2020

An Empirical Comparison of Machine Learning Models for Classification

Nubaira Rizvi

AN EMPIRICAL COMPARISON OF MACHINE LEARNING MODELS FOR CLASSIFICATION

by

Nubaira Rizvi

Bachelor of Science
University of Dhaka, 2017

Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Science in

Statistics

College of Arts and Sciences

University of South Carolina

2020

Accepted by:

David Hitchcock, Director of Thesis

Karl Gregory, Reader

Minsuk Shin, Reader

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

DEDICATION

Dedicated to my parents, Shakil and Rehana Rizvi, for their unconditional love and encouragement.

ACKNOWLEDGEMENTS

I would like to thank Dr. David Hitchcock for his guidance and support while completing this study. This thesis would not have been possible without his knowledge and endless patience.

I would like to thank Dr. Karl Gregory and Dr. Minsuk Shin for severing on my committee on such short notice.

I would also like to thank my wonderful professors especially Dr. Lianming Wang for being there when I needed them.

ABSTRACT

Classification problems are tackled across various industries throughout multiple disciplines. A model used for classification attempts to predict the class of an outcome variable based on some predictors. There are number of classification models available. But as the underlying population distribution of the predictors is always unknown it is difficult to know which model fits the situation best. Several studies have been done on which supervised model performs better given specific datasets. But little work has been done to compare the models' performance for predicting one or more outcomes under multivariate settings.

This study compares the performance of seven popular statistical learning models used for classification when the dataset is from a multivariate population. The models are: K-nearest neighbor, logistic regression, support vector machines, linear discriminant analysis, random forest, adaptive boosting and gradient boosting. We compare these methods under three different distribution settings, e.g., multivariate normal distribution, multivariate t distribution and multivariate log-normal distributions for both binary, k=2 and multiple outcomes, k=5. Three different sample sizes, n=100, 300, 500 are considered along with two different number of predictors, p=3,10 to check if the performance changes with sample size and number of predictors. We also compare the models for balanced and unbalanced datasets under these different settings. The models are evaluated using two criteria: accuracy, which works best for balanced datasets and Cohen's kappa coefficient, which gives good result under unbalanced datasets.

A 10-fold cross validation technique is used where the data is randomly split into 75% training set and 25 % testing set to test the models' prediction skills on new data. The model parameters are tuned under each setting to get the best performing model. Boxplots are used to show the spread of performance metrics calculated from multiple iterations. It is found that for the multivariate normal distribution, boosting algorithms are superior to others, whereas for the multivariate t distribution, support vector machines are preferable. Lastly, for the multivariate log-normal distribution, all models perform well but the random forest algorithm was better than the rest in most cases. The preference of models changes with an increase in the number of classes depending on balanced and unbalanced datasets. But overall, the gradient boosting algorithm and random forest have good performance accuracies for all the settings. This is further implemented on a real dataset of heart disease patients to verify the results of the simulation. Gradient boosting produces the highest prediction accuracy among all seven models and the K-nearest neighbor had the narrowest spread of accuracies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In recent years machine learning models have become one of the most appealing tools of choice for data analysis across different disciplines. Most machine learning models fall into three main categories: supervised learning, unsupervised learning and reinforcement learning. Among them, supervised learning is used in situations where each observation has a response measurement. Depending on whether the response is categorical or continuous we can further group supervised learning models into regression or classification models. This thesis presents a comparison study for some popular machine learning models used for classification.

1.1 K- Nearest Neighbors

The K-nearest neighbor rule (Fix & Hodges, 1951) is the simplest method among all techniques. It is easy to interpret and has good predictive powers. A set of training data with n-dimensional attributes are represented in n-dimensional pattern space. A point in the space is a sample. The unknown sample is classified when the K-nearest neighbor classifier finds a pattern in the space for K training samples that are closest to the unknown sample. They are instance-based or lazy learners since they store all the training data first and build a classifier only when a new or unlabelled sample needs to be classified. The closeness is defined by distance measures such as Euclidian distance, Manhattan distance, Chebyshev distance, etc for continuous variables and Hamming distance for categorical variables. The most popular distance used is the Euclidian distance. Standardization of numerical variables is necessary if there is both numerical and categorical variables present in the dataset.

The Euclidian distance between two points, X= (x_1, x_2, …,x_n) and Y= (y_1, y_2, …,y_n) is

$$D(X,Y) = \sqrt{\sum_{i}^{n}(X_i - Y_i)^2}$$

The key parameter in this method is naturally K. Careful consideration is required to select the value of K. It is selected such that validation error is minimum while maintaining a low training error rate. As the value of K decreases the prediction becomes less stable whereas a larger value of K will give a smoother fit.

1.2 Logistic Regression Classifier

Logistic regression is a popular machine learning model used to determine the probability of an observation being in a certain class. The closer the probability is to 1 the more likely it is the value is part of that class. The probability is calculated using the sigmoid function:

$$P(x) = \frac{e^{(\beta_o+\beta_1 X)}}{1 + e^{(\beta_o+\beta_1 X)}}$$

We can get the logit equation from this after some mathematical manipulation:

$$\log\left(\frac{p(x)}{1-p(x)}\right) = (\beta_o + \beta_1 X)$$

Since the logit equation is linear, for positive coefficients, a higher value of X results in higher probabilities. The coefficients can be estimated using the maximum likelihood method of estimation.

1.3 Linear Discriminant Analysis

Linear Discriminant Analysis is a model-based classifier that uses statistical distances as the basis. It projects the data onto a new axis in a way to maximize the separation of the two categories when we have predefined response classes. The goal is to classify objects into different types of pattern recognition. It is more interpretable

and easier to predict than other classification algorithms. In this technique, each observation is treated as a point in a N-dimensional space, where N is the number of predictor variables. The points are labelled by classes. LAD divides the dimensional space into parts based on categories and has linear boundaries. It treats all points in the same part as one category. This algorithm assumes that the dependent variables have the same multivariate normal distribution where different classes have their class-specific means and equal covariance. Mathematically, LAD uses input data to calculate group means and the probability of belonging to various groups. Then scoring function coefficients are computed for each category. The function is:

$$\widehat{\delta_k}(X) = X \,.\frac{\widehat{\mu_k}}{\sigma^2} - \frac{\widehat{\mu_k}}{2\sigma^2} + \log\left(\widehat{\pi_k}\right),$$

where, $\widehat{\delta_k}(X)$ is the discriminant function. $\widehat{\pi_k} = \frac{n_k}{n}$ is the proportion of training observations that are in category K, $\widehat{\mu_k}$ is the sample mean of class K and σ^2 is the weighted average of the sample variance for each class. The score calculated from each function uses the explanatory variables to get class-specific coefficients and a case is labelled with the class for which the score is highest.

1.4 Support Vector Machines

Support vector machines is a discriminative classification technique that uses a hyperplane in multidimensional space to categorize objects. The hyperplane is a flat surface of (p-1) dimension in a p dimensional space. It splits the data in a way that the distance between support vectors or training points and the plane is farthest. The minimal distance from training observations to hyperplane is called the margins. This is a constrained optimization problem that tries to maximize this margin while maintaining that none of the points are on the plane. It can be solved using the Lagrange multiplier. The maximal margin classifier is the simplest form of classification when a perfectly separating hyperplane exists, and it is linear. For non-separable cases, soft

margins are used that almost separates the classes. This case is called the method of support vector classifiers or soft margin classifiers. Rather than finding the largest margin, it incorrectly classifies some points to get greater robustness. Sometimes the points can not only be on the wrong side of the margin but also on the wrong side of the hyperplane. In that case, soft margin classifiers misclassify the points. A tuning parameter, C, accounts for this misclassification, representing the amount of violation that will be tolerated. Thus, it controls the bias-variance trade-off since smaller C means low bias but high variance and vice versa. If C=0 then no violation will be allowed in the model. C > 0 means only C points will be allowed to be misclassified. C is generally chosen through cross-validation. The support vector machine is the generalization of the maximal margin classifier. Both regression and classification analyses are available with it and it can handle multiple qualitative and quantitative variables. It can be used when there is a non-linear hyperplane in the space by enlarging the feature space using quadratic, cubic and higher order polynomial functions of the predictors or kernels. A kernel is a function that quantifies the similarity between two observations. The different types of kernel functions are:

$$K(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C)^d & \text{Polynomial} \\ \exp(-\gamma |\mathbf{X}_i - \mathbf{X}_j|^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C) & \text{Sigmoid} \end{cases}$$

Here, $K(\mathbf{X}_i, \mathbf{X}_j) = \phi(\mathbf{X}_i) \bullet \phi(\mathbf{X}_j)$, that is the kernel function, represents a dot product of input data points mapped into the higher dimensional feature space by the transformation. The radial basis function (RBF) is the most popular kernel used in support vector machines. Gamma is the kernel coefficient when using RBF, polynomial or Sigmoid as the function. A larger value of gamma will lead to overfitting of data since it will then try to exactly fit the training set. A higher value of gamma takes into account only the closest points while ignoring the points far from the decision boundary. More weight is

given to the closest points and this results in more wiggly line. Cross validation is used to find an optimum combination of gamma and the tuning parameters.

1.5 Random Forest Algorithm

The random forest is a bagging ensemble method that uses the decision trees by decorrelating the decision trees. This is a robust and powerful model. Instead of selecting all the predictors, it chooses m features out of p features each time a split is considered. This increases predictive accuracy without substantially increasing bias error. Since the random forest is a collection of trees, features that were not included in one tree are present in others. An aggregated result of all trees is presented in the end. The predictors are chosen randomly, otherwise the same one might get chosen as the base for many trees. In this case error due to variance will still be high. The basics of a random forest are:

1) Choosing the number of trees in the forest (M)
2) Choosing the number of samples in each tree (n)
3) Choosing the number of features in each tree (f)
4) For each tree in M:
    a) Select n samples with replacement from all observations
    b) Select f features at random
    c) Train a decision tree using the data set of n samples with f features
    d) Save the decision tree

The model does not overfit if M is large. So, a sufficiently large value of M is taken to reduce the error rate.

1.6 Adaptive Boosting Algorithm

The adaptive boosting (Freund & Schapire, 1999) is a boosting algorithm that uses a set of weak classifiers and aggregates them to make a stronger prediction. The

weaker models are generated sequentially. The latest model learns from the mistakes of the previous learners. The dependency between the models is used to give the mislabeled observation higher weight. The weak learners for this algorithm can be any of the basic classifiers like logistics regression to decision trees. In this study decision stumps are used as the weak learners. Decision stumps are different from random forest as they are not fully developed. They have only one node and two leaves. The steps for algorithm are

1. Initially a weak classifier is trained by giving equal weights to all the sample that needs to be classified correctly.
2. Fit the weak classifier for each predictor and check how accurately each one classifies the sample. The classifier with the lowest weight error, α is selected.
3. More weights are given to the incorrectly classified stumps so that they have a higher chance of being correctly labelled in the next decision stump.
4. Repeat from step 2 until all the observation have been correctly labelled or up to maximum number of iterations.

The initial weight is, $w = \frac{1}{n}$ and the updated weight is, $w_i = w_{i-1} * e^{\pm\alpha}$. Here, n is the sample size and alpha, α is calculated as

$$\alpha = \frac{1}{2}\ln(\frac{1 - Total\ error}{Total\ error})$$

The total error is the number of misclassifications divided by number of training points. After m classifications, the final prediction is calculated as the sum of the weighted prediction of each classifier. It can be expressed as-

$$F(X) = sign(\sum_{m=1}^{M} w_m\, f_m(x))$$

Here, $f_m(x)$ is the m-th weak classifier and w_m is their respective weights. The sum of weights is always 1. That means individual weights will always be between 0 to 1.

1.7 Gradient Boosting

The gradient boosting method is a generalization of the adaptive boosting. In this case, a new weak learner is added at each step to improve the performance whereas the older ones are left unchanged. It has three main parts:

i. A differentiable loss function that has to be optimized such as logarithmic loss for classification

ii. The weak learner of choice such as decision trees. Purity scores like Gini is used to choose the best split point for the tree.

iii. An additive model that sums up the weak learners. In the existing sequence of trees, the output of a new tree is added each time to improve the model.

Bauer and Kohavi (1999) conducted an empirical study to compare the voting classification models like bagging, boosting, decision trees and naive Bayes algorithm. The goal was to check how these models influences the classification error terms. The study shows that voting methods compared to non-voting methods lead to a significant reduction in classification error.

Caruana and Niculescu-Mizil (2006) did a large-scale empirical evaluation of ten different supervised learning algorithms. The models used are SVM, neural networks, naïve Bayes, logistic regression, memory-based learning, random forest, decision trees, bagged trees, boosted trees and boosted stumps. These were performed on eleven binary classification problems. The best fitted models were chosen using a variety of evaluation metrics like ROC area, F-score, squared error etc. The space of parameters was explored extensively for each algorithm. The specialty of this study was the different performance criteria used. The study concludes that calibrated boosted trees performed best while models like naive Bayes, logistic regression and decision trees showed the poorest performance.

Narassiguin and Bibimoune (2016) conducted an empirical comparison of nineteen supervised ensemble methods, e.g., random forests, boosting, bagging etc. along with recent methods like random patches for binary classification. Nineteen UCI datasets were used to compare the models with probability metrics, threshold metrics and ranking metrics like area under the ROC curve. The study concludes that rotation forest family of algorisms performs better than all other ensemble models by a noticeable margin.

Armitage (2010) did a comparison of supervised learnings on bat echo-location calls using random forest, discriminant analysis, support vector machines and neural networks. Overall, all the models worked quite well except for discriminant analysis for which accuracy was quite low.

Ahmed (2010) fitted K-nearest neighbor, regression trees, support vector machine and other methods for time series data. This empirical study showed a significant difference in the performances of the methods. It also evaluates the performance of different pre-processing settings.

Shao (2012) used the support vector machine, neural network, random forest for comparison for the land-cover classification. He found that overall, support vector machine produces higher accuracies. There was also less variability for its accuracies compared to other models.

Brown and Muse (2011) compared traditional classification methods like-logistic regression, decision trees and neural networks with support vector machines, random forest and gradient boosting for imbalanced datasets. Five real world credit sorting datasets were used where the class imbalance was gradually increased to check how well the models perform. The metric used for model evaluation was the area under

the receiver operating characteristic curve (AUC). Friedman's Statistic and Nemenyi post hoc test were used to test if the AUC varies significantly.

CHAPTER 2

SIMULATION STUDY

2.1 Introduction

The goal of the simulation study is to see how different machine learning models perform in certain multivariate settings. In real life, the dataset's underlying distribution is unknown along with its parameters. Simulating data from a known distribution gives us the ability to judge their performance by comparing their predictive results with the real values. In this study, we generate data from three different types of distribution: the multivariate normal, multivariate t-distribution and multivariate log-normal distributions. The models' performances were also evaluated for increasing sample sizes, e.g., 100, 300, 500 where the first two sample sizes were used to check the accuracy with a balanced dataset and the last one was used to check the performance with a unbalanced dataset for both binary and multiple class outcome. The validation of the evaluation was checked using 10-fold cross validation method.

2.2 Methodology

The data were generated from three different distributions, multivariate normal, multivariate T and multivariate log-normal, where the mean and variances were chosen accordingly. Under each distribution, the groups had the same covariance matrix across the classes. The generated predictors were also correlated to each other. Along with the distributions, the settings of the simulation varied in sample size (n=100,300,500), number of groups (k=2,5) and number of predictors (p=3,10). The package *mvtnorm*

was used in R to generate all three types of datasets. The function *mvrnorm()* generated multivariate normal, *rmvt()* generated multivariate T and the exponential the exponential *exp()* of the multivariate normal data generated the log-normal distribution. Next, the data was split into two sets, where 75% were randomly assigned into the training set and 25% into the testing set. The training set was used to train the data and the testing set was used to make prediction and to calculate accuracies. The models were trained using the *train()* function from the package *caret* in R. Boxplots are used to show the accuracy with interquartile range as whiskers of the boxplots. Each model was fitted using 10-fold repeated cross-validation with iteration set to 10. An extensive range of parameter values were tried to find the model that gives the best fit for each algorithm.

The evaluation metrics used are- accuracy, which works well for a balanced dataset and Cohen's kappa coefficient (k) which gives good results for an unbalanced dataset. The accuracy is measured as the percentage of correctly classified observations out of all the observations. The formula for accuracy is:

$$\text{Accuracy} = \frac{\text{True positive}}{\text{True positive} + \text{True negetive} + \text{False positive} + \text{False negetive}}$$

The Cohen's kappa coefficient is a model performance metric that measures inter-rater reliability. The coefficient is calculated as:

$$\text{kappa} = \frac{\text{Total accuracy} - \text{Random accuracy}}{1 - \text{Random accuracy}}$$

The higher the value the better the models are considered.

For example, if we were to generate data from a multivariate normal distribution with three predictors, p=3 and where the dependent variable has two classes, with sample size, n=300, the first few observations of the simulated dataset would look like the one given below:

Table 2.1: Data generated from multivariate normal distribution with p=3

X1	X2	X3	Y
-3.48	3.89	6.09	1
1.27	2.39	5.91	1
-1.19	6.53	10.12	1

2.3 Results

All the simulated data and corresponding figures to show the spread of accuracy are provided in the appendix. Under the first setting where data was generated from the multivariate normal distribution, with predictors, p=3 and dependent variable, y with two categories, k=2, the boosting algorithms, adaptive boosting, random forest and gradient boosting perform better than others. Adaptive boosting had 71.4% accuracy on average, which is highest out of all the models. As we increase the sample size, the accuracy increases. For the unbalanced dataset, random forest performs best with 97% accuracy.

Table 2.2: Accuracy for multivariate normal when p=3 & k=2

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	0.57	0.50	0.625	0.714	0.571	0.571	0.571
n=300	0.95	0.91	0.90	0.95	0.82	0.81	0.95
n=500	0.97	0.84	0.84	0.973	0.67	0.68	0.94

The values for the kappa coefficient show similar results where the boosting algorithms, adaptive boosting, random forest and gradient boosting perform better than others. It worked especially well for sample size of n=500 where the dataset was unbalanced. Logistic regression and linear discriminant analysis perform worst.

Table 2.3: Kappa for multivariate normal when p=3 & k=2:

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	0.57	0.50	0.625	0.714	0.571	0.571	0.571
n=300	0.95	0.91	0.90	0.95	0.82	0.81	0.95
n=500	0.97	0.84	0.84	0.973	0.67	0.68	0.94

The boxplot shows the spread of the accuracy. Adaptive boosting had the narrowest spread and has the highest accuracy even when the sample size was small followed by gradient boosting and support vector machine. Linear discriminant analysis had the largest spread.

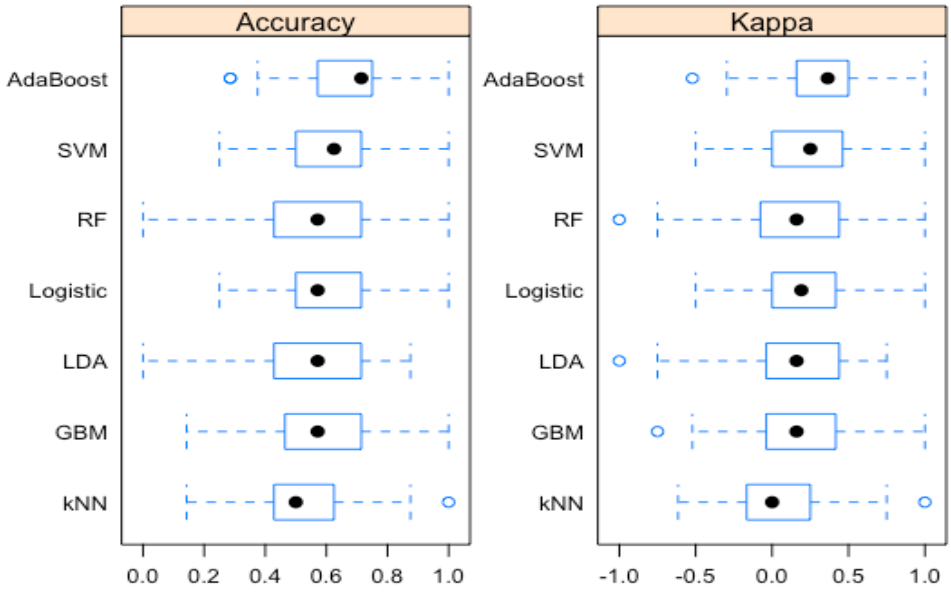

Figure 2.1:The boxplots for multivariate normal when n=100, p=3 & k=2

Under the setting where data was generated from the multivariate T distribution, with p=3 and dependent variable y had two categories, k=2, the support vector machine performs best across all the different sample sizes followed by the logistic regression and the gradient boosting algorithms. For the unbalanced dataset, it produced 65.4% accuracy and gradient boosting has 65.2% accuracy. The accuracy measures had larger spread as sample size decreases. Linear discriminant analysis had the narrowest spread for all sample sizes. Thus, it produced the most consistent result with the multivariate T distribution.

For the setting where data was generated from the multivariate log-normal distribution, with p=3 and dependent variable y had two categories, k=2, all the models perform equally well. The gradient boosting was the best of them all with 96%

accuracy. Logistic regression had the lowest accuracy with 63.7%. But K-nearest neighbor, logistic regression and linear discriminant analysis have the lowest spread of accuracy values. K-nearest neighbor produces the most consistent results on multiple iterations for all sample sizes. That is, using any one of them will give a highly accurate result when the data comes from a multivariate log-normal distribution.

For the next setting, where data was generated from the multivariate normal distribution, with ten predictor variables (p=10) and dependent variable having two classes (k=2), support vector machines had the highest accuracies for both balanced and unbalanced data. As sample size increased, accuracy also increased except for with the random forest model where it decreased slightly.

Table 2.4: Accuracy for multivariate normal when p=10 & k=2

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	1.00	0.988	1.00	0.985	1.00	0.987	0.986
n=300	0.998	0.998	1.00	0.992	1.00	1.00	0.995
n=500	0.997	0.9992	1.00	0.994	0.997	0.997	0.997

The kappa coefficient shows the similar results with the support vector machine performing best. All the other models had high accuracies as well since all of them have over 96% accuracy. The spread of accuracies for the models were extremely narrow, although they all had several outlier accuracies values.

Table 2.5: Kappa for multivariate normal when p=10 & k=2:

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	1.00	0.976	1.00	0.960	1.00	0.974	0.973
n=300	0.996	0.997	1.00	0.985	1.00	1.00	0.990
n=500	0.994	0.985	1.00	0.989	0.994	0.994	0.994

When data was generated from the multivariate T distribution, with ten predictor variables (p=10) and dependent variable having two classes (k=2), gradient boosting, adaptive boosting and random forest algorithm had 99.5% accuracy. Linear

discriminant analysis and logistic regression had the lowest accuracy with 57.5% and 65.8% respectively. The kappa coefficients showed similar results as gradient boosting and adaptive boosting both have 98.7%. But random forest had the highest value with 99%. These three models also had the narrowest spread of accuracy values.

For the setting where data was generated from the multivariate log-normal distribution, with ten predictor variables (p=10) and dependent variable having two classes (k=2), gradient boosting, adaptive boosting and random forest had the best results with 99.8%, 99.7% and 99.4% accuracies. Linear discriminant analysis had the worst accuracy with 60% when the sample size is small. But for larger samples, all the models performed well. From the boxplot, linear discriminant analysis and logistic regression had the largest spreads while others had relatively smaller ones.

Next, the data was generated from the distributions where the dependent variable had five classes (k=5). Since the number of classes are more than two, sample size 300 and 500 were considered and the dataset was balanced for both. Five models were compared instead of seven as binary logistic regression is designed for cases with two classes and adaptive boosting requires high computing power.

When data was generated from multivariate normal with three predictors (p=3) and five classes (k=5), all the models had high accuracies. As sample size increased, the accuracies dropped a small amount.

Table 2.6: Accuracy for multivariate normal when p=3 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	0.987	0.996	0.994	0.991	0.986
n=500	0.963	0.974	0.977	0.975	0.965

The kappa coefficient values increased as sample size increased, and all the models had high values. Support vector machines had the largest one. It also had the smallest spread along with K-nearest neighbors algorithm. For the rest of the models,

confidence intervals were relatively larger. The values for lower limit of the confidence interval were above 94%.

Table 2.7: Kappa for multivariate normal when p=3 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	0.954	0.968	0.971	0.968	0.957
n=500	0.984	0.995	0.993	0.989	0.983

For the setting where data was generated from multivariate T distribution with three predictors (p=3) and five classes (k=5), gradient boosting algorithm and random forest had the highest accuracies with 95.4% and 95.5% respectively. Linear discriminant analysis and support vector machines had the lowest values. They also had the largest spreads from the boxplot. The accuracies increased as sample size became larger in each group.

When data was generated from multivariate log-normal with three predictors (p=3) and five classes (k=5), the models had extremely high accuracies with narrow spreads. One reason for such high accuracy could be the class variance was set too low making it easier for the models to classify.

Lastly, the number of predictors were increased. For the setting where data was generated from a multivariate normal distribution with ten predictors (p=10) and five classes (k=5), linear discriminant analysis, random forest and gradient boosting had the best results with 86.6%, 84.2% and 83.9% accuracies. K-nearest neighbor had the lowest value at 66.9%. As the sample size increased, the accuracies increased.

Table 2.8: Accuracy for multivariate normal when p=10 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	0.842	0.669	0.787	0.866	0.839
n=500	0.886	0.696	0.863	0.915	0.887

The same results were obtained from the kappa coefficients. The spread was smallest for random forest and largest for K-nearest neighbors. They all had some outliers when sample size was 300 but as sample size increased there were no outliers.

Table 2.9: Kappa for multivariate normal when p=10 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	0.803	0.586	0.734	0.833	0.843
n=500	0.860	0.620	0.829	0.894	0.859

When data was generated from a multivariate T distribution with ten predictors (p=10) and five classes (k=5), random forest and gradient boosting had the highest values with 83.7% and 82.9%. The values increased with sample size but very slightly. Linear discriminant analysis performed worst with 22.8% accuracy. The same results were seen from the kappa coefficients. From the boxplot random forest and gradient boosting had the smallest spreads for both sample sizes.

Lastly, for data generated from multivariate log-normal distribution with ten predictors (p=10) and five classes (k=5), random forest and linear discriminant analysis had the highest accuracies with 80.7% and 81.6% respectively. K-nearest neighbor had the lowest value at 60.3%. Increasing the sample size raised the accuracies. The kappa coefficient had similar results with random forest resulting in the highest accuracy value.

Furthermore, to create a more difficult classification problem, a covariance matrix with higher values is used to check which log-normal model performs the best under each setting. In such case support vector machines and K-nearest neighbors had higher accuracy than most, while linear discriminant analysis had lower accuracies than others.

CHAPTER 3

COMPARING MODELS ON REAL DATA

3.1 Introduction to Dataset

In order to study the performance of the models in practice, a real dataset was examined. The dataset was donated by David W. Aha to the UCI repository and contains information about heart disease patients. It consists of 303 observations with 14 different attributes. The goal is to classify the patients in to 5 groups that denotes presence (1,2,3,4) and absence (0) of heart disease. The supervised models mentioned in this thesis were used to perform the classification. The first four observations of the dataset are given below:

Table 3.1: The heart disease dataset:

Age	Sex	CP	Trestbps	Chol	Fbs	Restecg	MaxHB	Exang	Oldpeak	Slope	Ca	Thal	Target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
56	1	1	120	236	0	1	178	0	0.8	2	0	2	1

3.2 Analysis

The dataset had to be pre-processed first before fitting the models. There were no missing values in the dataset. The numeric variables were scaled, and two mislabeled columns were re-labeled. The categorical variables were converted to factors in R. Then some exploratory analysis was done to check the distribution of the variables and the correlation matrix was created to see if there are any highly correlated variables to avoid multicollinearity.

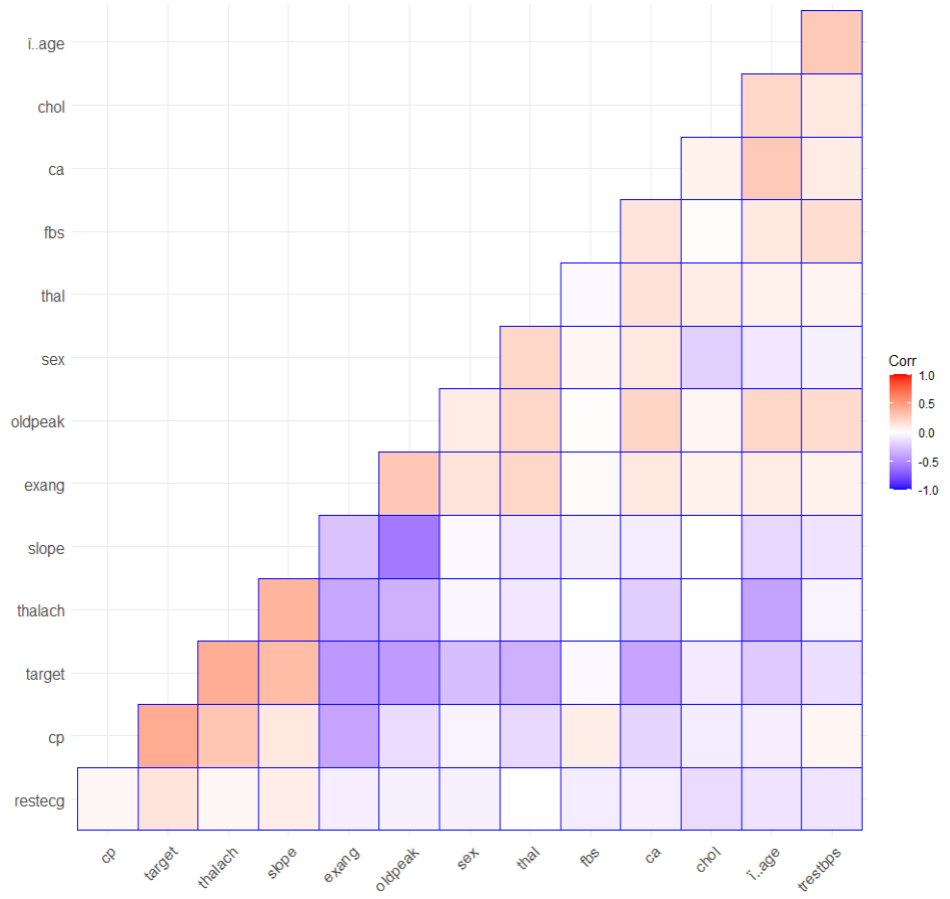

Figure 3.1 The correlation matrix for the dataset

Next, the dataset was then split into a training set (75%) and test set (25%). The models were fitted to the training set using the *caret* package in R. For each model, the parameter tuning was done when necessary. Some additional packages were used for this purpose, specially to get the plots such as *randomForest* and *rpart* for tuning random forest, *e1071* for tuning support vector machines, *fastAdaboost* for adaptive boosting.

The *caret* package chooses the best model after searching through the specified parameter values. The models were validated by 10-fold repeated cross validation with 10 repeats using the testing set. The accuracy measures were accuracy and Cohen's Kappa coefficient.

The K-nearest neighbor has only one tuning parameter, K. The value of K was chosen by looking at the accuracy plot. K=9 gives the highest accuracy for the model.

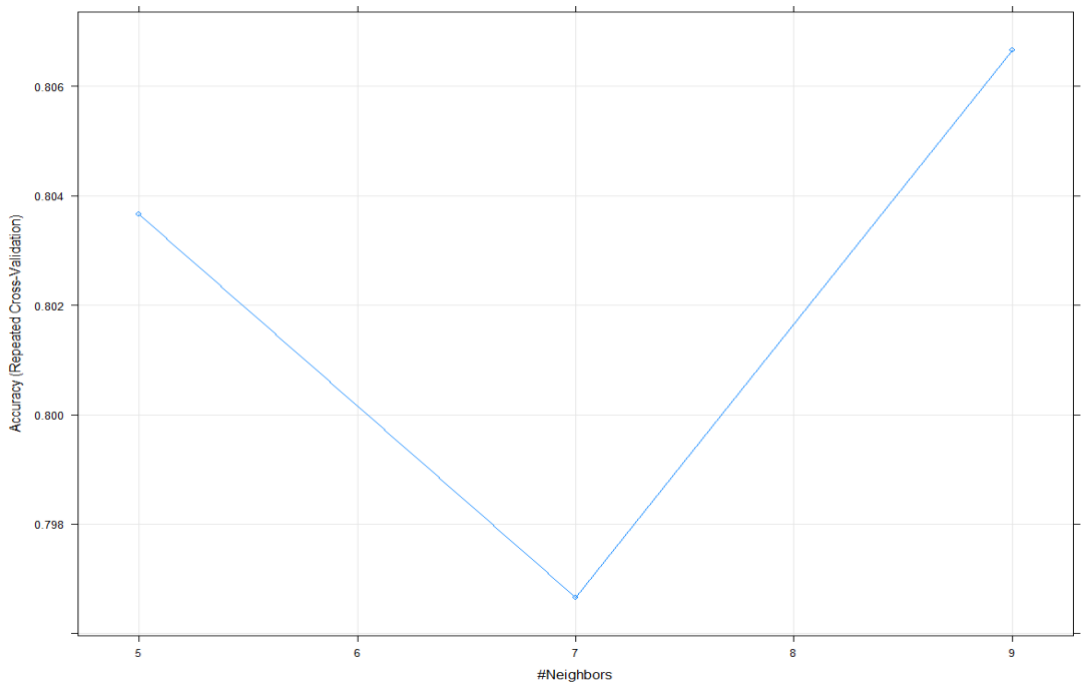

Figure 3.2 Accuracy plot for different K values

For training support vector machines, the radial basis function gives the best result. The cost parameter value C=0.25 was chosen since it gives the highest accuracy for the radial support vector machine with tuning parameter sigma=0.03409.

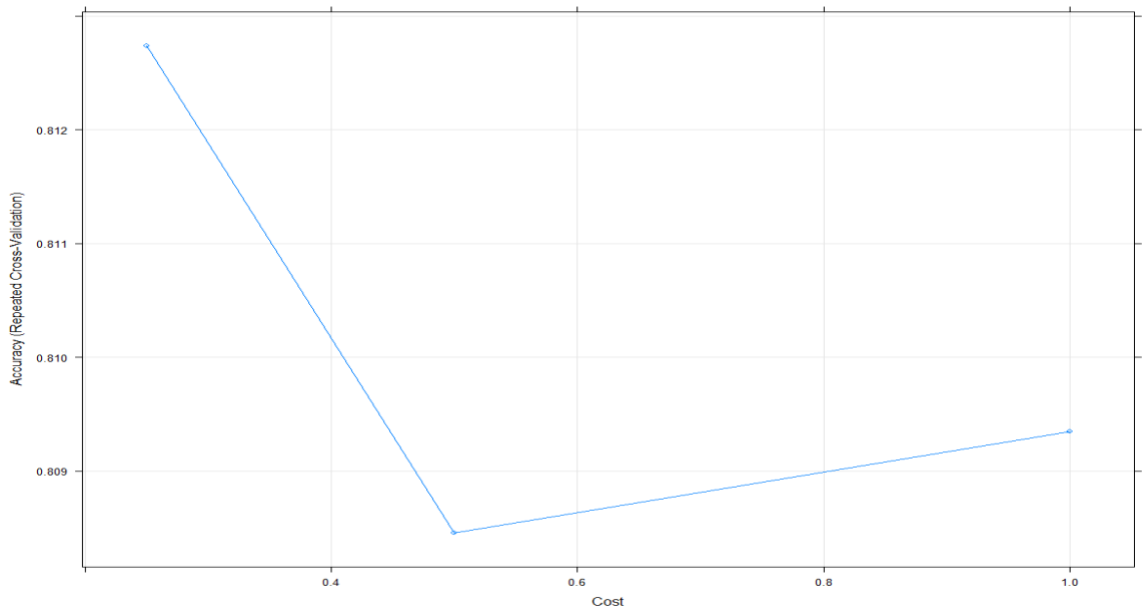

Figure 3.3 Accuracy plot for different cost values

For tuning random forest models, we tune the complexity parameter, CP. Here, CP=0.018 was chosen since it produces the lowest relative error for different tree sizes.

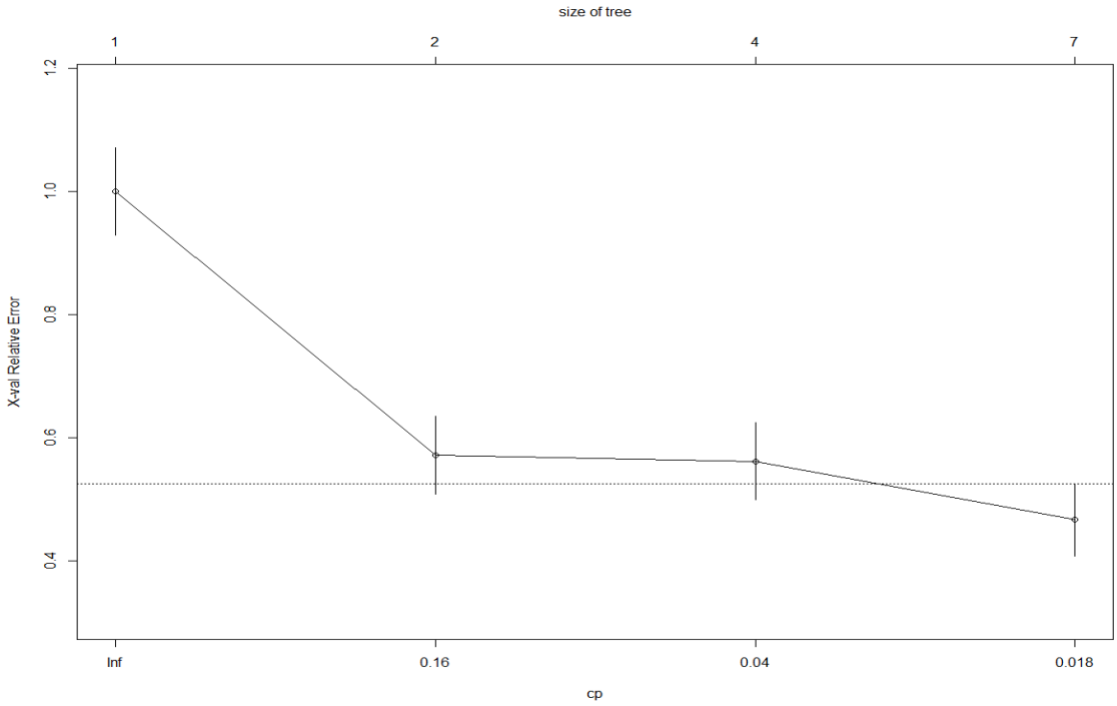

Figure 3.4 Relative error plot for complexity parameter

The random forest was also utilized for variable selection by exploring the variable selection plot. Except for the variable “fbs”, that is, fasting blood sugar, all the variables were important and so they were kept in the model.

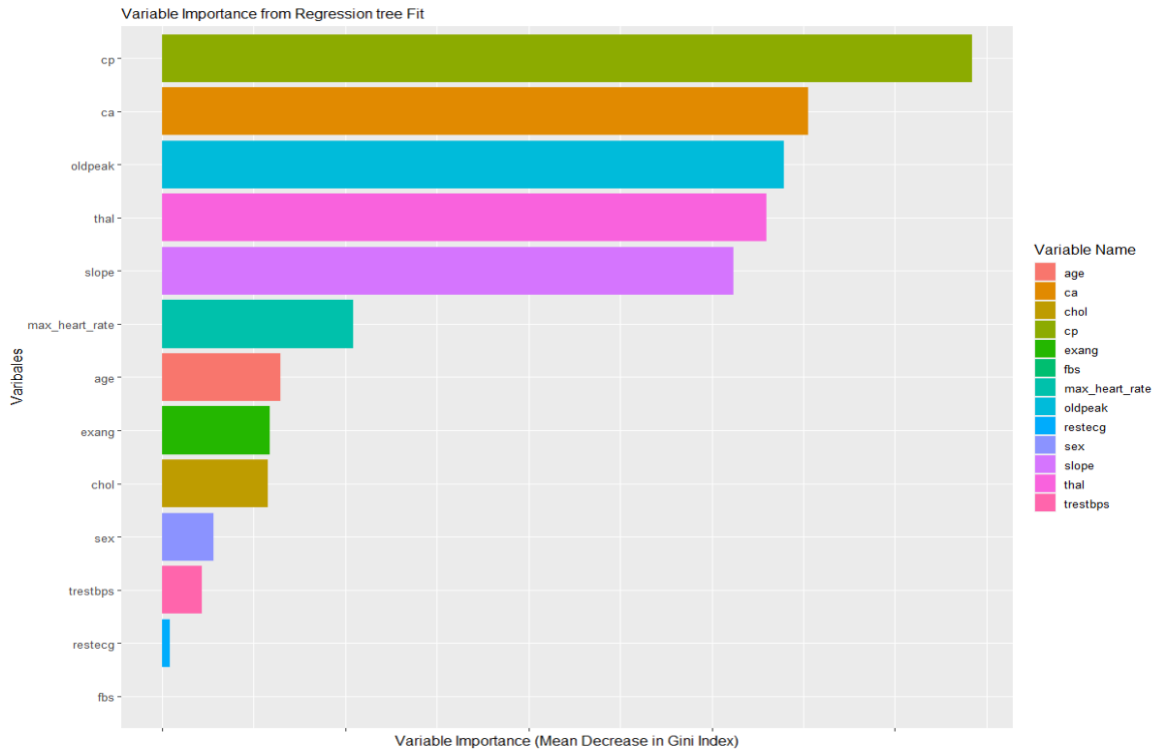

Figure 3.5 Variable importance plot using random forest

Similarly, for the adaptive boosting algorithm, the number of trees and method was selected by using the accuracy plot. The real adaptive boosting method gives the highest accuracy when the number of trees utilized is 150.

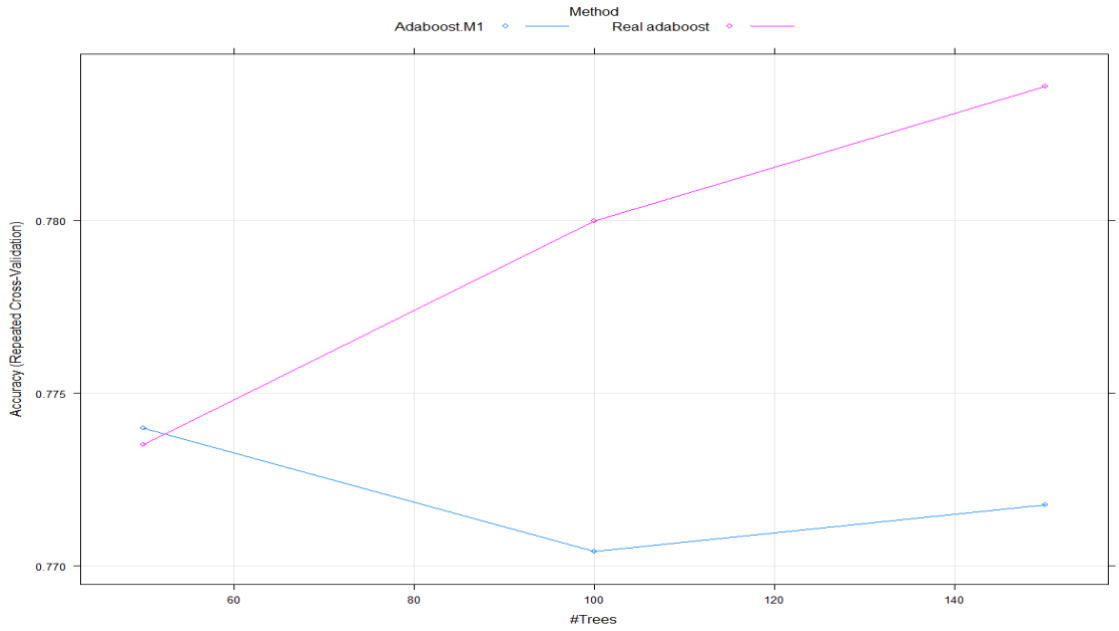

Figure 3.6: Accuracy plot for adaptive boosting

Lastly, the gradient boosting model was tuned to select the maximum depth of trees.

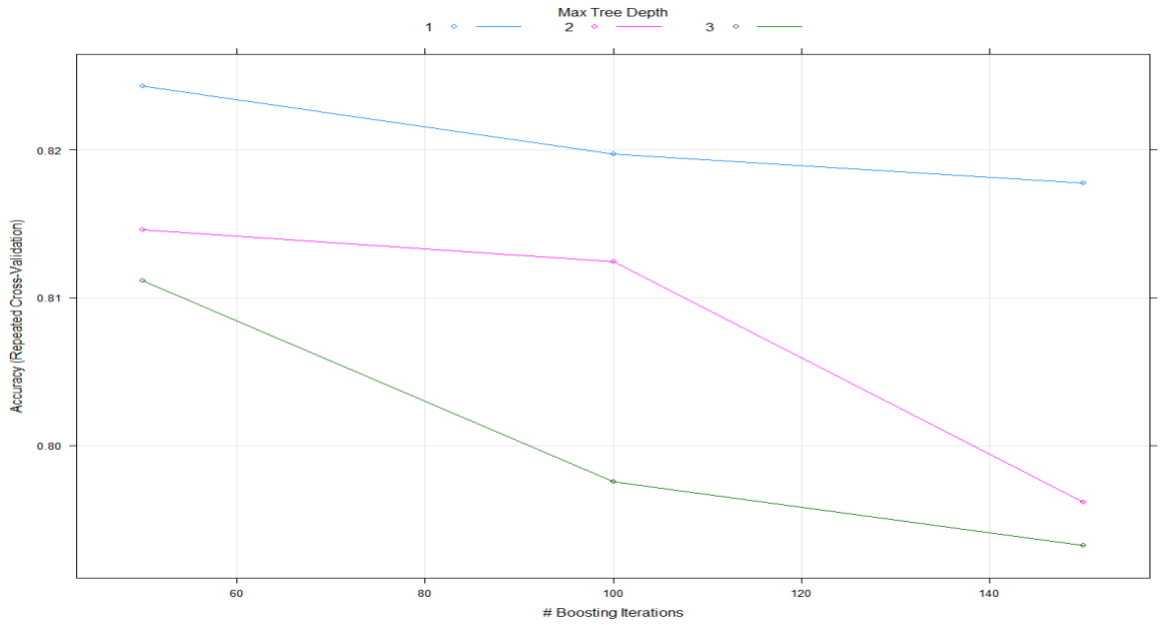

Figure 3.7: Accuracy plot for gradient boosting algorithm

3.3 Result

The models' performances are compared using the validation measures- accuracy and kappa coefficient. It can be seen that while all the classification models perform well, the gradient boosting algorithm gives the best results followed by support vector machines and random forest.

Table 3.2: The performance metrics of models:

Model	Accuracy	Kappa
GBM	0.838	0.673
SVM	0.835	0.667
RF	0.827	0.651
KNN	0.826	0.664
LDA	0.820	0.635
AdaBoost	0.818	0.631
Logistic	0.816	0.629

The boxplots of the accuracies show that, K-nearest neighbors has the narrowest spread. Logistic regression and adaptive boosting algorithm have some outliers.

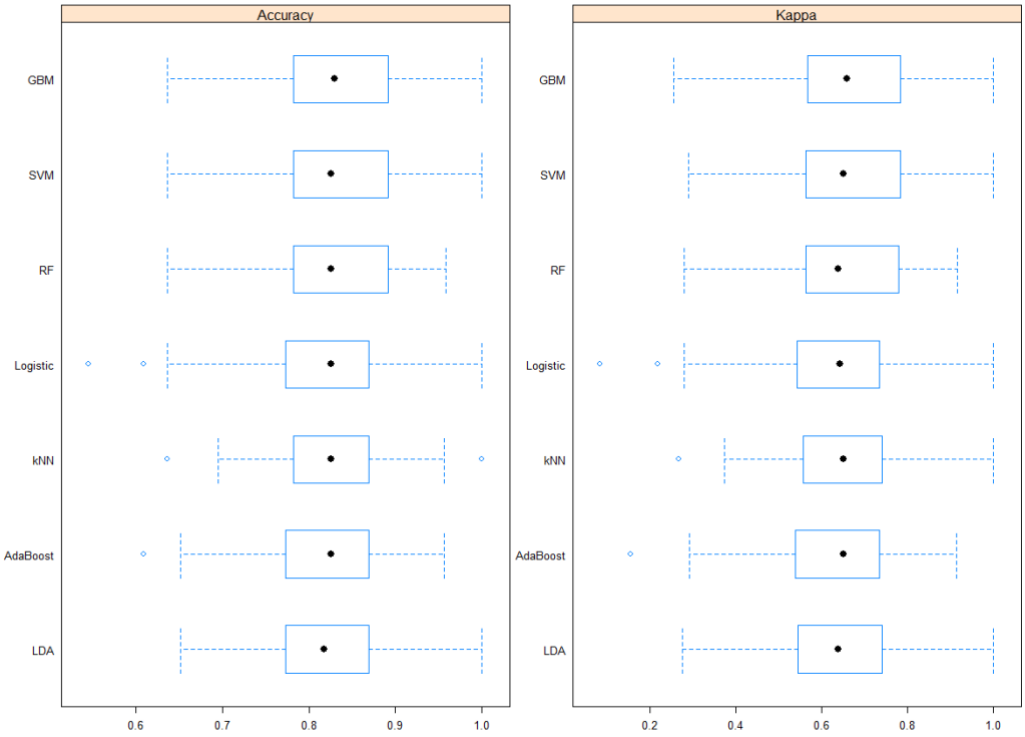

Figure 3.8: Confidence interval of accuracies for classification models

Overall, the gradient boosting algorithm has the best performance with 83.8% accuracy on average. This is consistent with the simulation results for the multivariate normal distribution. The support vector machines have 83.5% accuracy which is the next best results. K-nearest neighbor does moderately well for both the training and the testing dataset. But it has the narrowest spread, so its accuracy level is more consistent than others. Linear discriminant analysis gives better accuracy than logistic regression, which is understandable, since they are designed for a multiclass target variable. Logistic regression provides the least accuracy among all seven models with 81.6% accuracy. Overall, for the heart disease prediction, the gradient boosting algorithm should be used since it gives the highest accuracy.

CHAPTER 4

CONCLUSION

In this thesis a comparison study of popular supervised learning models for classification were presented. The goal was to identify which models perform better under different settings. The models considered were K-nearest neighbors, logistic regression, linear discriminant analysis, support vector machines, random forest, adaptive boosting and gradient boosting algorithms.

The data was generated from three different distributions: multivariate normal, multivariate T and multivariate log-normal. For each distribution, the models were fitted for three (p=3) and ten sets of predictor variables (p=10). The dependent variable had two classes (k=2) and five classes (k=5) for every setting. The models were also compared for increasing sample sizes of n=100, 300 and 500. Balanced datasets were considered for all the settings except for sample size 500 with two classes to differentiate between performance for balanced and unbalanced datasets. The generated datasets were scaled before fitting the models.

The metrics used for evaluating the models were classification accuracy calculated from the confusion matrix and Cohen's kappa coefficient, which measures how closely the values predicted by the models match the true values. The model parameters were tuned to find the best model possible for each setting.

For the multivariate normal distribution, gradient boosting algorithms had the best results at different settings overall. As predictor numbers increased, the support vector machines had higher accuracy, but gradient boosting's accuracy was close to it. Along with gradient boosting, the random forest algorithm also had very well prediction

accuracy throughout the settings. Both performed well when the number of classes were two and five. They also had narrower spreads than others, but K-nearest neighbor had the shortest spread for five classes. The accuracies for all the models increased as sample size was increased, except for when the number of predictors was small, and the number of classes were high.

When data was generated from a multivariate T distribution, gradient boosting and random forest produced higher accuracies than others. Linear discriminant analysis and logistic regression had the worst values. The consistency of performance of models changed when the number of classes changed. For a smaller number of predictors, linear and two class discriminant analysis had the narrowest spread but for 5 classes it had the largest spread. Support vector machines accuracy was the highest for the unbalanced dataset followed by the gradient boosting accuracy. Increasing sample size gave better performance measures for most of the settings. Support vector machines and linear discriminant analysis accuracies decreased when there were five classes instead of two with three predictors.

For data generated from a multivariate log-normal distribution, almost all the models performed significantly well when the number of classes were two. But for more difficult classification with five classes, random forest and linear discriminant analysis had the highest accuracies. K nearest neighbor performed the worst. The spread for random forest and gradient boosting was the narrowest for most of the settings.

The results from the simulation study were consistent with the results of real-life data for heart patients. The numerical variables in the dataset mostly followed normal distribution. The gradient boosting algorithm gave the best prediction accuracy followed by support vector machines. K- nearest neighbors had the narrowest spread of accuracies.

Overall, while there is no one method best for every setting, gradient boosting and random forest algorithms seemed to give good results throughout. Especially if the parameters are tuned carefully, they produce high prediction accuracies. Further studies can be done for classes with unequal variances and larger number of classes.

REFERENCES

1. James, G., Witten, D., Hastie, T. and Tibshirani, R (2013), Introduction to Statistical Learning: with Applications in R, New York: Springer
2. Silverman, B., & Jones, M. (1989). E. Fix and J.L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951). International Statistical Review / Revue Internationale De Statistique,57(3), 233-238. doi:10.2307/1403796
3. Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780), 1612.
4. Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, *36*(1-2), 105-139.
5. Caruana, R., & Niculescu-Mizil, A. (2006, June). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning* (pp. 161-168).
6. Narassiguin, A., Bibimoune, M., Elghazel, H., & Aussem, A. (2016). An extensive empirical comparison of ensemble learning methods for binary classification. *Pattern Analysis and Applications*, *19*(4), 1093-1128.
7. Armitage, D. W., & Ober, H. K. (2010). A comparison of supervised learning techniques in the classification of bat echolocation calls. *Ecological Informatics*, *5*(6), 465-473.

8. Ahmed, N. K., Atiya, A. F., Gayar, N. E., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, *29*(5-6), 594-621.
9. Shao, Y., & Lunetta, R. S. (2012). Comparison of support vector machine, neural network, and CART algorithms for the land-cover classification using limited training data points. *ISPRS Journal of Photogrammetry and Remote Sensing*, *70*, 78-87.
10. Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, *39*(3), 3446-3453.
11. Phyu T. N. (2009), "Survey of Classification Techniques in Data Mining", International MultiConference of Engineers and Computer Scientists, Vol IIMECS 2009, Hong Kong
12. Castanon, J. (2019), 10 Machine Learning Methods that Every Data Scientist Should Know, Towards Data Science, Available athttps://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9
13. Liberman, N. (2017), Decision Trees and Random Forests, Towards Data Science, Available athttps://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991
14. Ray, S. (2017), Understanding Support Vector Machine Algorithm from Examples, Analytics Vidhya, Available athttps://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

15. Hoare, J. (2017), Linear Discriminant Analysis in R: an introduction, DisplayR blog, Available athttps://www.displayr.com/linear-discriminant-analysis-in-r-an-introduction/

APPENDIX A: TABLES FROM SIMULATION STUDY

Table A.1: Accuracy for multivariate T when p=3 & k=2

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	0.55	.56	0.55	0.515	0.516	0.542	0.557
n=300	0.59	0.581	0.62	0.60	0.586	0.60	0.58
n=500	0.624	0.611	0.654	0.634	0.621	0.638	0.652

Table A.2: Kappa for multivariate T when p=3 & k=2:

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	0.092	0.124	0.07	0.024	0.033	0.036	0.109
n=300	0.195	0.161	0.238	0.199	0.185	0.227	0.160
n=500	0.217	0.179	0.272	0.236	0.119	0.173	0.265

Table A.3: Accuracy for multivariate lognormal when p=3 & k=2

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	0.952	0.967	0.954	0.949	0.970	0.971	0.954
n=300	0.987	0.987	0.983	0.982	0.984	0.978	0.986
n=500	0.958	0.967	0.957	0.951	0.9634	0.637	0.960

Table A.4: Kappa for multivariate lognormal when p=3 & k=2:

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	0.904	0.935	0.908	0.898	0.940	0.941	0.909
n=300	0.964	0.975	0.966	0.964	0.969	0.957	0.972
n=500	0.912	0.931	0.911	0.898	0.923	0.924	0.918

Table A.5: Accuracy for multivariate T when p=10 & k=2

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	0.995	0.747	0.644	0.995	0.575	0.658	0.995
n=300	0.995	0.747	0.644	0.995	0.575	0.658	0.995
n=500	0.995	0.844	0.738	0.994	0.609	0.739	0.994

Table A.6: Kappa for multivariate T when p=10 & k=2:

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	0.990	0.491	0.280	0.991	0.113	0.324	0.991
n=300	0.990	0.491	0.280	0.991	0.113	0.324	0.991
n=500	0.990	0.674	0.431	0.987	0.044	0.441	0.988

Table A.7: Accuracy for multivariate lognormal when p=10 & k=2

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	0.997	0.823	0.732	0.994	0.600	0.670	0.996
n=300	0.971	0.944	0.932	0.971	0.956	0.948	0.973
n=500	0.963	0.962	0.955	0.963	0.966	0.953	0.961

Table A.8: Kappa for multivariate lognormal when p=10 & k=2:

Sample size	RF	KNN	SVM	AdaBoost	LDA	Logistic	GBM
n=100	0.994	0.637	0.424	0.988	0.038	0.253	0.992
n=300	0.943	0.887	0.863	0.942	0.912	0.897	0.946
n=500	0.923	0.921	0.907	0.924	0.930	0.903	0.919

Table A.9: Accuracy for multivariate T when p=3 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	0.955	0.881	0.303	0.277	0.954
n=500	0.984	0.934	0.260	0.238	0.984

Table A.10: Kappa for multivariate T when p=3 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	0.980	0.917	0.713	0.043	0.981
n=500	0.943	0.850	0.112	0.078	0.943

Table A.11: Accuracy for multivariate lognormal when p=3 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	1.00	1.00	1.00	1.00	1.00
n=500	0.996	1.00	1.00	1.00	0.995

Table A.12: Kappa for multivariate lognormal when p=3 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	1.00	1.00	1.00	1.00	0.994
n=500	0.995	1.00	1.00	1.00	0.993

Table A.13: Accuracy for multivariate T when p=10 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	0.837	0.743	0.317	0.228	0.829
n=500	0.885	0.769	0.375	0.240	0.885

Table A.14: Kappa for multivariate T when p=10 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	0.7963	0.678	0.150	0.030	0.786
n=500	0.856	0.711	0.219	0.055	0.856

Table A.15Accuracy for multivariate lognormal when p=10 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	0.807	0.603	0.782	0.816	0.781
n=500	0.847	0.641	0.847	0.8403	0.8404

Table A.16: Kappa for multivariate lognormal when p=10 & k=5

Sample size	RF	KNN	SVM	LDA	GBM
n=300	0.758	0.503	0.728	0.769	0.726
n=500	0.808	0.550	0.809	0.8002	0.800

APPENDIX B: FIGURES FROM SIMULATION STUDY

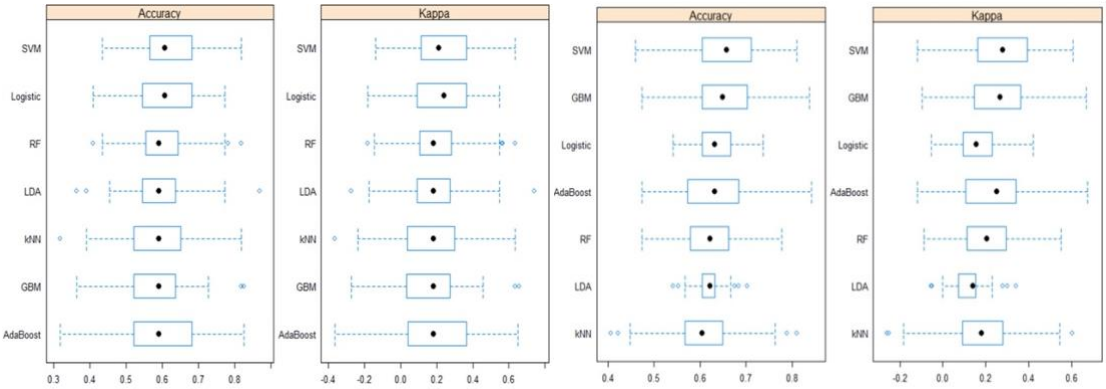

Figure B.1: Boxplot for n=300 & n=500 for multivariate-T, p=3 & k=2

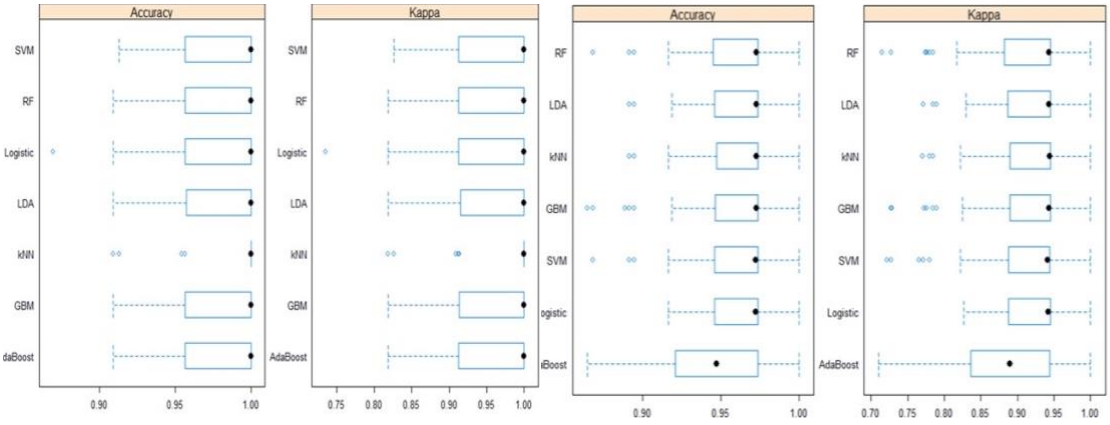

Figure B.2: Boxplot for n=300 & n=500 for multivariate-lognormal, p=3 & k=2

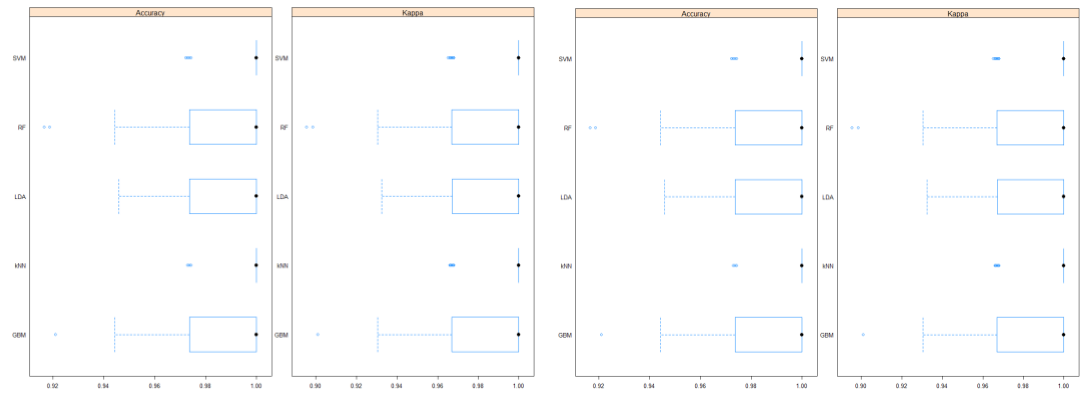

Figure B.3: Boxplot for n=300 & n=500 for multivariate normal, p=3 & k=5

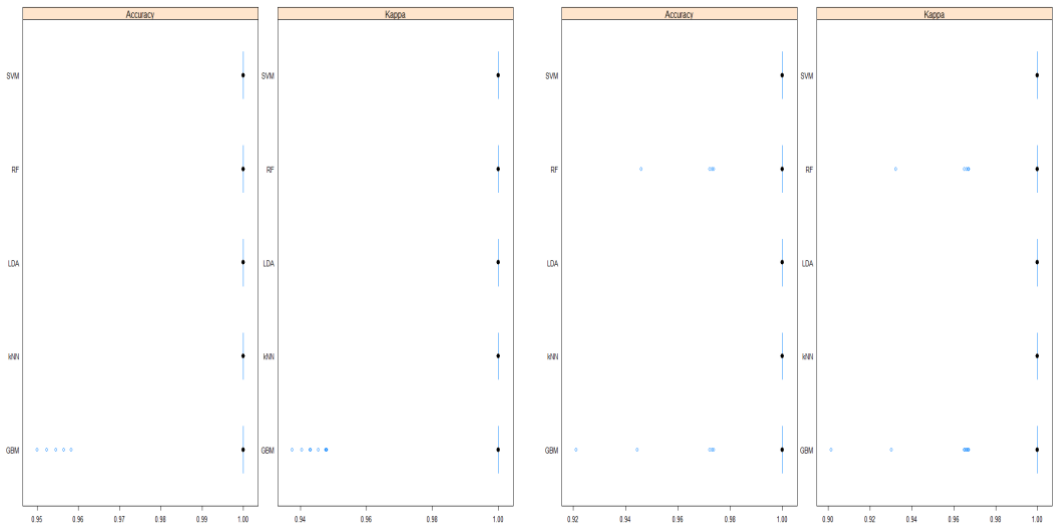

Figure B.4: Boxplot for n=300 & n=500 for multivariate lognormal, p=3 & k=5

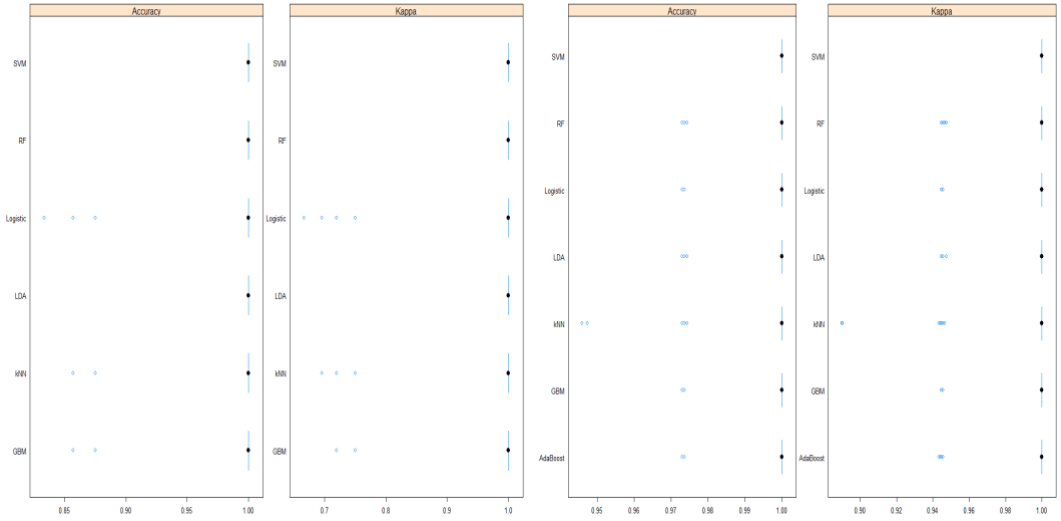

Figure B.5: Boxplot for n=300 & n=500 for multivariate normal, p=10 & k=2

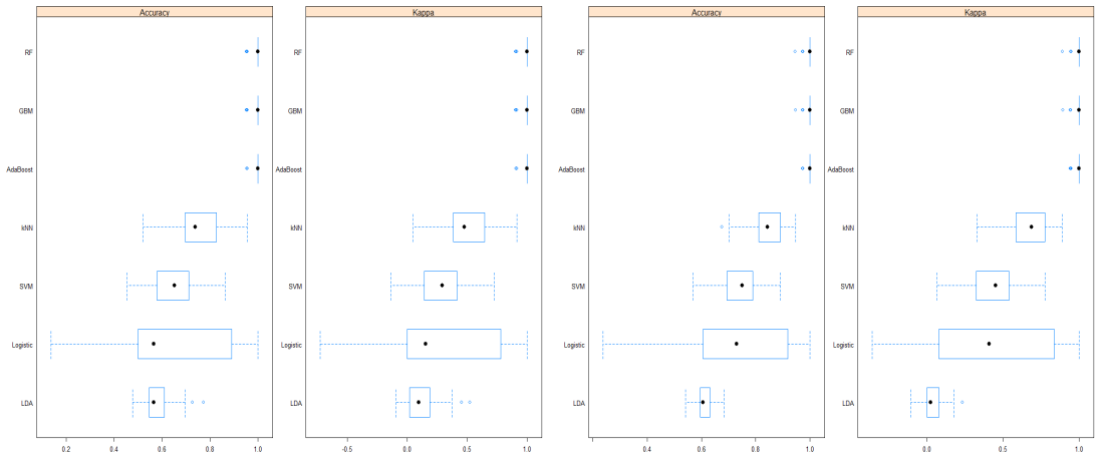

Figure B.6: Boxplot for n=300 & n=500 for multivariate T, p=10 & k=2

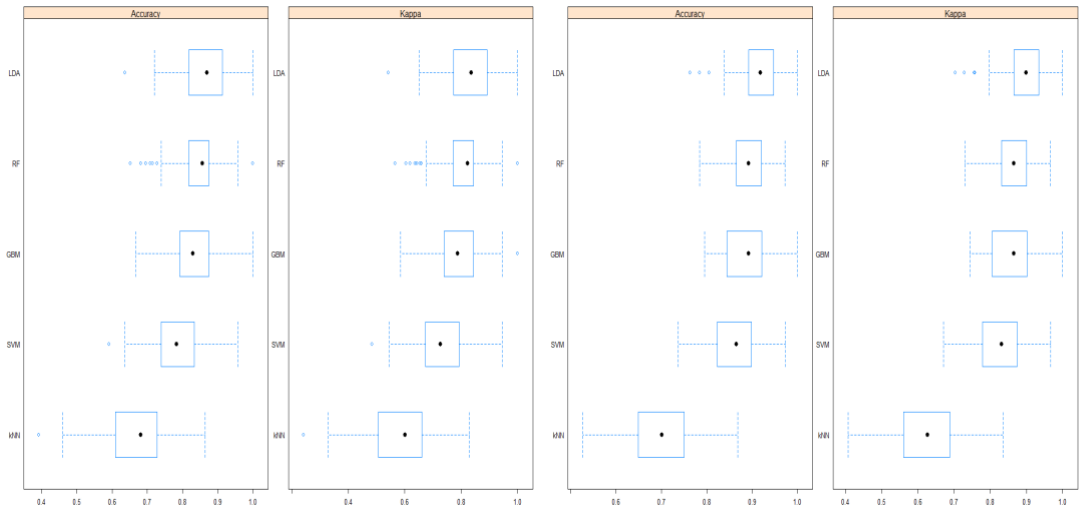

Figure B.7: Boxplot for n=300 & n=500 for multivariate normal, p=10 & k=5

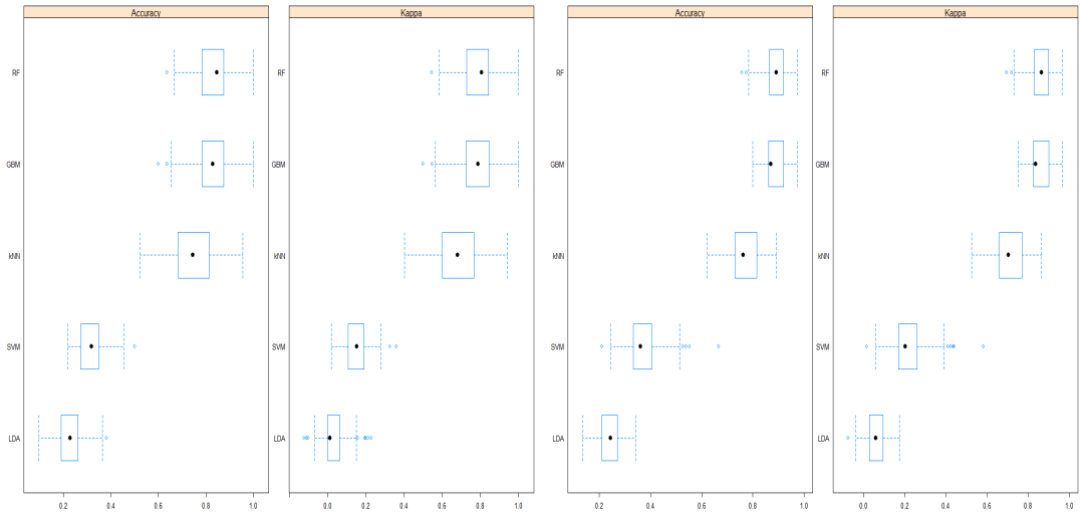

Figure B.8: Boxplot for n=300 & n=500 for multivariate T, p=10 & k=5

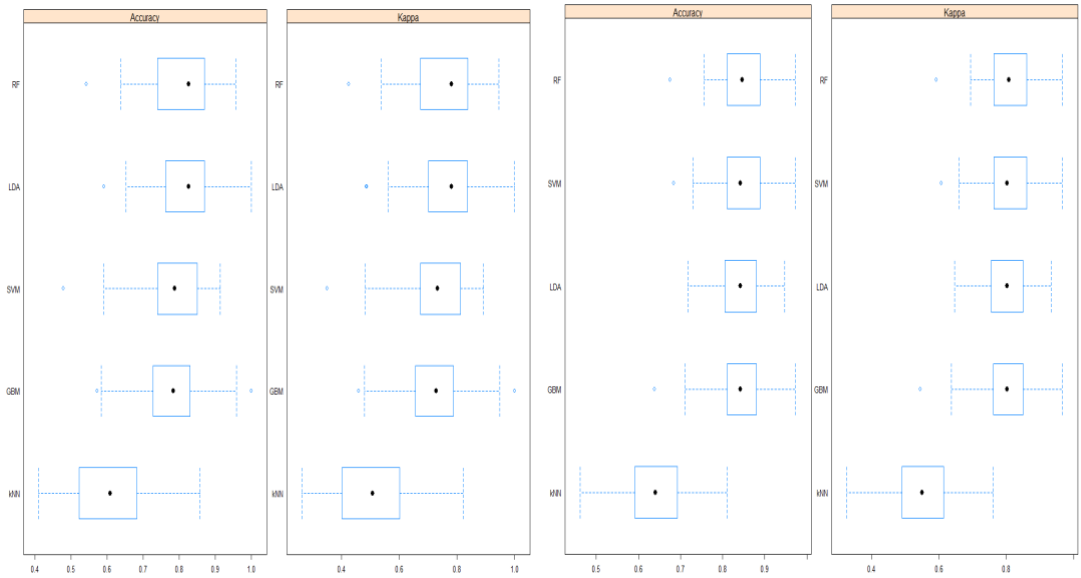

Figure B.9: Boxplot for n=300 & n=500 for multivariate lognormal, p=10 & k=5