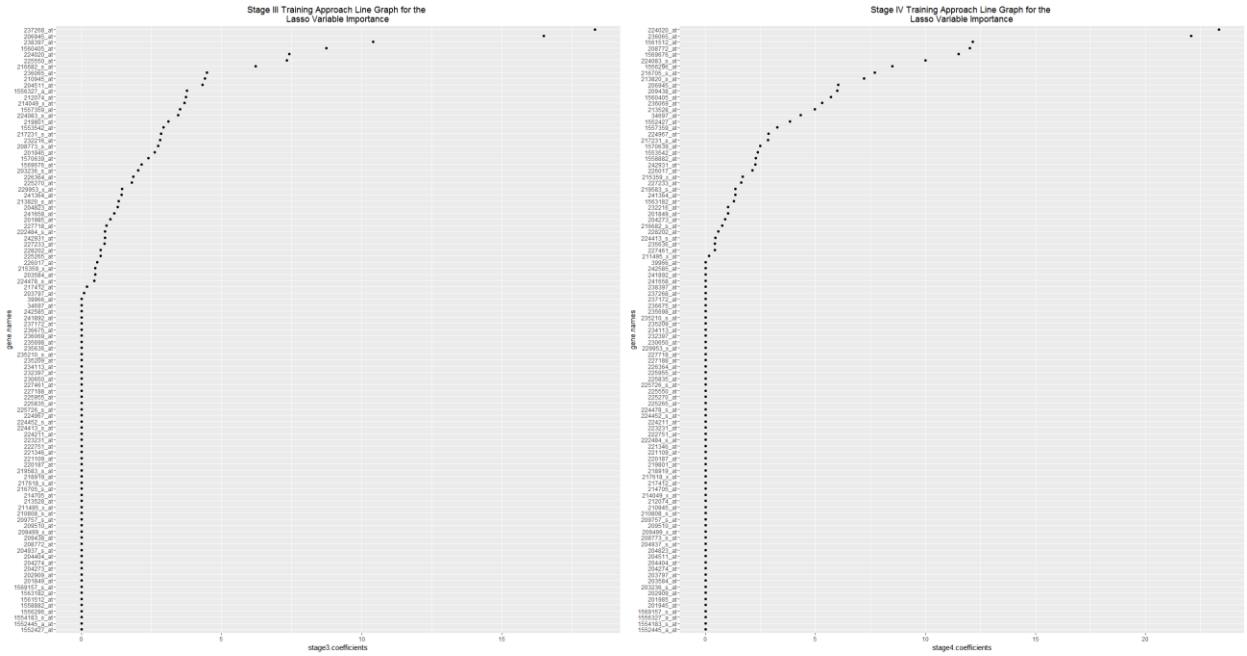


FIGURE 11. Variable importance plots for the lasso stages III and IV training approach.

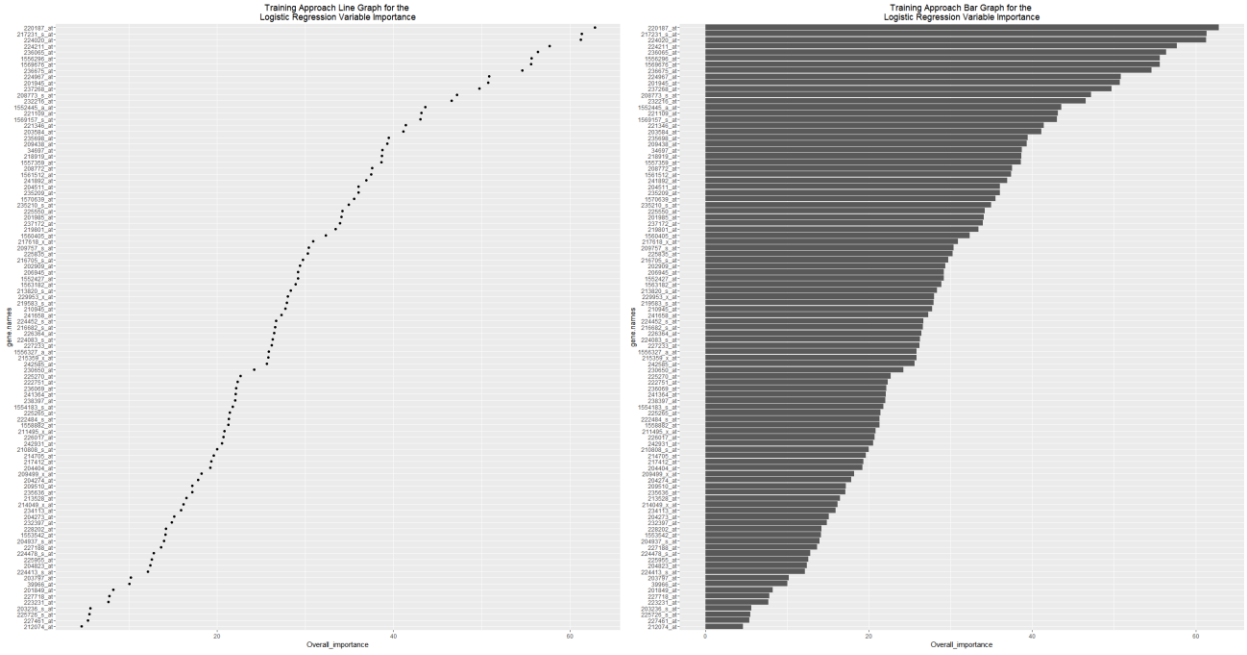


FIGURE 12. Variable importance plots for the logistic regression training approach.

APPENDIX B

R CODE FOR THE ANOVA F-TEST AND LOOCV TO SELECT THE SIGNIFICANT GENES

```
###COLON CANCER DATA ANALYSIS
###Data Cleaning
memory.limit(size=50000)
dataset<- read.csv("C:/Users/Denis/Desktop/fdrive2/ExcelFiles/colon17536.csv", header=FALSE,
stringsAsFactors=FALSE)
dataset2=dataset[-(1:45),]
dataset2=dataset2[-(32:33),]
dataset2=dataset2[-(12:32),]
rownames(dataset2)[12:nrow(dataset2)]=dataset2[12:nrow(dataset2),1]
dataset2[1,]=gsub("age: ","",dataset2[1,])
rownames(dataset2)[1:11]=c("age","gender", "ethinicity", "ajcc_stage","grade", "overall_event
(death from any cause)", "dss_event (disease specific survival; death from cancer)", "dfs_event
(disease free survival; cancer recurrence)", "overall survival follow-up time","dss_time",
"dfs_time")
dataset2[2,]=gsub("gender: ","",dataset2[2,])
dataset2[3,]=gsub("ethnicity: ","",dataset2[3,])
dataset2[4,]=gsub("ajcc_stage: ","",dataset2[4,])
dataset2[5,]=gsub("grade: ","",dataset2[5,])
dataset2[6,]=gsub("overall_event (death from any cause): ","",dataset2[6,])
dataset2[7,]=gsub("dss_event (disease specific survival; death from cancer): ","",dataset2[7,])
dataset2[7,]<-gsub(pattern = "dss_event (disease specific survival; death from cancer): ",
replacement ="", x = dataset2[7,])
dataset2[8,]=gsub("dfs_event (disease free survival; cancer recurrence): ","",dataset2[8,])
dataset2[9,]=gsub("overall survival follow-up time: ","",dataset2[9,])
dataset2[10,]=gsub("dss_time: ","",dataset2[10,])
dataset2[11,]=gsub("dfs_time: ","",dataset2[11,])
dataset3<-dataset2[-(1:11),]
dataset3[,1]=NULL
dataset3<-dataset3[-54677,]
colnames(dataset3)<-dataset3[1,]
```

```
dataset3 <- dataset3[-1,]
dataset4=apply(dataset3,2,as.numeric)
genescol<-data.frame(t(dataset4))
cols=data.frame(colnames(genescol))
library(dplyr)
library(plyr)
cols <- rename(cols, c("colnames.genescol."="ID"))
gen=data.frame(row.names(dataset3))
colnames(gen)<- c("ID_REF")
#gen <- rename(gen, c("row.names.dataset2..13.54687."="ID_REF"))
genes<-data.frame(cols$ID,gen$ID_REF)
dataset4=cbind(gen, dataset4)
rownames(dataset4) <- dataset4$ID_REF
dataset4$ID_REF <- NULL
genescol<-data.frame(t(dataset4))
### The Four Colon Cancer Stages
stage=numeric(177)
stage[1:24]=1
stage[25:81]=2
stage[82:138]=3
stage[139:177]=4
stage=as.factor(stage)
######EXPLORATORY DATA ANALYSIS
ajcc_stage<-apply(dataset2[4,], 1, as.factor)
ajcc_age<-apply(dataset2[1,], 1, as.numeric)
ajcc_gender<-apply(dataset2[2,], 1, as.factor)
ajcc_grade<-apply(dataset2[5,], 1, as.factor)
ajcc_ethnicity<-apply(dataset2[3,], 1, as.factor)
ajcc_survival<-apply(dataset2[9,], 1, as.factor)
summary(ajcc_grade)
summary(ajcc_gender)
```

```
table(ajcc_ethnicity,ajcc_stage)
summary(ajcc_ethnicity)
summary(ajcc_age)

#####ANOVA F TEST
####METHOD 1 FOR EXTRACTING P-VALUES
stage<-as.factor(unlist(stage))

for(i in 1:ncol(genescol)) {
  fit=lm(genescol[,i]~stage)
pval[i]=pf(summary(fit)$fstatistic[1],summary(fit)$fstatistic[2],summary(fit)$fstatistic[3],
lower.tail=F)
  }
pval=data.frame(pval)
write.csv(pval,file='stage pvalues.csv')
write.csv(pval, 'C:/Users/Denis/Desktop/P VALUES/pval_ANOVAF_TEST8.csv')

####METHOD 2 FOR EXTRACTING P-VALUES
for(i in 1:ncol(genescol)) {
pval1[i]<- anova(lm(genescol[,i]~stage))[1,5]
}
pval1=data.frame(pval1)
write.csv(pval1,file='stage pvalues6.csv')
write.csv(pval1, 'C:/Users/Denis/Desktop/P VALUES/pval_ANOVAF_TEST7.csv')
###TOP 1670 GENES||ALPHA=0.01
pval<-read.csv("C:/Users/Denis/Desktop/P VALUES/pval_ANOVAF_TEST8.csv", header=T,
stringsAsFactors=FALSE)
pval$X<-NULL
genes_pval <- data.frame(c(genes, pval))
colnames(genes_pval) <- c("X","gene", "pval")
genes_pval<-data.frame(genes_pval)
```

```
### Merging Genetic data with their respective p values
colon<-data.frame(cbind(dataset4, pval))

### Subsetting the Top 1670 gene|| p-value <= 0.01
pvalsig1670<-subset(colon, subset=(colon$pval<=0.01))
pvalsig1670$pval<-NULL
pvalsig1670<-as.data.frame(pvalsig1670)
pvalsig1670t<-t(pvalsig1670)
pvalsig1670t<-data.frame(pvalsig1670t)

##############################################################################
library(MASS); library(class); library(rpart); # recursive partitioning, tree predictors....
library(rpart.plot); library(tree); library(e1071); library(randomForest); library(mlbench);
library(mvtnorm); library(ipred); library(plyr) ; library(car); library(kernlab);
library(caret); require(ggplot2); require(foreign); require(reshape2);
##############################################################################
### RANDOM FOREST & LOOCV FOR TOP 100 GENES                                ####
##############################################################################
library(sqldf)
set.seed(698)
gini_ind <- list()
mda_dat <- list()
mdg_dat <- list()
ID.MDG.MDA <- list()
RF_MDA.FINAL1 <- list()
RF_MDA.FINAL2 <- list()
predRF1670<-c()
prediction1670<-c()
c3 = 0
m3 = 0
```

```
trainingdata1670<-pvalsig1670t
trainingdata1670<-scale(trainingdata1670)
training1670<-data.frame(cbind(trainingdata1670, stage))
for(i in 1:nrow(training1670)) {
  loocv.predictorRF1670 <- training1670[i, ]
  patient.predictorRF1670 <- training1670[-c(i),]

 ###Determining best mtry value
  #mtry1670 <- tuneRF(patient.predictorRF1670[,-101], patient.predictorRF1670$stage,
ntreeTry=1000, stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
  #best.m1670 <- mtry1670[mtry1670[, 2] == min(mtry1670[, 2]), 1]
  #plot(mtry1670,main='RANDOM FOREST MTRY TUNING FOR TOP 100 GENES', type='l')
  #print(mtry1670)
  #print(best.m1670)
  ################################
  loocv.rf1670 <- randomForest(factor(patient.predictorRF1670$stage) ~
.,data=patient.predictorRF1670, importance=TRUE, proximity=TRUE,  ntree=1000, plot =
TRUE)#mtry1670=15,
  RFVarImp <- data.frame(importance(loocv.rf1670))
  #MD <- data.frame(sqldf("select * from  RFVarImp where maxMDG = max(MeanDecreaseGini)",
row.names = TRUE))
  new_data <- RFVarImp[order(-RFVarImp$MeanDecreaseAccuracy),]
  RF_MDA <- data.frame(new_data[1:100,])
  gini_ind[[i]] <- data.frame(rownames(RF_MDA))
  mda_dat[[i]] <- data.frame(RF_MDA$MeanDecreaseAccuracy)
  mdg_dat[[i]] <- data.frame(RF_MDA$MeanDecreaseGini)
  RF_MDA.FINAL1[[i]] <- data.frame(RF_MDA)
  RF_MDA.FINAL2[[i]] <- data.frame(Gene1=gini_ind[[i]]$Gene, MDAcc=mda_dat[[i]]$MDAcc,
MDGin=mdg_dat$MDGin)
MDAccuracy=mda_data[[i]]$MDAccuracy, MDGini=mdg_data$MDGini)
```

```
  loocv.predicted1670 <- predict(loocv.rf1670, loocv.predictorRF1670, type="response")
  prediction1670[i] <- loocv.predicted1670
  if (loocv.predictorRF1670$stage == loocv.predicted1670) {
    c3 = c3 + 1
  } else {
    m3 = m3 + 1
  }
  print(i)
  cat("m3 ", m3, " c3 ", c3)
  predRF1670 <- c(predRF1670, prediction1670 )
}
cat('\n\nc3:', c3, ' Accuracy:', c3/(c3 + m3), '\n')
cat('m3:', m3, ' Misclass_Rate:', m3/(c3 + m3), '\n')
write.csv(data.frame(c3, m3))
cont_table1670 = confusionMatrix(prediction1670, stage)
gini_index <- data.frame(gini_index)
GENE_MDG_MDA <- cbind(gini_index, mda_data, mdg_data)
write.csv(GENE_MDG_MDA,
'C:/Users/Denis/Desktop/LOOCV_RF_IMP.GENE_MDG_MDA.csv')
write.csv( mda_dat, 'C:/Users/Denis/Desktop/MDA1670_100.csv')
write.csv( mdg_dat, 'C:/Users/Denis/Desktop/MDG1670_100.csv')
write.csv(gini_ind, 'C:/Users/Denis/Desktop/gini_ind1670_100.csv')
write.csv(ID.MDG.MDA, 'C:/Users/Denis/Desktop/ID_MDG_MDA1670_100.csv')

gini_indD <- read.csv("C:/Users/Denis/Desktop/gini_indD.csv", sep="")
unique_gini_indD = unique(gini_indD)
library(plyr)
unique_gini_indD_counts=data.frame(ddply(gini_indD,.(genesID),nrow))
colnames(unique_gini_indD_counts)[2]="count"
unique_gini_indD_counts_sorted <-unique_gini_indD_counts[order(-
unique_gini_indD_counts$count),]
```

```
row.names(unique_gini_indD_counts_sorted)= NULL
write.csv(unique_gini_indD_counts_sorted,
'C:/Users/Denis/Desktop/unique_gini_indD_counts_sorted.csv')

###SELECTING TOP 100 GENES BY COUNT
final_data1<-subset(unique_gini_indD_counts_sorted,
subset=(unique_gini_indD_counts_sorted$count>=29))
final_data100 <- merge(final_data1, dataset4, by.x = "genesID", by.y="ID_REF")
final_data100$count <- NULL
final_data100<- t(final_data100)
colnames(final_data100) = final_data100[1,]
final_data100 =  final_data100[-1,]
final_data100 = data.frame(cbind( final_data100, stage))
write.csv(final_data100, 'C:/Users/Denis/Desktop/final_data100.csv')
```

APPENDIX C

R CODE FOR CLASSIFICIATION AND REGRESSION TREES (CART) METHOD

```
### LOOCV & RPART FOR TOP 99 GENES ###
predRP100<-c()
prediction100<-c()
correct100 = 0
misclass100 = 0
training.final.100<-final_data100
rownames(training.final.100) <- training.final.100[,1]
training.final.100$X<- NULL

for(i in 1:nrow(training.final.100)) {
  loocv.predictors100 = training.final.100[i, ]
  patient.predictors100 = training.final.100[-c(i),]
  rpart.100 <- rpart(factor(patient.predictors100$stage)~., method="class", data =
patient.predictors100)
  current.pred100 <- predict(rpart.100, loocv.predictors100[,-100], type = "class")
  prediction100[i]=current.pred100
  if (current.pred100 == loocv.predictors100$stage ) {
    correct100 = correct100 + 1
  } else {
    misclass100 = misclass100 + 1
  }
  print(i)
  cat("misclass100 ", misclass100, " correct100 ", correct100)
  predRP100<-c(predRP100, current.pred100)
}
cat('\n\ncorrect100:', correct100, ' Accuracy:', correct100/(correct100 + misclass100), '\n')
cat('misclass100:', misclass100, ' Misclass1 Rate:', misclass100/(correct100 + misclass100), '\n')
write.csv(data.frame(correct100, misclass100))
```

```
loocv_rpt.100=confusionMatrix(predRP100,training.final.100$stage)

loocv_rpt.100

###APPROACH II    :   TRAINING APPROACH

library(rpart)

training.final.100<-final_data100

rownames(training.final.100) <- training.final.100[,1]

training.final.100$X<- NULL

rpart.100 <- rpart(factor(training.final.100$stage)~., data = training.final.100, method="class",
control=rpart.control(minsplit = 10, xval=177, cp=0.0001))

current.prediction.100 <- predict(rpart.100, data=training.final.100[,-100], type = "class")

training_rpt.100=confusionMatrix(current.prediction.100, training.final.100$stage)

training_rpt.100


### plot tree

plot(rpart.100, uniform=TRUE, main="CART for 99 Top-Ranked Colon Cancer Genes")

text(rpart.100, use.n=TRUE, all=TRUE, cex=.8, pretty = 0)

printcp(rpart.100) # display the results

#plotcp(rpart.100) # visualize cross-validation results

summary(rpart.100) # detailed summary of splits


###Boxplot of model measures

t=confusionMatrix(current.prediction.100, training.final.100$stage)

bx100=t$byClass

bx100=data.frame(apply(bx100,2,as.numeric))

boxplot(bx100, ylab= "Percentage", xlab = "Performance Measures", main="CART
Predictive Performance Measures Boxplots for Top 99 Genes ")

means=table(mean(bx100[,1]),mean(bx100[,2]), mean(bx100[,3]), mean(bx100[,4]),
mean(bx100[,5]), mean(bx100[,6]), mean(bx100[,7]), mean(bx100[,8]))


### Variable Importance

cart_imp = varImp(rpart.100)
```

```
gene.names=substring(rownames(cart_imp), 2)

Overall_importance = cart_imp[1]

gene.rpt.imp =data.frame(cbind(gene.names, Overall_importance))

colnames(gene.rpt.imp) <- c("gene.names", "Overall_importance")

genes.rpt.coef_range <- range(0, gene.rpt.imp$Overall_importance)


library(ggplot2)

library(gridExtra)

gene.rpt.imp1 <-gene.rpt.imp <-data.frame(gene.nam=rownames(gene.rpt.imp), gene.rpt.imp,
row.names=NULL)


### Relevel the gene.names by Overall_importance

gene.rpt.imp1$gene.names <-factor(gene.rpt.imp1$gene.names,
levels=gene.rpt.imp1[order(gene.rpt.imp$Overall_importance), "gene.names"])

s <-ggplot(gene.rpt.imp1, aes(y=gene.names, x=Overall_importance)) +
geom_point(stat="identity") + ggtitle("Training Approach Line Graph for the \nCART Variable
Importance")

t <-ggplot(gene.rpt.imp1, aes(x = gene.names, y=Overall_importance)) +
geom_bar(stat="identity") + coord_flip()+ggtitle("Training Approach Bar Graph for the \nCART
Variable Importance")

grid.arrange(s, t, ncol=2)


### prune the tree

prun_rpart.100<- prune(rpart.100, cp=
rpart.100$cptable[which.min(rpart.100$cptable[,"xerror"]),"CP"])

current.prediction_prun100 <- predict(prun_rpart.100, data=training.final.100, type = "class")

table(predicted=current.prediction_prun100, true=training.final.100$stage)

confusionMatrix(current.prediction_prun100, training.final.100$stage)


###plot the pruned tree

plot(prun_rpart.100, uniform=TRUE, main="Pruned Classification Tree for Top 100 colon
Cancer Genes")

text(prun_rpart.100, use.n=TRUE, all=TRUE, cex=.8)
```

```
###KENDALL'S TAU|| KENDALL'S CORRELATION COEFFICIENT

###RPART: LOOCV

RPART.LOOCV_kendall100 <- t(loocv_rpt.100$table)

Desc(RPART.LOOCV_kendall100, verbose="high")


### RPART: TRAINING

RPART.TRAINING_kendall100 <- t(training_rpt.100$table)

Desc(RPART.TRAINING_kendall100, verbose="high")

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

###ESTIMATING MAXIMUM TOTAL MISCLASSIFICATION COST FOR THE RPART
###LOOCV

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

###Optimization using the constrOptim (Lange, 2001) function

MAXTC.RPT.LOOCV4171 <-c()

for (i in 1:4){

  fQP.RPT.LOOCV417 <- function(b) {-sum(c(0,5,0,0)*b)+0.5*sum(b*b)}

  Amat.RPT.LOOCV417 <- cost_matrix
###matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),4,4)

  bvec.RPT.LOOCV417 <- t(loocv_rpt.100$table)

  MAXTC.RPT.LOOCV417 <- constrOptim(c(15,15,15,15), fQP.RPT.LOOCV417, NULL,
ui = t(Amat.RPT.LOOCV417), ci = bvec.RPT.LOOCV417[,i])

  MAXTC.RPT.LOOCV4171[i] <- MAXTC.RPT.LOOCV417$barrier.value

}

### derivative

MAXTC.RPT.LOOCV41 <-c()

for (i in 1:4){

  gQP.RPT.LOOCV417 <- function(b) {-c(0,5,0,0) + b}

  MAXTC.rpt.LOOCV41 <-constrOptim(c(15,15,15,15), fQP.RPT.LOOCV417, gQP.RPT417,
ui = t(Amat.RPT.LOOCV417), ci = bvec.RPT.LOOCV417[,i])
```

```
  MAXTC.RPT.LOOCV41[i] <- MAXTC.rpt.LOOCV41$barrier.value
}
### Now with maximisation instead of minimisation
MAXTC.RPT.LOOCV4 <-c()
for (i in 1:4){
  hQP.RPT.LOOCV417 <- function(b) {sum(c(0,5,0,0)*b)-0.5*sum(b*b)}
  max_TC.LOOCV417 <- constrOptim(c(15,15,15,15), hQP.RPT.LOOCV417, NULL, ui =
t(Amat.RPT.LOOCV417), ci = bvec.RPT.LOOCV417[,i],
                          control = list(fnscale = -1))
  MAXTC.RPT.LOOCV4[i] <- max_TC.LOOCV417$barrier.value
}
MAXTC.RPT.LOOCV4171 # minimisation
MAXTC.RPT.LOOCV41   # derivative
MAXTC.RPT.LOOCV4    ## Now with maximisation instead of minimisation
maxTC_rpt_LOOCV100 <- max(MAXTC.RPT.LOOCV4)
maxTC_rpt_LOOCV100
distance_RPT.LOOCV100= sqrt((1-(loocv_rpt.100$overall[1]))^2 +
(maxTC_rpt_LOOCV100)^2)
distance_RPT.LOOCV100
##############################################################################
###ESTIMATING MAXIMUM TOTAL MISCLASSIFICATION COST FOR THE RPART
###TRAINING APPROACH #
##############################################################################
###Optimization using the constrOptim (Lange, 2001) function
MAXTC.RPT4171 <-c()
for (i in 1:4){
  fQP.RPT417 <- function(b) {-sum(c(0,5,0,0)*b)+0.5*sum(b*b)}
  Amat.RPT417 <-
matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),4,4)
  bvec.RPT417 <- t(training_rpt.100$table)
```

```
  MAXTC.RPT417 <- constrOptim(c(15,15,15,15), fQP.RPT417, NULL, ui = t(Amat.RPT417), ci = bvec.RPT417[,i])

  MAXTC.RPT4171[i] <- MAXTC.RPT417$barrier.value

}

### derivative

MAXTC.RPT41 <-c()

for (i in 1:4){

  gQP.RPT417 <- function(b) {-c(0,5,0,0) + b}

  MAXTC.rpt41 <-constrOptim(c(15,15,15,15), fQP.RPT417, gQP.RPT417, ui = t(Amat.RPT417), ci = bvec.RPT417[,i])

  MAXTC.RPT41[i] <- MAXTC.rpt41$barrier.value

}

## #Now with maximisation instead of minimisation

MAXTC.RPT4 <-c()

for (i in 1:4){

  hQP.RPT417 <- function(b) {sum(c(0,5,0,0)*b)-0.5*sum(b*b)}

  max_TC.417 <- constrOptim(c(15,15,15,15), hQP.RPT417, NULL, ui = t(Amat.RPT417), ci = bvec.RPT417[,i], control = list(fnscale = -1))

  MAXTC.RPT4[i] <- max_TC.417$barrier.value

}

MAXTC.RPT4171 ## minimisation

MAXTC.RPT41   ## derivative

MAXTC.RPT4    ## Now with maximisation instead of minimisation

maxTC_rpt_training100 <- max(MAXTC.RPT4)

maxTC_rpt_training100


distance_RPT.TRAINING_col100= sqrt((1-(training_rpt.100$overall[1]))^2 + (maxTC_rpt_training100)^2)

distance_RPT.TRAINING_col100
```

APPENDIX D

R CODE FOR SUPPORT VECTOR MACHINE (SVM) LEARNING METHOD

```
final_data99 <- read.csv("C:/Users/Denis/Desktop/final_data100.csv")
library(MASS); library(class); library(rpart) # recursive partitioning, tree predictors....
library(rpart.plot); library(tree); library(e1071); library(randomForest)
library(mlbench); library(mvtnorm); library(ipred); library(plyr)
library(car); library(kernlab); library(caret); require(ggplot2)
require(foreign); require(reshape2)

##############################################################################
### APPROACH I: TRAINING & SVM FOR TOP 100 GENES                          ###
##############################################################################
training.final.100<-final_data100
rownames(training.final.100) <- training.final.100[,1]
training.final.100$X<- NULL

###Getting the tuning parameters ||gamma and cost
tune.svm.100 <- tune.svm(stage~., data = training.final.100, gamma = 2^(-6:6), cost = 2^(0:6),
            tunecontrol = tune.control(cross = length(training.final.100$stage)))
tune.svm.100

###Best perfomance: gamma=0.015625, cost=4
##cost=tune.svm.100$best.parameter[[2]], gamma=tune.svm.100$best.parameter[[1]]
training.svm.model.100 <- svm(stage~., data = training.final.100, gamma=0.015625 , cost=4,
scale=TRUE, kernel = "radial", type="C-classification")
current.prediction.svm.100 <- predict(training.svm.model.100, training.final.100[,-100])
prediction.svm.100 <- paste(current.prediction.svm.100)
training_svm.100 <- confusionMatrix(prediction.svm.100, training.final.100$stage)
training_svm.100
```

```
#############################################################################
### APPROACH II: LOOCV & SVM FOR TOP 100 GENES                            ###
#############################################################################
predSVM100 <- c()
loocv.prediction.100 <- c()
corr1 = 0
misc1 = 0
training.final.100<-final_data100
rownames(training.final.100) <- training.final.100[,1]
training.final.100$X<- NULL
for(i in 1:nrow(training.final.100)) {
  loocv.pred.100 <- training.final.100[i, ]
  patient.pred.100 <- training.final.100[-i,]
  tune.svm.loocv.100 <- tune.svm(stage~.,   data = patient.pred.100, gamma = 2^(-6:6), cost =
2^(0:6),
              tunecontrol = tune.control(cross = length(patient.pred.100$stage)))
  tune.svm.loocv.100


  ###Best perfomance: cost=tune.svm.loocv.100$best.parameter[[2]]
||gamma=tune.svm.loocv.100$best.parameter[[1]]
  ###Best perfomance: cost=4 & gamma=0.015625
  loocv.svm.model.100 <- svm(patient.pred.100$stage~., data =
patient.pred.100,gamma=0.015625 , cost=2 , kernel = "radial", type="C-classification")
  loocv.current.pred.100 <- predict(loocv.svm.model.100,loocv.pred.100[,-100])
  loocv.prediction.100[i] <- paste(loocv.current.pred.100)
  if ( loocv.pred.100$stage == loocv.current.pred.100 ) {
    corr1 = corr1 + 1
  } else {
    misc1 = misc1 + 1
  }
```

```
  print(i)
  cat("misc1 ", misc1, " corr1 ", corr1)
  predSVM100<-c(predSVM100, loocv.current.pred.100)
}
cat('\n\ncorr1:', corr1, ' Accuracy:', corr1/(corr1 + misc1), '\n')
cat('misc1:', misc1, ' Misc1 Rate:', misc1/(corr1 + misc1), '\n')
write.csv(data.frame(corr1, misc1))
loocv_svm.100 = confusionMatrix(predSVM100,training.final.100$stage)
loocv_svm.100

###COST MATRIX
cost_matrix <- matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),
nrow=4, dimnames=list("true"=c("1","2","2","4"), "predicted"=c("1","2","3","4")))

###KENDALL'S TAU|| KENDALL'S CORRELATION COEFFICIENT
library(Kendall)
library(DescTools)
#LOOCV
SVM.LOOCV_kendall.100 <- t(loocv_svm.100$table)
Desc(SVM.LOOCV_kendall.100, verbose="high")

#TRAINING
SVM.TRAINING_kendall.100 <- t(training_svm.100$table)
Desc(SVM.TRAINING_kendall.100, verbose="high")

##############################################################################
# ESTIMATING MAXIMUM TOTAL MISCLASSIFICATION COST FOR THE SVM LOOCV
##############################################################################
###Optimization using the constrOptim (Lange, 2001) function
```

```
MAXTC.SVM.LOOCV4171 <-c()

for (i in 1:4){

  fQP.SVM.LOOCV417 <- function(b) {-sum(c(0,5,0,0)*b)+0.5*sum(b*b)}

  Amat.SVM.LOOCV417 <- cost_matrix
#matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),4,4)

  bvec.SVM.LOOCV417 <- t(loocv_svm.100$table)

  MAXTC.SVM.LOOCV417 <- constrOptim(c(15,15,15,15), fQP.SVM.LOOCV417, NULL, ui
= t(Amat.SVM.LOOCV417), ci = bvec.SVM.LOOCV417[,i])

  MAXTC.SVM.LOOCV4171[i] <- MAXTC.SVM.LOOCV417$barrier.value

            }

# derivative

MAXTC.SVM.LOOCV41 <-c()

for (i in 1:4){

  gQP.SVM.LOOCV417 <- function(b) {-c(0,5,0,0) + b}

  MAXTC.svm.LOOCV41 <-constrOptim(c(15,15,15,15), fQP.SVM.LOOCV417,
gQP.SVM.LOOCV417, ui = t(Amat.SVM.LOOCV417), ci = bvec.SVM.LOOCV417[,i])

  MAXTC.SVM.LOOCV41[i] <- MAXTC.svm.LOOCV41$barrier.value

          }

## Now with maximisation instead of minimisation

MAXTC.SVM.LOOCV4 <-c()

for (i in 1:4){

  hQP.SVM.LOOCV417 <- function(b) {sum(c(0,5,0,0)*b)-0.5*sum(b*b)}

  SVM.max_TC.LOOCV417 <- constrOptim(c(15,15,15,15), hQP.SVM.LOOCV417, NULL, ui
= t(Amat.SVM.LOOCV417), ci = bvec.SVM.LOOCV417[,i],

                          control = list(fnscale = -1))

  MAXTC.SVM.LOOCV4[i] <- SVM.max_TC.LOOCV417$barrier.value

          }

MAXTC.SVM.LOOCV4171 # minimisation

MAXTC.SVM.LOOCV41   # derivative

MAXTC.SVM.LOOCV4    ## Now with maximisation instead of minimisation

maxTC_svm_LOOCV.100 <- max(MAXTC.SVM.LOOCV4)
```

```
maxTC_svm_LOOCV.100

distance_SVM.LOOCVl.100= sqrt((1-(loocv_svm.100$overall[1]))^2 +
(maxTC_svm_LOOCV.100)^2)

distance_SVM.LOOCVl.100

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

### ESTIMATING MAXIMUM TOTAL MISCLASSIFICATION COST FOR THE SVM
TRAINING APPROACH

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

###Optimization using the constrOptim (Lange, 2001) function

MAXTC.SVM4171 <-c()

for (i in 1:4){

  fQP.SVM417 <- function(b) {-sum(c(0,5,0,0)*b)+0.5*sum(b*b)}

  Amat.SVM417 <- cost_matrix
#matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),4,4)

  bvec.SVM417 <- t(training_svm.100$table)

  MAXTC.SVM417 <- constrOptim(c(15,15,15,15), fQP.SVM417, NULL, ui =
t(Amat.SVM417), ci = bvec.SVM417[,i])

  MAXTC.SVM4171[i] <- MAXTC.SVM417$barrier.value

}

# derivative

MAXTC.SVM41 <-c()

for (i in 1:4){

  gQP.SVM417 <- function(b) {-c(0,5,0,0) + b}

  MAXTC.svm41 <-constrOptim(c(15,15,15,15), fQP.SVM417, gQP.SVM417, ui =
t(Amat.SVM417), ci = bvec.SVM417[,i])

  MAXTC.SVM41[i] <- MAXTC.svm41$barrier.value

}


## Now with maximisation instead of minimisation

MAXTC.SVM4 <-c()

for (i in 1:4){
```

```
  hQP.SVM417 <- function(b) {sum(c(0,5,0,0)*b)-0.5*sum(b*b)}

  SVM.max_TC.417 <- constrOptim(c(15,15,15,15), hQP.SVM417, NULL, ui =
t(Amat.SVM417), ci = bvec.SVM417[,i],

                          control = list(fnscale = -1))

  MAXTC.SVM4[i] <- SVM.max_TC.417$barrier.value

}

MAXTC.SVM4171 # minimisation

MAXTC.SVM41   # derivative

MAXTC.SVM4    ## Now with maximisation instead of minimisation


maxTC_svm_training.100 <- max(MAXTC.SVM4)

maxTC_svm_training.100


distance_SVM.TRAINING_col.100= sqrt((1-(training_svm.100$overall[1]))^2 +
(maxTC_svm_training.100)^2)

distance_SVM.TRAINING_col.100
```

APPENDIX E

R CODE FOR RANDOM FORESTS (RF) METHOD

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

###RANDOM FOREST : TRAINING [USING DEFAULT MTRY]FOR TOP 100 GENES BY
###RANDOM FOREST VARIABLE IMPORTANCE ###

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

set.seed(698)

training.final.100<-final_data100

rownames(training.final.100) <- training.final.100[,1]

training.final.100$X<- NULL

train.rf.100 <- randomForest(factor(training.final.100$stage) ~ .,data=training.final.100,
importance=TRUE, proximity=TRUE, ntree=1000, plot = TRUE)

trainRF.predicted.100 <- predict(train.rf.100 , training.final.100[,-100], type="response")

training_rf.100=confusionMatrix(trainRF.predicted.100, training.final.100$stage)

training_rf.100


###Plots

par( mfrow = c(1,1) )

plot(train.rf.100, main="Random Forest Plot for the 99 Top-Ranked Genes by Training
Approach")

###Variable Importance

rf.100_imp = importance(train.rf.100)

varImpPlot(train.rf.100, main="Random Forest Variance Importance for the 99 Top-Ranked
Genes by Training Approach")

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

### RANDOM FOREST : LOOCV FOR TOP 99 GENES BY RANDOM FOREST
###VARIABLE IMPORTANCE ####

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

set.seed(698)

predRNF.100<-c()

prediction.100<-c()

corr3 = 0
```

```
misc3 = 0


set.seed(698)
training.final.100<-final_data100
rownames(training.final.100) <- training.final.100[,1]
training.final.100$X<- NULL
for(i in 1:nrow(training.final.100)) {
  loocv.predictorRF.100 <- training.final.100[i, ]
  patient.predictorRF.100 <- training.final.100[-c(i),]


  ###Determining best mtry value
mtry.100<- tuneRF(patient.predictorRF.100[,-100], patient.predictorRF.100$stage,
ntreeTry=1000, stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)
best.m.100 <- mtry.100[mtry.100[, 2] == min(mtry.100[, 2]), 1]
plot(mtry.100,main='RANDOM FOREST MTRY TUNING FOR TOP 90 COMMON GENES
BY RF IMPORTANCE & P-VALUE METHODS', type='l')
  print(mtry.100)
  print(best.m.100)
  loocv.rf.100 <- randomForest(factor(patient.predictorRF.100$stage) ~
.,data=patient.predictorRF.100, importance=TRUE, proximity=TRUE, mtry=best.m.100,
ntree=1000, plot = TRUE)
  loocv.predicted.100 <- predict(loocv.rf.100, loocv.predictorRF.100[,-100], type="response")
  prediction.100[i] <- loocv.predicted.100
  if (loocv.predictorRF.100$stage == loocv.predicted.100) {
    corr3 = corr3 + 1
         } else {
    misc3 = misc3 + 1
            }
  print(i)
  cat("misc3 ", misc3, " corr3 ", corr3)
  predRNF.100 <- c(predRNF.100, loocv.predicted.100 )
```

```
                    }
cat('\n\ncorr3:', corr3, ' Accuracy:', corr3/(corr3 + misc3), '\n')
cat('misc3:', misc3, ' Misc1 Rate:', misc3/(corr3 + misc3), '\n')
write.csv(data.frame(corr3, misc3))
loocv_rf.100=confusionMatrix(predRNF.100, training.final.100$stage)

###KENDALL'S TAU|| KENDALL'S CORRELATION COEFFICIENT
###RPART
#LOOCV1
RF.LOOCV_kendall.100 <- t(loocv_rf.100$table)
Desc(RF.LOOCV_kendall.100, verbose="high")

#TRAINING
RF.TRAINING_kendall.100 <- t(training_rf.100$table)
Desc(RF.TRAINING_kendall.100, verbose="high")
##############################################################################
###ESTIMATING MAXIMUM TOTAL MISCLASSIFICATION COST FOR THE RF LOOCV
##############################################################################
###Optimization using the constrOptim (Lange, 2001) function
cost_matrix <- matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),4,4)
MAXTC.RF.LOOCV4171 <-c()
for (i in 1:4){
  fQP.RF.LOOCV417 <- function(b) {-sum(c(0,5,0,0)*b)+0.5*sum(b*b)}
  Amat.RF.LOOCV417 <- cost_matrix
  bvec.RF.LOOCV417 <- t(loocv_rf.100$table)
  MAXTC.RF.LOOCV417 <- constrOptim(c(15,15,15,15), fQP.RF.LOOCV417, NULL,
ui = t(Amat.RF.LOOCV417), ci = bvec.RF.LOOCV417[,i])
  MAXTC.RF.LOOCV4171[i] <- MAXTC.RF.LOOCV417$barrier.value
```

```
}


# derivative
MAXTC.RF.LOOCV41 <-c()
for (i in 1:4){
  gQP.RF.LOOCV417 <- function(b) {-c(0,5,0,0) + b}
  MAXTC.rf.LOOCV41 <-constrOptim(c(15,15,15,15), fQP.RF.LOOCV417,
gQP.RF.LOOCV417, ui = t(Amat.RF.LOOCV417), ci = bvec.RF.LOOCV417[,i])
  MAXTC.RF.LOOCV41[i] <- MAXTC.rf.LOOCV41$barrier.value
}
## Now with maximisation instead of minimisation
MAXTC.RF.LOOCV4 <-c()
for (i in 1:4){
  hQP.RF.LOOCV417 <- function(b) {sum(c(0,5,0,0)*b)-0.5*sum(b*b)}
  RF.max_TC.LOOCV417 <- constrOptim(c(15,15,15,15), hQP.RF.LOOCV417, NULL, ui =
t(Amat.RF.LOOCV417), ci = bvec.RF.LOOCV417[,i],
                            control = list(fnscale = -1))
  MAXTC.RF.LOOCV4[i] <- RF.max_TC.LOOCV417$barrier.value
}
MAXTC.RF.LOOCV4171 # minimisation
MAXTC.RF.LOOCV41   # derivative
MAXTC.RF.LOOCV4    ## Now with maximisation instead of minimisation


maxTC_rf_LOOCV.100 <- max(MAXTC.RF.LOOCV4)
maxTC_rf_LOOCV.100


distance_RF.LOOCV.100= sqrt((1-(loocv_rf.100$overall[1]))^2 + (maxTC_rf_LOOCV.100)^2)
distance_RF.LOOCV.100
```

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

###ESTIMATING MAXIMUM TOTAL MISCLASSIFICATION COST FOR THE RPART
###TRAINING APPROACH #

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

###Optimization using the constrOptim (Lange, 2001) function

MAXTC.RF4171 <-c()

for (i in 1:4){

  fQP.RF417 <- function(b) {-sum(c(0,5,0,0)*b)+0.5*sum(b*b)}

  Amat.RF417 <- cost_matrix
###matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),4,4)

  bvec.RF417 <- t(training_rf.100$table)

  MAXTC.RF417 <- constrOptim(c(15,15,15,15), fQP.RF417, NULL, ui = t(Amat.RF417), ci =
bvec.RF417[,i])

  MAXTC.RF4171[i] <- MAXTC.RF417$barrier.value

}


# derivative

MAXTC.RF41 <-c()

for (i in 1:4){

  gQP.RF417 <- function(b) {-c(0,5,0,0) + b}

  MAXTC.rf41 <-constrOptim(c(15,15,15,15), fQP.RF417, gQP.RF417, ui = t(Amat.RF417), ci
= bvec.RF417[,i])

  MAXTC.RF41[i] <- MAXTC.rf41$barrier.value

}


## Now with maximisation instead of minimisation

MAXTC.RF4 <-c()

for (i in 1:4){

  hQP.RF417 <- function(b) {sum(c(0,5,0,0)*b)-0.5*sum(b*b)}
```

```
  RF.max_TC.417 <- constrOptim(c(15,15,15,15), hQP.RF417, NULL, ui = t(Amat.RF417),
ci = bvec.RF417[,i],  control = list(fnscale = -1))

  MAXTC.RF4[i] <- RF.max_TC.417$barrier.value

}

MAXTC.RF4171 # minimisation

MAXTC.RF41   # derivative

MAXTC.RF4    ## Now with maximisation instead of minimisation

maxTC_rf_training.100 <- max(MAXTC.RF4)

maxTC_rf_training.100

distance_RF.TRAINING_col.100= sqrt((1-(training_rf.100$overall[1]))^2 +
(maxTC_rf_training.100)^2)

distance_RF.TRAINING_col.100
```

APPENDIX F

R CODE FOR LASSO REGRESSION METHOD

```
#############################################################################
###LASSO REGRESSION
#############################################################################
###TRAINING APPROACH
library(glmnet)
training.final.100<-final_data100
rownames(training.final.100) <- training.final.100[,1]
training.final.100$X<- NULL
fit.lasso.100=glmnet(as.matrix(training.final.100[,-
100]),training.final.100$stage,alpha=1,family="multinomial")
predictions <- predict(fit.lasso.100, as.matrix(training.final.100[,-
100]),s=min(fit.lasso.100$lambda), type="class")
###Confusion Matrices
training.lasso.100 <- confusionMatrix(predictions, training.final.100$stage)
###Plots
plot(fit.lasso.100, main="Training Lasso Model for the Common 99 Genes")
histogram(fit.lasso.100$lambda) ##Histogram of the Lampda values
### Variable Importance
impor = varImp(fit.lasso.100, lambda=4.935061e-05)
gene.names=substring(rownames(impor), 2)
gene.coeff1=impor[1]; gene.coeff2=impor[2]; gene.coeff3=impor[3]; gene.coeff4=impor[4];
gene.lasso.imp =data.frame(cbind(gene.names, gene.coeff1, gene.coeff2, gene.coeff3,
gene.coeff4))
gene.lasso.imp1 =data.frame(cbind(gene.names, gene.coeff1))
row.names(gene.lasso.imp) <-gene.lasso.imp$gene.names
colnames(gene.lasso.imp) <- c("gene.names", "stage1.coeff", "stage2.coeff", "stage3.coeff",
"stage4.coeff")
genes.coef_range <- range(0, gene.lasso.imp$stage1.coeff, gene.lasso.imp$stage2.coeff,
gene.lasso.imp$stage3.coeff, gene.lasso.imp$stage4.coeff)
```

```
library(ggplot2); library(gridExtra);

###For Stage I

gene.lasso.imp.stg1 <- data.frame (gene.lasso.imp$gene.names, gene.lasso.imp$stage1.coeff)

colnames(gene.lasso.imp.stg1) <- c("gene.names", "stage1.coefficients")

gene.lasso.imp.stg111 <-gene.lasso.imp.stg11 <-
data.frame(gene.nam=rownames(gene.lasso.imp.stg1), gene.lasso.imp.stg1, row.names=NULL)

###For Stage II

gene.lasso.imp.stg2 <- data.frame (gene.lasso.imp$gene.names, gene.lasso.imp$stage2.coeff)

colnames(gene.lasso.imp.stg2) <- c("gene.names", "stage2.coefficients")

gene.lasso.imp.stg222 <-gene.lasso.imp.stg22 <-
data.frame(gene.nam=rownames(gene.lasso.imp.stg2), gene.lasso.imp.stg2, row.names=NULL)

###For Stage III

gene.lasso.imp.stg3 <- data.frame (gene.lasso.imp$gene.names, gene.lasso.imp$stage3.coeff)

colnames(gene.lasso.imp.stg3) <- c("gene.names", "stage3.coefficients")

gene.lasso.imp.stg333 <-gene.lasso.imp.stg33 <-
data.frame(gene.nam=rownames(gene.lasso.imp.stg3), gene.lasso.imp.stg3, row.names=NULL)

###For Stage IV

gene.lasso.imp.stg4 <- data.frame (gene.lasso.imp$gene.names, gene.lasso.imp$stage4.coeff)

colnames(gene.lasso.imp.stg4) <- c("gene.names", "stage4.coefficients")

gene.lasso.imp.stg444 <-gene.lasso.imp.stg44 <-
data.frame(gene.nam=rownames(gene.lasso.imp.stg4), gene.lasso.imp.stg4, row.names=NULL)


### Relevel the gene.names by importance

gene.lasso.imp.stg111$gene.names <-factor(gene.lasso.imp.stg111$gene.names,
levels=gene.lasso.imp.stg111[order(gene.lasso.imp.stg11$stage1.coefficients), "gene.names"])

gene.lasso.imp.stg222$gene.names <-factor(gene.lasso.imp.stg222$gene.names,
levels=gene.lasso.imp.stg222[order(gene.lasso.imp.stg22$stage2.coefficients), "gene.names"])

gene.lasso.imp.stg333$gene.names <-factor(gene.lasso.imp.stg333$gene.names,
levels=gene.lasso.imp.stg333[order(gene.lasso.imp.stg33$stage3.coefficients), "gene.names"])

gene.lasso.imp.stg444$gene.names <-factor(gene.lasso.imp.stg444$gene.names,
levels=gene.lasso.imp.stg444[order(gene.lasso.imp.stg44$stage4.coefficients), "gene.names"])
```

```
### Variable Importance Plots for stages I, II, III & IV

w <-ggplot(gene.lasso.imp.stg111, aes(y=gene.names, x=stage1.coefficients)) +
geom_point(stat="identity") + ggtitle("Stage I Training Approach Line Graph for the \nLasso
Variable Importance")

x <-ggplot(gene.lasso.imp.stg222, aes(y = gene.names, x=stage2.coefficients)) +
geom_point(stat="identity") + ggtitle("Stage II Training Approach Line Graph for the \nLasso
Variable Importance")

grid.arrange(w,x, ncol=2)

y <-ggplot(gene.lasso.imp.stg333, aes(y=gene.names, x=stage3.coefficients)) +
geom_point(stat="identity") + ggtitle("Stage III Training Approach Line Graph for the \nLasso
Variable Importance")

z <-ggplot(gene.lasso.imp.stg444, aes(y = gene.names, x=stage4.coefficients)) +
geom_point(stat="identity") + ggtitle("Stage IV Training Approach Line Graph for the \nLasso
Variable Importance")

grid.arrange(y,z, ncol=2)


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

###LOOCV LASSO

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

pred.lasso.100<-c()

preds100<-c()

correct.class = 0

miss.class = 0

training.final.100<-final_data100

rownames(training.final.100) <- training.final.100[,1]

training.final.100$X<- NULL


for(i in 1:nrow(training.final.100)) {

  loocv.predictors100 = training.final.100[i, ]

  lasso.predictors100 = training.final.100[-c(i),]

  fit_lasso.100=glmnet(as.matrix(lasso.predictors100[,-100]), lasso.predictors100$stage,alpha=1,
family="multinomial")
```

```
  lasso.pred100 <- predict(fit_lasso.100, newx=as.matrix(loocv.predictors100[,-100]) ,
s=min(fit_lasso.100$lambda), type = "class")
  preds100[i] = lasso.pred100
  if (lasso.pred100 == loocv.predictors100$stage ) {
    correct.class = correct.class + 1
  } else {
    miss.class = miss.class + 1
  }
  print(i)
  cat("miss.class ", miss.class, " correct.class ", correct.class)
  pred.lasso.100<-c(pred.lasso.100, lasso.pred100)
}
cat('\n\ncorrect.class:', correct.class, ' Accuracy:', correct.class/(correct.class + miss.class), '\n')
cat('miss.class:', miss.class, ' Miss.class Rate:', miss.class/(correct.class + miss.class), '\n')
write.csv(data.frame(correct.class, miss.class))
loocv.lasso.100 <- confusionMatrix(pred.lasso.100,training.final.100$stage)

### KENDALL'S CORRELATION COEFFICIENT: LASSO REGRESSION
###LOOCV
LASSO.LOOCV_kendall.100 <- t(loocv.lasso.100$table)
Desc(LASSO.LOOCV_kendall.100, verbose="high")
###TRAINING
LASSO.TRAINING_kendall.100 <- t(training.lasso.100$table)
Desc(LASSO.TRAINING_kendall.100, verbose="high")
```

```
###ESTIMATING MAXIMUM TOTAL MISCLASSIFICATION COST FOR THE LASSO
###LOOCV

###Optimization using the constrOptim (Lange, 2001) function

cost_matrix <- matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),
nrow=4,  dimnames=list("true"=c("1","2","2","4"), "predicted"=c("1","2","3","4")))

MAXTC.LASSO.LOOCV4171 <-c()

for (i in 1:4){

  fQP.LASSO.LOOCV417 <- function(b) {-sum(c(0,5,0,0)*b)+0.5*sum(b*b)}

  Amat.LASSO.LOOCV417 <- cost_matrix

  bvec.LASSO.LOOCV417 <- t(loocv.lasso.100$table)

  MAXTC.LASSO.LOOCV417 <- constrOptim(c(15,15,15,15), fQP.LASSO.LOOCV417,
NULL, ui = t(Amat.LASSO.LOOCV417), ci = bvec.LASSO.LOOCV417[,i])

  MAXTC.LASSO.LOOCV4171[i] <- MAXTC.LASSO.LOOCV417$barrier.value

}

###derivative

MAXTC.LASSO.LOOCV41V <-c()

for (i in 1:4){

  gQP.LASSO.LOOCV417 <- function(b) {-c(0,5,0,0) + b}

  MAXTC.LASSO.LOOCV41 <-constrOptim(c(15,15,15,15), fQP.LASSO.LOOCV417,
gQP.LASSO.LOOCV417, ui = t(Amat.LASSO.LOOCV417), ci = bvec.LASSO.LOOCV417[,i])

  MAXTC.LASSO.LOOCV41V[i] <- MAXTC.LASSO.LOOCV41$barrier.value

}

### Now with maximisation instead of minimisation

MAXTC.LASSO.LOOCV4 <-c()

for (i in 1:4){

  hQP.LASSO.LOOCV417 <- function(b) {sum(c(0,5,0,0)*b)-0.5*sum(b*b)}

  max_TC.LOOCV417 <- constrOptim(c(15,15,15,15), hQP.LASSO.LOOCV417, NULL, ui =
t(Amat.LASSO.LOOCV417), ci = bvec.LASSO.LOOCV417[,i], control = list(fnscale = -1))

  MAXTC.LASSO.LOOCV4[i] <- max_TC.LOOCV417$barrier.value

}

MAXTC.LASSO.LOOCV4171 ### minimisation
```

```
MAXTC.LASSO.LOOCV41V   ### derivative

MAXTC.LASSO.LOOCV4    ### Now with maximisation instead of minimisation

maxTC_LASSO_LOOCV100 <- max(MAXTC.LASSO.LOOCV4)

maxTC_LASSO_LOOCV100

distance_LASSO.LOOCV100= sqrt((1-loocv.lasso.100$overall[1])^2 +
(maxTC_LASSO_LOOCV100)^2)

distance_LASSO.LOOCV100


###ESTIMATING MAXIMUM TOTAL MISCLASSIFICATION COST FOR THE LASSO
###TRAINING APPROACH #

###Optimization using the constrOptim (Lange, 2001) function

MAXTC.LASSO.4171 <-c()

for (i in 1:4){

  fQP.LASSO.417 <- function(b) {-sum(c(0,5,0,0)*b)+0.5*sum(b*b)}

  Amat.LASSO.417 <- cost_matrix
###matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),4,4)

  bvec.LASSO.417 <- t(training.lasso.100$table)

  MAXTC.LASSO.417 <- constrOptim(c(15,15,15,15), fQP.LASSO.417, NULL, ui =
t(Amat.LASSO.417), ci = bvec.LASSO.417[,i])

  MAXTC.LASSO.4171[i] <- MAXTC.LASSO.417$barrier.value

}

### derivative

MAXTC.LASSO.41W <-c()

for (i in 1:4){

  gQP.LASSO.417 <- function(b) {-c(0,5,0,0) + b}

  MAXTC.LASSO.41 <-constrOptim(c(15,15,15,15), fQP.LASSO.417, gQP.LASSO.417, ui =
t(Amat.LASSO.417), ci = bvec.LASSO.417[,i])

  MAXTC.LASSO.41W[i] <- MAXTC.LASSO.41$barrier.value

}
```

```
### Now with maximisation instead of minimisation

MAXTC.LASSO.4 <-c()

for (i in 1:4){

  hQP.LASSO.417 <- function(b) {sum(c(0,5,0,0)*b)-0.5*sum(b*b)}

  max_TC.LASSO.417 <- constrOptim(c(15,15,15,15), hQP.LASSO.417, NULL, ui =
t(Amat.LASSO.417), ci = bvec.LASSO.417[,i], control = list(fnscale = -1))

  MAXTC.LASSO.4[i] <- max_TC.LASSO.417$barrier.value

}

MAXTC.LASSO.4171 ###minimisation

MAXTC.LASSO.41W   ### derivative

MAXTC.LASSO.4    ### Now with maximisation instead of minimisation

maxTC_LASSO_training.final.100 <- max(MAXTC.LASSO.4)

maxTC_LASSO_training.final.100

distance_LASSO.training.final.100= sqrt((1-training.lasso.100$overall[1])^2 +
(maxTC_LASSO_training.final.100)^2)

distance_LASSO.training.final.100
```

APPENDIX G

R CODE FOR MULTINOMIAL LOGISTIC REGRESSION (MLR) METHOD

```
###LOGISTIC REGRESSION FOR CLASSIFICATION FOR TOP 99 GENES
###LOOCV APPROACH
library(nnet)
training.final.100<-final_data100
rownames(training.final.100) <- training.final.100[,1]
training.final.100$X<- NULL
predLR100 <- c()
pred.probs.lr100 <- c()
cor1 = 0; mis1 = 0;
for(i in 1:nrow(training.final.100)) {
  #training.final.100$stage1 <- relevel(factor(patient.predictors100$stage), ref = "1")
  loocv.predictors100 = training.final.100[i, ]
  patient.predictors100 = training.final.100[-c(i),]
  mlogit.100 <- multinom(factor(patient.predictors100$stage) ~., data = patient.predictors100)
  ### Getting predictions
  pred.probs100 <- predict(mlogit.100, newdata = loocv.predictors100[,-100], "class")
  pred.probs.lr100[i] <- pred.probs100
  if ( pred.probs100  == loocv.predictors100$stage ) {
    cor1 = cor1 + 1
  } else {
    mis1 = mis1 + 1
  }
  print(i)
  cat("misclass1 ", mis1, " correct1 ", cor1)
  pred.LR.100<-c(predLR100, pred.probs.lr100)
}
cat('\n\ncor1:', cor1, ' Accuracy:', cor1/(cor1 + mis1), '\n')
cat('mis1:', mis1, ' Mis1 Rate:', mis1/(cor1 + mis1), '\n')
write.csv(data.frame(cor1, mis1))
LOOCV.LOGIT.100 <- confusionMatrix(pred.LR.100,training.final.100$stage)
```

```
### TRAINING APPROACH
library(nnet)
training.final.100<-final_data100
rownames(training.final.100) <- training.final.100[,1]
training.final.100$X<- NULL
training.final.100$stage <- relevel(factor(training.final.100$stage), ref = "1")
mlogit100 <- multinom(stage ~., data = training.final.100)
z100 <- summary(mlogit100)$coefficients/summary(mlogit100)$standard.errors
###2-tailed z test
p100 <- (1 - pnorm(abs(z100), 0, 1))*2
###extract the coefficients from the model and exponentiate
exp.logit100 <- exp(coef(mlogit100))

### Getting predicted classes
pred.prob.100 <- predict(mlogit100, newdata = training.final.100[,-100], "class")
pred.probs.fitted100 <- fitted(mlogit100)#fitted function gives predicted probabilities too
TRAINING.LOGIT.100 <- confusionMatrix(pred.prob.100, training.final.100$stage)

### Variable Importance
importz = varImp(mlogit100)
gene.names=substring(rownames(importz), 2)
Overall_importance = importz[1]
gene.logit.imp =data.frame(cbind(gene.names, Overall_importance))
colnames(gene.logit.imp) <- c("gene.names", "Overall_importance")
genes.logit.coef_range <- range(0, gene.logit.imp$Overall_importance)
library(ggplot2); library(gridExtra
gene.logit.imp1 <-gene.logit.imp <-data.frame(gene.nam=rownames(gene.logit.imp),
gene.logit.imp, row.names=NULL)
```

```
## Relevel the gene.names by Overall_importance

gene.logit.imp.ordered$gene.names <-factor(gene.logit.imp1$gene.names,
levels=gene.logit.imp1[order(gene.logit.imp$Overall_importance), "gene.names"])

x <-ggplot(gene.logit.imp.ordered, aes(y=gene.names, x=Overall_importance)) +
geom_point(stat="identity") + ggtitle("Training Approach Line Graph for the \nLogistic
Regression Variable Importance")

y <-ggplot(gene.logit.imp.ordered, aes(x = gene.names, y=Overall_importance)) +
geom_bar(stat="identity") + coord_flip()+ggtitle("Training Approach Bar Graph for the
\nLogistic Regression Variable Importance")

grid.arrange(x, y, ncol=2)


### KENDALL'S CORRELATION COEFFICIENT: LOGIT REGRESSION

###LOOCV

LOGIT.LOOCV_kendall.100 <- t(LOOCV.LOGIT.100$table)

Desc(LOGIT.LOOCV_kendall.100, verbose="high")

###TRAINING

LOGIT.TRAINING_kendall.100 <- t(TRAINING.LOGIT.100$table)

Desc(LOGIT.TRAINING_kendall.100, verbose="high")


###ESTIMATING MAXIMUM TOTAL MISCLASSIFICATION COST FOR THE LASSO
###LOOCV

###Optimization using the constrOptim (Lange, 2001) function

cost_matrix <- matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),
nrow=4, dimnames=list("true"=c("1","2","2","4"), "predicted"=c("1","2","3","4")))


MAXTC.LOGIT.LOOCV4171 <-c()

for (i in 1:4){

  fQP.LOGIT.LOOCV417 <- function(b) {-sum(c(0,5,0,0)*b)+0.5*sum(b*b)}

  Amat.LOGIT.LOOCV417 <- cost_matrix

  bvec.LOGIT.LOOCV417 <- t(LOOCV.LOGIT.100$table)

  MAXTC.LOGIT.LOOCV417 <- constrOptim(c(15,15,15,15), fQP.LOGIT.LOOCV417,
NULL, ui = t(Amat.LOGIT.LOOCV417), ci = bvec.LOGIT.LOOCV417[,i])

  MAXTC.LOGIT.LOOCV4171[i] <- MAXTC.LOGIT.LOOCV417$barrier.value
```

```
}

### derivative

MAXTC.LOGIT.LOOCV41V <-c()

for (i in 1:4){

  gQP.LOGIT.LOOCV417 <- function(b) {-c(0,5,0,0) + b}

  MAXTC.LOGIT.LOOCV41 <-constrOptim(c(15,15,15,15), fQP.LOGIT.LOOCV417,
gQP.LOGIT.LOOCV417, ui = t(Amat.LOGIT.LOOCV417), ci = bvec.LOGIT.LOOCV417[,i])

  MAXTC.LOGIT.LOOCV41V[i] <- MAXTC.LOGIT.LOOCV41$barrier.value

}

### Now with maximisation instead of minimisation

MAXTC.LOGIT.LOOCV4 <-c()

for (i in 1:4){

  hQP.LOGIT.LOOCV417 <- function(b) {sum(c(0,5,0,0)*b)-0.5*sum(b*b)}

  LOGIT.max_TC.LOOCV417 <- constrOptim(c(15,15,15,15), hQP.LOGIT.LOOCV417,
NULL, ui = t(Amat.LOGIT.LOOCV417), ci = bvec.LOGIT.LOOCV417[,i],

                                control = list(fnscale = -1))

  MAXTC.LOGIT.LOOCV4[i] <- LOGIT.max_TC.LOOCV417$barrier.value

}

MAXTC.LOGIT.LOOCV4171 ### minimisation

MAXTC.LOGIT.LOOCV41V   ### derivative

MAXTC.LOGIT.LOOCV4    ### Now with maximisation instead of minimisation

maxTC_LOGIT_LOOCV100 <- max(MAXTC.LOGIT.LOOCV4)

maxTC_LOGIT_LOOCV100

distance_LOGIT.LOOCV100= sqrt((1-LOOCV.LOGIT.100$overall[1])^2 +
(maxTC_LOGIT_LOOCV100)^2)

distance_LOGIT.LOOCV100
```

```
###ESTIMATING MAXIMUM TOTAL MISCLASSIFICATION COST FOR THE LOGIT
###TRAINING APPROACH #

###Optimization using the constrOptim (Lange, 2001) function

MAXTC.LOGIT.4171 <-c()

for (i in 1:4){

  fQP.LOGIT.417 <- function(b) {-sum(c(0,5,0,0)*b)+0.5*sum(b*b)}

  Amat.LOGIT.417 <-
matrix(c(0,2.68,5.37,11.78,5,0,2.11,6.15,10,2.11,0,3.08,17.25,4.84,2.42,0),4,4)

  bvec.LOGIT.417 <- t(TRAINING.LOGIT.100$table)

  MAXTC.LOGIT.417 <- constrOptim(c(15,15,15,15), fQP.LOGIT.417, NULL, ui =
t(Amat.LOGIT.417), ci = bvec.LOGIT.417[,i])

  MAXTC.LOGIT.4171[i] <- MAXTC.LOGIT.417$barrier.value

}

### derivative

MAXTC.LOGIT.41W <-c()

for (i in 1:4){

  gQP.LOGIT.417 <- function(b) {-c(0,5,0,0) + b}

  MAXTC.LOGIT.41 <-constrOptim(c(15,15,15,15), fQP.LOGIT.417, gQP.LOGIT.417,
ui = t(Amat.LOGIT.417), ci = bvec.LOGIT.417[,i])

  MAXTC.LOGIT.41W[i] <- MAXTC.LOGIT.41$barrier.value

}

###Now with maximisation instead of minimisation

MAXTC.LOGIT.4 <-c()

for (i in 1:4){

  hQP.LOGIT.417 <- function(b) {sum(c(0,5,0,0)*b)-0.5*sum(b*b)}

  max_TC.LOGIT.417 <- constrOptim(c(15,15,15,15), hQP.LOGIT.417, NULL, ui =
t(Amat.LOGIT.417), ci = bvec.LOGIT.417[,i], control = list(fnscale = -1))

  MAXTC.LOGIT.4[i] <- max_TC.LOGIT.417$barrier.value

}

MAXTC.LOGIT.4171 ### minimisation

MAXTC.LOGIT.41W   ### derivative
```

```
MAXTC.LOGIT.4    ### Now with maximisation instead of minimisation

maxTC_LOGIT_training100 <- max(MAXTC.LASSO.4)

maxTC_LOGIT_training100

distance_LOGIT.TRAINING100= sqrt((1-TRAINING.LOGIT.100$overall[1])^2 +
(maxTC_LOGIT_training100)^2)

distance_LOGIT.TRAINING100
```

REFERENCES

REFERENCES

Agresti, Alan. 2002. *Categorical Data Analysis.* Hoboken, NJ: John Wiley & Sons.

Ahn, Hongshik, and Hojin Moon. 2010. "Classification: Supervised Learning with High-Dimensional Biological Data." In *Statistical Bioinformatics for Biomedical and Life Science Researchers*, edited by Jae K. Lee, 129-156. Hoboken, NJ: Wiley-Blackwell.

Angulo, Cecilio, Parra Xavier, and Andreu Catala. 2003. "K-SVCR. A Support Vector Machine for Multi-class Classification." *Neurocomputing* 55, no. 1-2 (September): 57–77

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45: 5-32.

Breiman, Leo, Friedman Jerome, Stone Charles and Richard Olshen. 1984. *Classification and Regression Trees*. Boca Raton, FL: Chapman & Hall/CRC Press LLC.

Carillo, Henry, Kay H. Brodersen, and Jose A. Castellanos. 2014. "Probabilistic Performance Evaluation for Multiclass Classification Using the Posterior Balanced Accuracy." In *ROBOT2013: First Iberian Conference-Advances in Robotics* 1 (November), edited by Manuel A. Armada**,** Alberto Sanfeliu, Manuel Ferre, 347-361. Madrid, Spain: Springer.

Chang, Ching-Wei, George Nysia, and Lu Tzu-Pin. 2016. "Cost-Sensitive Performance Metric for Comparing Multiple Ordinal Classifiers." *Journal of Artificial Intelligence Research* 5, no. 1 (January). Accessed February 10, 2016. http:/dx.doi.org/10.5430/air.v5n1p135.

Cutler, Richard D., Thomas C. Edwards, Karen H. Beard, Adele Cutler, Kyle T. Hess, Jacob C. Gibson, and Joshua J. Lawler. 2007. "Random Forests for Classification in Ecology." *Ecology* 88 (11): 2783-2792.

Czepiel, Scott. 2002. "Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation." Scott Czepiel. Accessed March 20, 2016. http://czep.net/

Dziuda, Darius. 2010. *Data Mining for Genomics and Proteomics: Analysis of Gene and Protein Expression Data.* Hoboken, NJ: John Wiley & Sons.

Foulkes, Andrea S. 2009. *Applied Statistical Genetics with R: For Population-Based Association Studies.* New York: Springer.

Ghosh, Debashis, and Arul M. Chinnaiyan. 2005. "Classification and Selection of Biomarkers in Genomic Data Using LASSO." *Journal of Biomedicine and Biotechnology* 2:147-154.

Green, Hayden G., Betsy V. Boze, Askar H. Choundhury, and Simon Power. 1998. "Using Logistic Regression in Classification: The Marketing of Reclaimed Potentially Environmentally Damaged Residential Property". *Marketing Research* 31: 5-11.

Herbrich, Ralf, Thore Graepel, and Klaus Obermayer. 1999. *Regression Models for Ordinal Data: A machine learning approach, Technical report.* Berlin: Technical University of Berlin.

Hsu, Chih-Wei and Chih-Jen Lin. 2002. "A Comparison of Methods for Multiclass Support Vector Machines." *IEEE Trans Neural Networks* 13 (March):415–425.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning: with Applications in R.* New York: Springer.

Karatzoglou, Alexandros, David Meyer, and Kurt Hornik. 2006. "Support Vector Machines in R." *Journal of Statistical Software* 15 (April), no. 9.

Kotsiantis, Sotiris, and Panagiotis Pintelas. 2004. "A Cost Sensitive Technique for Ordinal Classification Problems." *Methods and Applications of Artificial Intelligence* 3025: 220-229.

Larocque, Denis, Hatem Ben-Ameur, and Imad Bou-Hamad. 2011. "A Review of Survival Trees." *Statistics Surveys* 5: 44-71.

Leha, Andreas, Klaus Jung, and Tim Beißbarth. 2013. "Utilization of Ordinal Response Structures in Classification with High-Dimensional Expression Data." In *German Conference on Bioinformatics 2013*, edited by Tim Beißbarth, M. Kollmar, Andreas Leha, B. Morgenstern, A. K. Schultz, S. Waack, and E. Wingender, 90-100. Göttingen Germany: Dagstuhl Publishing.

Li, Tao, Chegliang Zhang, and Mitsunori Ogihara. 2004. "A Comparative Study of Feature Selection and Multiclass Classification Methods for Tissue Classification Based on Gene Expression." *Bioinformatics* 20, no. 15 (March): 2429–2437.

Mukherjee, Siddhartha. 2010. *The Emperor of all Maladies: A Biography of Cancer.* New York: Scribner.

Ovidiu, Ivanciuc. 2007. "Applications of Support Vector Machines in Chemistry." In *Reviews in Computational Chemistry*, edited by Kenny Lipkowitz and Thomas Cundari, 291-400. Weinheim: Wiley-VCH.

Rameswar, Debnath, N. Takahide, and Haruhisa Takahashi. 2004. "A Decision-Based One-Against-One Method for Multi-class Support Vector Machine." *Pattern Analysis and Applications* 7 (July):164–175.

Rifkin, Ryan and Aldebaro Klautau. 2004. "In Defense of One-vs-All Classification." *Journal of Machine Learning Research* 5 (December):101–141.

Sanchez-Montero, Javier, Pedro A. Gutierrez, Francisco Fernandez-Navarro, and Hervas-Martinez CeSar. 2011. "Weighting Efficient Accuracy and Minimum Sensitivity for Evolving Multi-Class Classifiers." *Neural Processing Letters* 34, no. 2 (October): 101-116.

Schoelkopf, Bernhard and Alex Smola. 2002. "Support Vector Machines and Kernel Algorithms." Alex Smola. Accessed March 20, 2016. http://alex.smola.org/papers/2002/SchSmo02b.pdf

Scikit-learn developers. (2010-2014). http://scikit-learn.org/stable/modules/cross_validation.html Accessed 11-10-2015.

Signorell, Andri. 2015. *Tables in R – A Quick Practical Overview*. Zurich: Helsana Versicherungen AG, Health Sciences.

Smith, Joshua J., Natasha G. Deane, Fei Wu, Nipun B. Merchant, Bing Zhang, Aixiang Jiang, Pengcheng Lu, Chad J. Johnson, Carl Schmidt, Christina E. Bailey, Steven Eschrich, Christian Kis, Shawn Levy, Kay M. Washington, Martin J. Heslin, Robert J. Coffey, Timothy J. Yetaman, Yu Shyr, and Daniel Beauchamp. 2010. "Experimentally Derived Metastasis Gene Expression Profile Predicts Recurrence and Death in Patients with Colon Cancer." *Gastroenterology* 138 (3) (March): 958-968.

Statnikov, Alexander, Constantin F. Aliferis, Ioannis Tsamardinos, Douglas Hardin, and Levy Shawn. 2005. "A Comprehensive Evaluation of Multicategory Classification Methods for Microarray Gene Expression Cancer Diagnosis." *Bioinformatics* 21, no. 5 (September): 631–643.

Stewart, Benard W., and Paul Kleihues. 2003. *World Cancer Report*. Lyon:IARC Press.

Therneau, Terry, Beth Atkinson, and Brian Ripley. 2015. "Recursive Partitioning and Regression Trees." The Comprehensive R Archive Network, June 29. Accessed November 10, 2015. https://cran.r-project.org/web/packages/rpart/rpart.pdf .

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society.* Series B (Methodological) 58, no 1 (February): 267-288.

Trevor, Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. New York: Springer Science + Business Media, LLC.

Wackerly, Dennis, William Mendenhall III, and Richard L. Scheaffer. 2008. *Mathematical Statistics with Applications.* 7th ed. Belmont, California: Thomson Learning.

Xu, Ying, Juan Cui, and David Puett. 2014. *Cancer Bioinformatics*. NewYork: Springer.

Yamagiwa, Katsusaburo, and Ichikawa Koichi. 1918. “Experimental Study of the Pathogenesis of Carcinoma.” *The Journal of Cancer Research* 3, no. 1: 1-29.

Zhang, J. et al. 2012. “The Genetic Basis of Early T-Cell Precursor Acute Lymphoblastic Leukaemia.” *Nature* 481 (January) :157–163. doi:10.1038/nature10725.