MATH 748 Final Report
Professor Tao He, Ph.D.

Diabetes Prediction Model

Safaa Dorian
December 20, 2020

Executive Summary

In analyzing a dataset describing the health characteristics of 768 Pima Native American female adults, we find that the most significant variables which can predict diabetes for this population are Blood Glucose Levels, Body Mass Index (BMI), Insulin Levels, and Age. This determination is made by using decision tree methods, namely cross-validation, bagging, and random forest, which provide the most accurate predictions.

Although the regression methods give us some inferences about the predictors and how much they predict the outcome, their general accuracy of 70-75% is not as reliable as the KNN, support vector machine, and decision tree methods which provide 80-85% accuracy.

Table of Contents

I. Introduction

According to the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK), “diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high… Over time, having too much glucose in your blood can cause health problems. Although diabetes has no cure, you can take steps to manage diabetes and stay healthy.” (2016)

The objective of this project is to use a dataset to build a model that can predict whether a patient has diabetes, based on certain factors such as age, family history of diabetes, weight, race and more diagnostic measures like blood pressure. In addition, based on the outcome of these predictors, a person can take steps to prevent the development of some health problems related to diabetes.

II. Data Exploration

The original dataset can be found on Kaggle.com, under the UCI Machine Learning page, and is from the Pima Indians Diabetes Database (2018). Starting from subsection C below and onward, RStudio was used in order to analyze the data.

A. Understanding the data

The dataset describes health characteristics of 768 Pima Native American female adults. It consists of 9 variables which are divided into 8 independent variables and 1 outcome. The predictor/independent variables are number of pregnancies, body mass index (BMI), insulin level, glucose level, age, skin thickness, blood pressure, diabetes pedigree function; and the outcome variable is having diabetes or not.

B. List of variables

i. Independent variables:

- Pregnancies: Number of pregnancies that the women had (Numeric)
- Glucose: Plasma glucose concentration 2 hours in an oral glucose tolerance test (Numeric)
- Blood Pressure: Diastolic blood pressure (mm Hg) (Numeric)
- Skin Thickness: Triceps skin fold thickness (mm) (Numeric)

- Insulin: 2-Hour serum insulin (mu U/ml) (Numeric)
- BMI: Body mass index in kg/m^2 (Numeric)
- Diabetes Pedigree Function: A function which scores the likelihood of diabetes based on family history. It provided some data on diabetes mellitus history in relatives and the genetic relationship of those relatives to the patient (Numeric)

$$\text{DPF}=\frac{\sum Ki(88-ADMi)+20}{\sum Kj(ALCj-14)+50}$$

i: The total number of relatives with diabetes
j: The total number of relatives without diabetes
x: Degree of genetic match with a particular relative
(0.5: Parent, brothers / 0.25: Grandparents, Brothers of Parents / 0.125: The children of brothers of parents)
ADM: the age at which relatives with diabetes(i) developed.
ACL: the age at which a non-diabetic relative (j) tested for diabetes

- Age: Age in years (Numeric)

ii. Dependent variable:

- Outcome: class variable (0 or 1), 0 for a healthy woman and 1 for woman that has diabetes (Boolean)

C. Descriptive statistics

Table 1: Summary of data

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
Min. : 0.000	Min. : 0.0	Min. : 0.00	Min. : 0.00	Min. : 0.0	Min. : 0.00
1st Qu.: 1.000	1st Qu.: 99.0	1st Qu.: 62.00	1st Qu.: 0.00	1st Qu.: 0.0	1st Qu.:27.30
Median : 3.000	Median :117.0	Median : 72.00	Median :23.00	Median : 30.5	Median :32.00
Mean : 3.845	Mean :120.9	Mean : 69.11	Mean :20.54	Mean : 79.8	Mean :31.99
3rd Qu.: 6.000	3rd Qu.:140.2	3rd Qu.: 80.00	3rd Qu.:32.00	3rd Qu.:127.2	3rd Qu.:36.60
Max. :17.000	Max. :199.0	Max. :122.00	Max. :99.00	Max. :846.0	Max. :67.10

DiabetesPedigreeFunction	Age	Outcome
Min. :0.0780	Min. :21.00	Min. :0.000
1st Qu.:0.2437	1st Qu.:24.00	1st Qu.:0.000
Median :0.3725	Median :29.00	Median :0.000
Mean :0.4719	Mean :33.24	Mean :0.349
3rd Qu.:0.6262	3rd Qu.:41.00	3rd Qu.:1.000
Max. :2.4200	Max. :81.00	Max. :1.000

In examining Table 1 above, we can say that on average the women that participated in the study are 33 years old and have had around 4 pregnancies. Their BMI is close to 32 kg/m^2 which is considered obese, and that explains the measurement of skin thickness that is close to 21 mm. The average level of the glucose, insulin, and blood pressure are 120 mg/dL, 127 U/ml and 69 mm Hg, respectively.

This dataset does not have any missing values. But we notice from the table that some variables such as glucose, insulin, blood pressure, skin thickness and BMI have a minimum value of 0. This value does not make any sense in context and it seems very likely that these zero values encode missing data. Therefore, we need to replace these zeros by null.

D. Data visualization

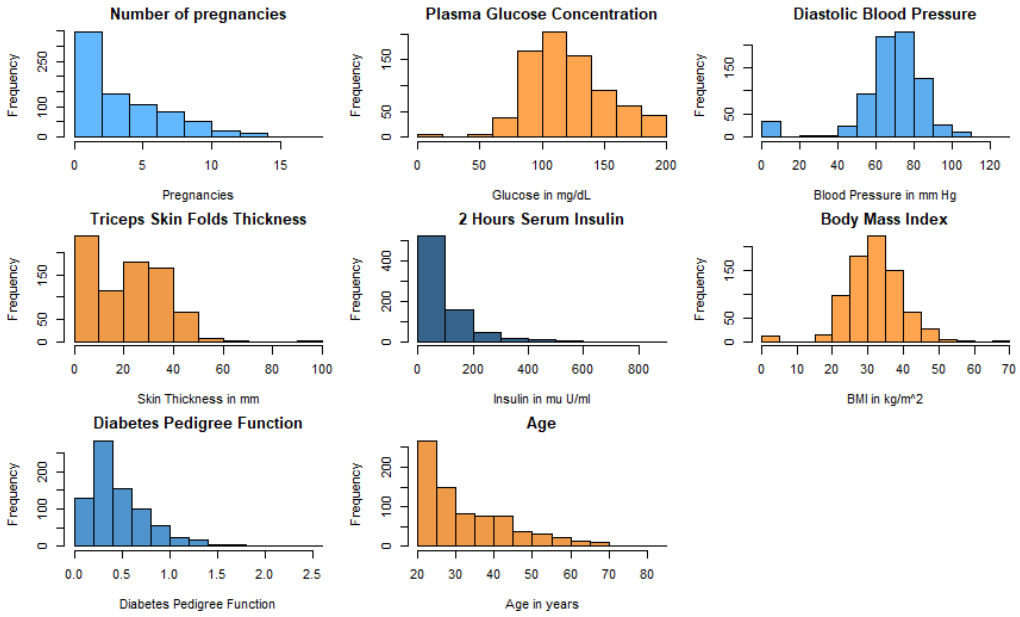


Figure 1: Histograms of independent variables

Analyzing the histograms in Figure 1 above, we notice some variables are skewed to the left such as Pregnancies, Insulin, Diabetes Pedigree Function, and age.

To have a better understanding of the data we transformed the variable Outcome from Boolean (0,1), to categorical (Non-Diabetic, Diabetic). As seen in Figure 2 below, 65% of the women are healthy and 35% are diabetic.

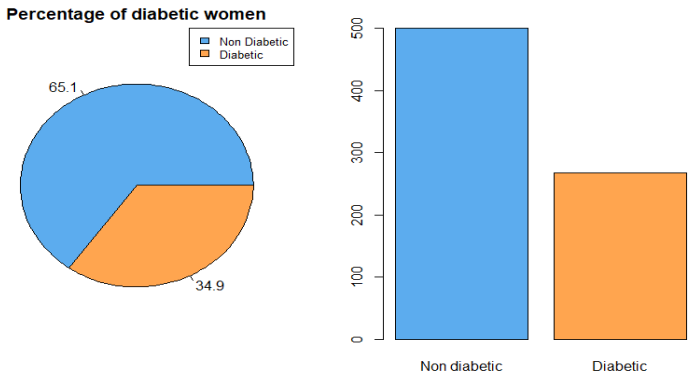


Figure 2: Pie and bar charts of the outcome variable

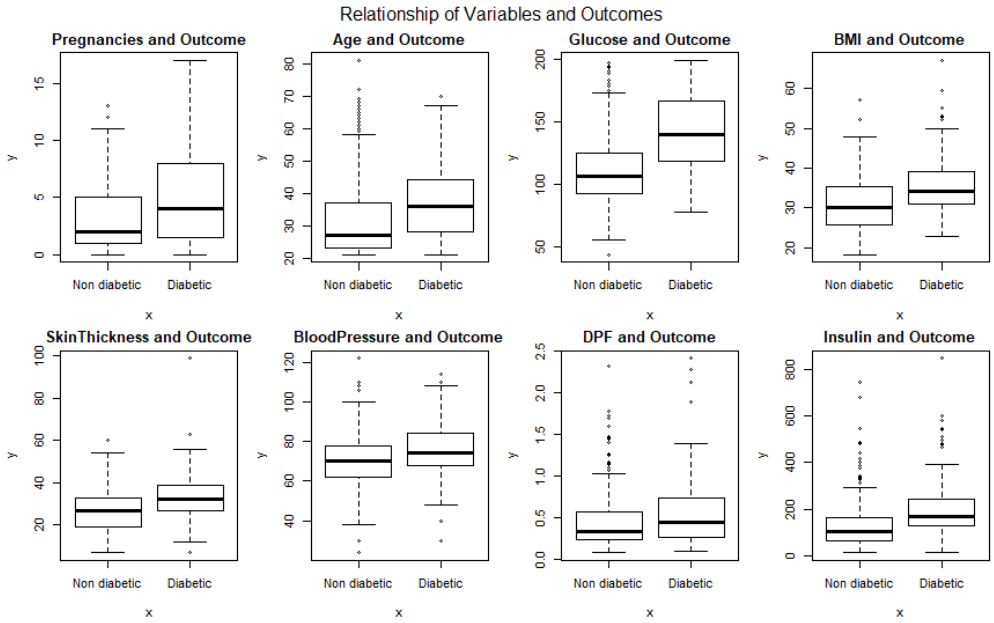


Figure 3: Boxplots of independent variables, based on the outcome

In the boxplots shown above, we notice that for almost all the variables the women are diabetic when they have a larger value. On average, the diabetic women are in their forties, have had an average of 5 pregnancies, and their BMI is closer to 40, which is considered obese. Glucose and Insulin levels of diabetic women are very high (around 150 mg/dL for glucose and 200 U/ml for Insulin).

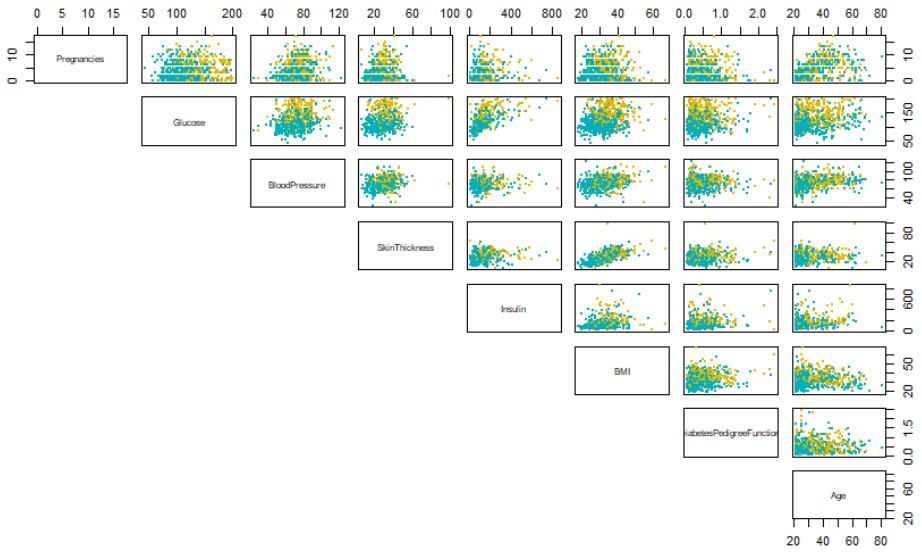


Figure 4: Scatterplot matrix

In the scatterplot matrix above, we are trying to roughly determine if we have a linear correlation between variables. We can say that Glucose and Insulin, Skin Thickness and BMI show positive trends. But we need to confirm this through the correlation matrix.

III. Data Cleaning

A. Missing Value

As we noticed earlier from the summary table (Table 1), we have a value of zero as a minimum value for some variables like Glucose, Insulin, Blood Pressure and BMI. We are suspecting that these values are missing data since it is impossible for someone to have a very low level of glucose, insulin, blood pressure or even BMI. To predict these missing values, we replaced the zeros with null. There are 652 total missing values, as depicted in table 2 below:

Table 2: Summary of data after transforming zeros to NA

```
 Pregnancies        Glucose      BloodPressure    SkinThickness      Insulin           BMI
Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00   Min.   : 14.00   Min.   :18.20
1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.:22.00   1st Qu.: 76.25   1st Qu.:27.50
Median : 3.000   Median :117.0   Median : 72.00   Median :29.00   Median :125.00   Median :32.30
Mean   : 3.845   Mean   :121.7   Mean   : 72.41   Mean   :29.15   Mean   :155.55   Mean   :32.46
3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:36.00   3rd Qu.:190.00   3rd Qu.:36.60
Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.00   Max.   :67.10
                 NA's   :5       NA's   :35       NA's   :227     NA's   :374      NA's   :11
DiabetesPedigreeFunction      Age          Outcome
Min.   :0.0780           Min.   :21.00   Min.   :0.000
1st Qu.:0.2437           1st Qu.:24.00   1st Qu.:0.000
Median :0.3725           Median :29.00   Median :0.000
Mean   :0.4719           Mean   :33.24   Mean   :0.349
3rd Qu.:0.6262           3rd Qu.:41.00   3rd Qu.:1.000
Max.   :2.4200           Max.   :81.00   Max.   :1.000

> sum(is.na(diabetes))
[1] 652
```

Since these variables are numeric, a regression model can be used to predict these missing values. We will use the K-nearest neighbors algorithm K-NN method.

B. Correlation Matrix

Table 3: Correlation matrix of the numeric variables

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Pregnancies	1.00000000	0.1300741	0.2140764696	0.1320222	0.1008247	0.02209358	-0.0335226730	0.54434123
Glucose	0.13007413	1.0000000	0.2289397099	0.2317676	0.6129553	0.23684548	0.1384564487	0.26885045
BloodPressure	0.21407647	0.2289397	1.0000000000	0.2269567	0.1549335	0.29441029	-0.0008954921	0.33365445
SkinThickness	0.13202219	0.2317676	0.2269567112	1.0000000	0.2515871	0.66486115	0.1290415648	0.16062798
Insulin	0.10082471	0.6129553	0.1549335461	0.2515871	1.0000000	0.28764155	0.1550272700	0.27635325
BMI	0.02209358	0.2368455	0.2944102934	0.6648612	0.2876415	1.00000000	0.1535030251	0.02718476
DiabetesPedigreeFunction	-0.03352267	0.1384564	-0.0008954921	0.1290416	0.1550273	0.15350303	1.0000000000	0.03356131
Age	0.54434123	0.2688504	0.3336544468	0.1606280	0.2763533	0.02718476	0.0335613124	1.00000000

From this correlation matrix we confirm what we saw in the scatterplot matrix. Glucose and Insulin have a positive correlation (≈0.6). That means that as the glucose level increases in our bodies the insulin level increases as well. And we have another positive correlation between skin thickness and BMI (≈0.66). This can be explained by the extra fat that a person gains with a higher BMI. The third positive

correlation that we can see is between Age and number of pregnancies (≈0.54). This is logical since a woman needs years to have more children.

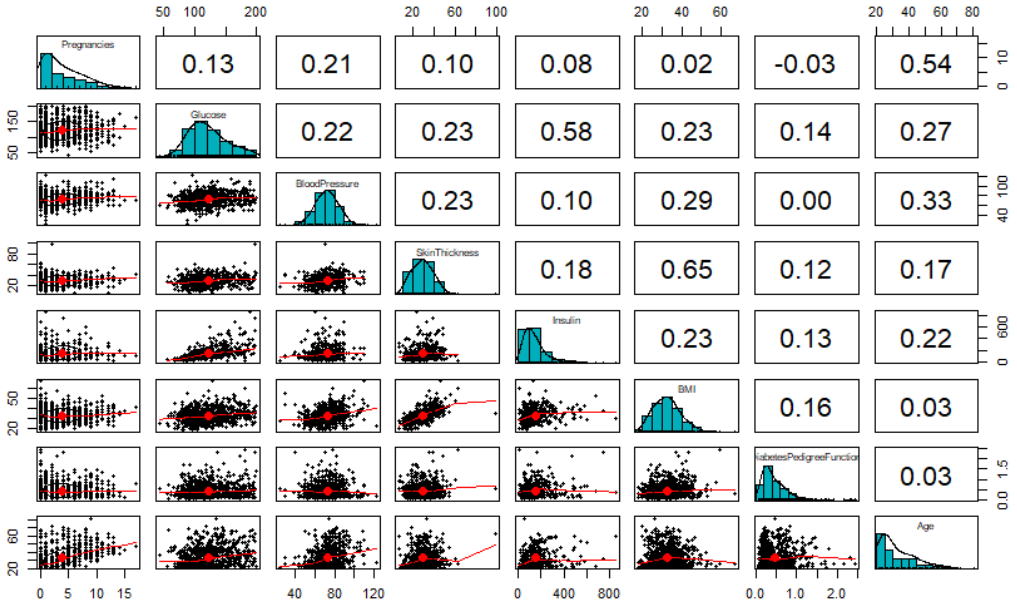


Figure 5: Matrix of scatterplot with coefficients of correlation

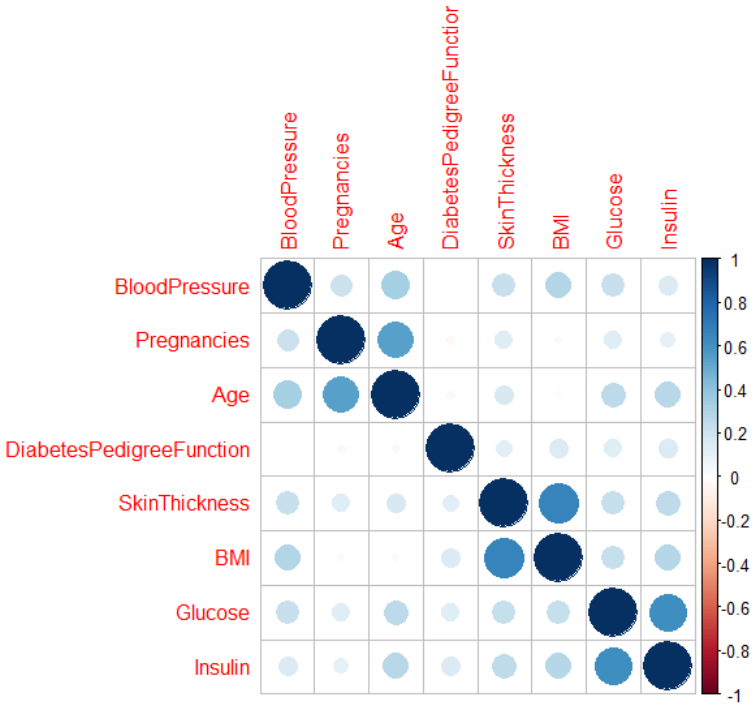


Figure 6: Correlation matrix

Figures 5 and 6 confirm what we already concluded from the correlation matrix of numeric values in Table 3. Glucose and Insulin, Pregnancies and Age, Skin Thickness and BMI have relatively high values. Diabetes Pedigree Function appears to have little correlation with other variables. Since all the coefficients of correlation are less than 0.75, we don't need to remove any variables.

C. Outliers detection

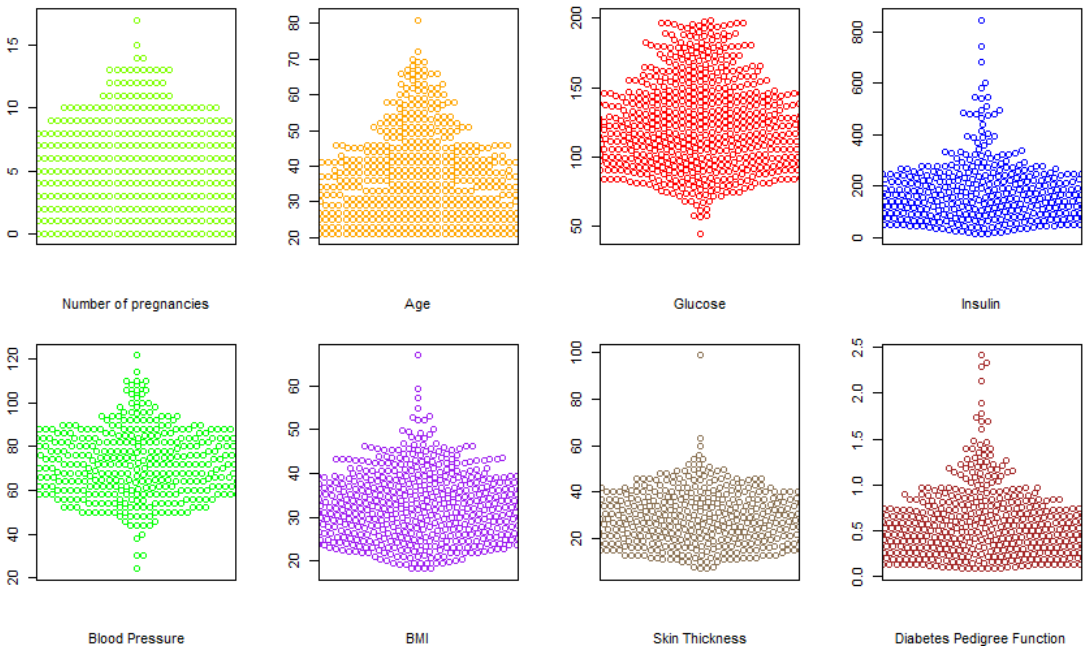


Figure 7: Beeswarm charts

From the beeswarm plots above we can see that some of the variables have some outliers, but we decided to not remove or replace them, so it doesn't affect our prediction model.

D. Near zero variance

Checking the variances of the variables, we notice that we don't have any variable that has a variance near zero.

E. Skewness

Checking the level of skewness for each variable we notice that variables Age, Insulin and Diabetes Pedigree Function have high skewed distribution (greater than 1). So, we applied the method of Box cox trans to fix them:

Table 4: Level of skewness for each variable

Pregnancies	Glucose	BloodPressure	SkinThickness
0.8981549	0.5317852	0.1538784	0.6832310
Insulin	**BMI**	**DiabetesPedigreeFunction**	**Age**
2.2504549	0.6048157	1.9124179	1.1251880

F. Balancing the data

After checking the summary of the data, we notice that the distribution of the binary variable outcome is imbalanced. In other words, there are more healthy women(non-diabetic) than the diabetic women (268 out of 768 women). The imbalanced outcome may cause problems during analysis like “Accuracy Paradox” and can misclassify some observations into the majority class. Therefore, a resampling method should be applied to our data before fitting the prediction models.

We used the upsampling technique, which consists of sampling more data points for the minority class to even out the distribution of the classes and avoid misleading model accuracy. The function upSample() from the caret package randomly samples the data with replacement to stabilize the distribution of the classes. After reinforcing the signal of the minority class in the diagnosis variable, the dataset consists of 1000 observations as opposed to the 768 from the original dataset. Before conducting the prediction methods, I divided my data into two subsets: train and test data. Train set constitutes 75% of the total data.

IV. Exploring different machine learning techniques

A. Logistic Regression:

It is interesting to fit a logistic regression model on the data since the outcome variable is categorical. We started by fitting the model on the full data. This method shows that 4 of 8 predictors make a significant contribution to the prediction model. The best variables that predict diabetes are: Number of Pregnancies, Glucose level, BMI and Diabetes Pedigree Function. The smallest p-value is

associated with the Glucose level and BMI, which makes sense because glucose level is the main factor that determines diabetes and a higher BMI means that a person is obese, and obesity leads to diabetes. The AIC of the model is 763.32 and AUC=0.850. Error rate: 27%. That means that almost 27% of the Pima Indian women got misclassified of having diabetes or not.

Now we use the Stepwise exhaustive selection based on the AIC to find the best predictors for diabetes. We remove the variables that appear not to be helpful in predicting diabetes to obtain a more effective model. The best model according to this selection method is the model that includes: Number of Pregnancies, Glucose Level, Age, BMI, Diabetes Pedigree Function and Blood Pressure. In this model with only four independent variables, we got the same classification error rate but lower AIC (745.5169) same AUC (0.8542), which means that the second model with less variables is slightly better according to its AIC.

Finally, we tried the best glm function to find what are the best models. We fitted the model on our train data and predicted the outcome on the test data. We got five different best models. The best model among these five is model 5 with lowest error rate (23%) and lowest AIC (745.52) and AUC (0.854).

B. LASSO:

We will now use the shrinkage technique to find the best subset of predictor variables. LASSO estimates the parameters by optimizing the binomial likelihood with the respect of some penalty on the parameters. We run LASSO on the train set, and we use the Cross-Validation method to find the best lambda which minimizes the classification error rate. We can also find other best lambdas that minimize the binomial deviance and maximize AUC. Then we use this best lambda to rerun the LASSO on our test set. We get error rate 25.6%, greater than the error we found with the subset selection above.

C. Ridge Regression:

Similarly, we will apply the ridge regression to shrink our coefficients. Ridge regression puts constraints on the coefficients (w). The penalty term (lambda) regularizes the coefficients such that if the coefficients take large values the optimization function is penalized. So, ridge regression shrinks the coefficients, and

it helps to reduce the model complexity and multi-collinearity. The classification error rate with this method is 25.2%.

D. Linear Discriminant Analysis LDA:

The Linear Discriminant Analysis is a method that is used to find a linear combination of features that characterizes or separates two or more classes of objects or events. It assumes that all the independent variables follow a multivariate distribution with a specific mean and similar covariance matrix. Fitting the model on our test set, we get an error rate that is higher than the one we got in Logistic Regression (25.6%). The LDA summary shows that the 4 most important variables that predict Diabetes are: Glucose level, Number of Pregnancies, BMI and Diabetes Pedigree function.

E. K-Nearest Neighbors:

KNN or K-Nearest Neighbors is a non-linear way to predict an outcome based on the Euclidian distance between a test point and its nearest K neighbors. Though this method allows a more flexible and non-linear boundary line, and therefore a higher prediction accuracy, it does not provide any information about whether specific factors are significant or impactful in any way. Choosing the value of K is the most important, the optimal K will balance the bias-variance tradeoff, that is for lower values of K the training prediction accuracy will be higher, therefore, a lower bias, but the variance will be considerable.

After analyzing the KNN prediction for K=1,5 and 10, we see that the classification error rate decreases with K. Using Cross validation, values for K = 2 through 15 are analyzed. We find that the test classification error is minimized at K = 4. The test classification error rate at K=4 is 19.6% which is much lower than logistic regression (both subset selection and shrinkage methods). If the goal of analyzing the data is solely to improve prediction accuracy, KNN is a suitable method.

F. Decision Tree:

1. Cross-Validation

The Decision Tree method does not predict a specific model. But it builds a tree that is easy to interpret by dividing the predictor space into j distinct region that do

not overlap. Each observation that falls in a specific region will be classified to the majority class of this region. The tree is built in a way that minimizes the error rate. Creating the decision tree, we got an error rate of 22%. But the complete tree is so complex that it cannot be interpretable and because it is so specific it is possible that it will overfit the data. Pruning is best in this case to avoid this problem. We use cross validation to obtain the best tuning parameter.

The 10-fold cross validation method shows that for a tree of size 9, we will get the minimum error rate 24.8%. The variables included in the tree are: Glucose, Insulin, BMI, and Age.

2. Bagging

As we know that the decision tree suffers from a high variance. The Bagging method helps to reduce the variance of a decision tree by averaging (reduce variance). Applying the Bagging method gives us the lowest error rate 14.4%. and it indicates what are the most helpful variables that affect the outcome. For instance, we can see that the variables with the mean decrease Gini are Glucose level and BMI.

3. Random Forest

Random forest builds on the idea of bagging, but it provides an improvement because it decorrelates the trees. It will force each split to consider only one subset of the predictors. This method gives us an error rate that is also very low, 14.4%. But it shows that the most important predictors to decrease the mean Gini are Glucose and Insulin levels.

G. Support Vector Machine

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. It uses C as a budget for the amount that the margin can be violated by the n observations. C is treated as a tuning parameter that is generally chosen via cross-validation. From 10 folds CV we get the best cost c=10 and best gamma=0.5. We get misclassification error rate=15.6%.

V. Conclusion

After applying and comparing all these prediction methods we can conclude that the bagging and random forest are the best at giving the minimum classification error rate 14.4%. Hence, if our main goal is to choose the method with the highest accuracy, we would choose tree-based methods, KNN or Support Vectors Machine. However, the regression methods such as Logistic regression, Linear discriminant analysis LDA, LASSO and Ridge regression would give us models that can help to apply on other data sets in the future.

VI. References

Kaggle (2018). Pima Indians diabetes database. https://www.kaggle.com/uciml/pima-indians-diabetes-database

National Institute of Diabetes and Digesting and Kidney Diseases. (2016, December). What is diabetes? Diabetes Overview. https://www.niddk.nih.gov/health-information/diabetes/overview/what-is-diabetes

VII. Appendix 1: Data visualization in R

```
#Importing the necessary libraries
library(caret)

library(Hmisc)

library(DMwR)

library(corrplot)

library(e1071)

library(beeswarm)
library(dplyr)

library(ggplot2)
library(psych)

#Loading the dataset
diabetes <- read.csv("C:/Users/safaa/OneDrive/Desktop/Math748/data/diabetes.c
sv")

#quick look on the data
head(diabetes) #top five rows
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       0 25.6
##   DiabetesPedigreeFunction Age Outcome
## 1                    0.627  50       1
## 2                    0.351  31       0
## 3                    0.672  32       1
## 4                    0.167  21       0
## 5                    2.288  33       1
## 6                    0.201  30       0
```

```
names(diabetes) #variables name
```

```
## [1] "Pregnancies"              "Glucose"
## [3] "BloodPressure"            "SkinThickness"
## [5] "Insulin"                  "BMI"
## [7] "DiabetesPedigreeFunction" "Age"
## [9] "Outcome"
```

```
dim(diabetes) # number of rows, number of columns
```

```
## [1] 768   9
```

```
summary(diabetes)
```

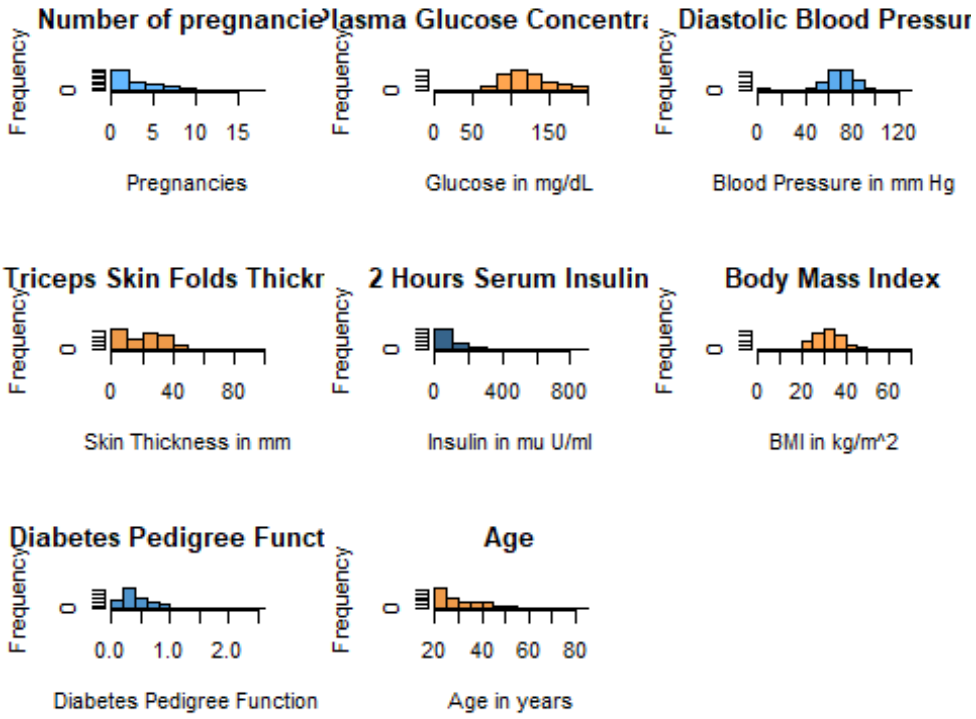
```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
table(diabetes$Outcome) #frequency table for one categorical variable
```

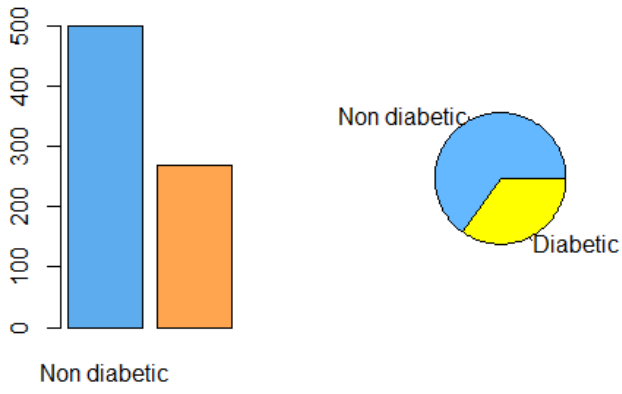
```
##
##   0   1
## 500 268
```

```
#visualization
#Histogram of single variable
par(mfrow = c(3, 3))
hist(diabetes$Pregnancies,col="steelblue1", xlab="Pregnancies", main="Number
of pregnancies")
hist(diabetes$Glucose,col="tan1", xlab="Glucose in mg/dL", main="Plasma Gluco
se Concentration")
hist(diabetes$BloodPressure,col="steelblue2", xlab="Blood Pressure in mm Hg",
main="Diastolic Blood Pressure")
hist(diabetes$SkinThickness,col="tan2", xlab="Skin Thickness in mm", main="Tr
iceps Skin Folds Thickness")
hist(diabetes$Insulin,col="steelblue4", xlab="Insulin in mu U/ml", main="2 Ho
urs Serum Insulin")
hist(diabetes$BMI,col="tan1", xlab="BMI in kg/m^2", main="Body Mass Index")
hist(diabetes$DiabetesPedigreeFunction,col="steelblue3", xlab="Diabetes Pedig
ree Function", main="Diabetes Pedigree Function")
hist(diabetes$Age,col="tan2", xlab="Age in years", main="Age")
```

```
diabetes$Outcome = as.factor(diabetes$Outcome)
diabetes$Outcome = factor(diabetes$Outcome,labels = c('Non diabetic','Diabeti
c'))
par(mfrow = c(1, 2))
```

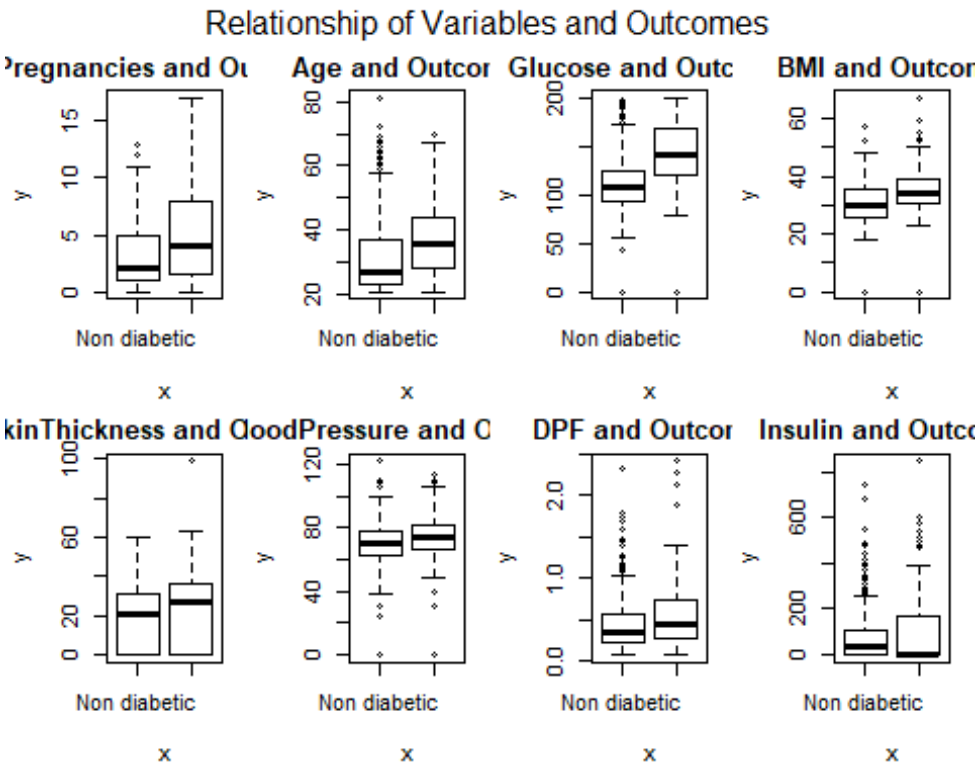



```
plot(diabetes$Outcome, col=c("steelblue2", "tan1"))
d=table(diabetes$Outcome)
pie(d,col=c("steelblue1", "yellow"))
```



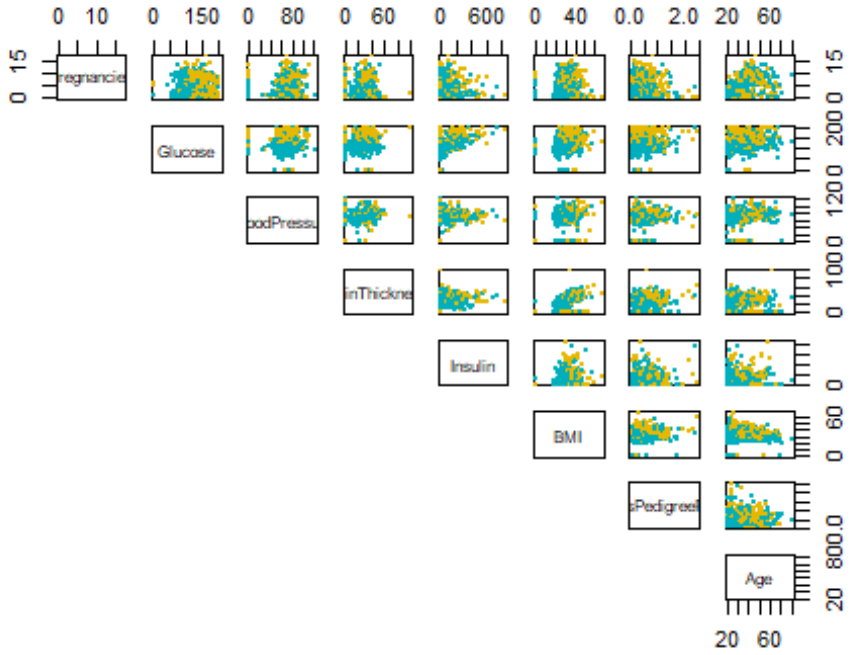
```
piepercent<- round(100*d/sum(d), 1)
pie(d, labels=piepercent, main = "Percentage of diabetic women",col=c("steelb
lue2", "tan1"))
legend("topright", c("Non Diabetic","Diabetic"), cex = 0.8,
       fill =c("steelblue2", "tan1"))

#Boxplots
par(mfrow = c(2, 4), mar = c(4, 4, 2, 1), oma = c(0, 0, 2, 0)) # 2 rows 4 col
umns
```

```
with(diabetes, {
  plot(Outcome, Pregnancies, main = "Pregnancies and Outcome")
  plot(Outcome, Age, main = "Age and Outcome")
  plot(Outcome, Glucose, main = "Glucose and Outcome")
  plot(Outcome, BMI, main = "BMI and Outcome")
  plot(Outcome, SkinThickness, main = "SkinThickness and Outcome")
  plot(Outcome, BloodPressure, main = "BloodPressure and Outcome")
  plot(Outcome, DiabetesPedigreeFunction, main = "DPF and Outcome")
  plot(Outcome, Insulin, main = "Insulin and Outcome")
  mtext("Relationship of Variables and Outcomes", outer = T)
})
```




```
#Matrix of Scatterplot
my_cols <- c("#00AFBB", "#E7B800", "#FC4E07")
pairs(diabetes[,1:8], pch=19, cex=0.5,
```

```
      col=my_cols[diabetes$Outcome],
      lower.panel=NULL)
```




```
#Transform zero to NA
diabetes$Glucose[diabetes$Glucose==0]=NA
diabetes$BloodPressure[diabetes$BloodPressure==0]=NA
diabetes$SkinThickness[diabetes$SkinThickness==0]=NA
diabetes$Insulin[diabetes$Insulin==0]=NA
diabetes$BMI[diabetes$BMI==0]=NA
summary(diabetes)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.:22.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :29.00
##  Mean   : 3.845   Mean   :121.7   Mean   : 72.41   Mean   :29.15
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:36.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##                   NA's   :5       NA's   :35       NA's   :227
##     Insulin            BMI        DiabetesPedigreeFunction      Age
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0780           Min.   :21.00
##  1st Qu.: 76.25   1st Qu.:27.50   1st Qu.:0.2437           1st Qu.:24.00
##  Median :125.00   Median :32.30   Median :0.3725           Median :29.00
##  Mean   :155.55   Mean   :32.46   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:190.00   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##  NA's   :374      NA's   :11
```

```
##          Outcome
##  Non diabetic:500
##  Diabetic    :268
##
##
##
##
##
```

```
sum(is.na(diabetes))
```

```
## [1] 652
```

```
#prepare the data for learning algorithms.

#Missing value
#total number of missing:
summary(diabetes)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.:22.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :29.00
##  Mean   : 3.845   Mean   :121.7   Mean   : 72.41   Mean   :29.15
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:36.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##                   NA's   :5       NA's   :35       NA's   :227
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0780           Min.   :21.00
##  1st Qu.: 76.25   1st Qu.:27.50   1st Qu.:0.2437           1st Qu.:24.00
##  Median :125.00   Median :32.30   Median :0.3725           Median :29.00
##  Mean   :155.55   Mean   :32.46   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:190.00   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##  NA's   :374      NA's   :11
##          Outcome
##  Non diabetic:500
##  Diabetic    :268
##
##
##
##
##
```

```
sum(is.na(diabetes))
```

```
## [1] 652
```

```
imp.knn.all <- knnImputation(diabetes[,!names(diabetes) %in% "Outcome"])
summary(imp.knn.all)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.:22.31
##  Median : 3.000   Median :117.0   Median : 72.00   Median :29.00
##  Mean   : 3.845   Mean   :121.7   Mean   : 72.31   Mean   :29.03
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:35.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin            BMI        DiabetesPedigreeFunction      Age
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0780           Min.   :21.00
##  1st Qu.: 91.32   1st Qu.:27.50   1st Qu.:0.2437           1st Qu.:24.00
##  Median :135.57   Median :32.15   Median :0.3725           Median :29.00
##  Mean   :153.88   Mean   :32.43   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:189.82   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200           Max.   :81.00
```

```
sum(is.na(imp.knn.all))
```

```
## [1] 0
```

```
#correlation matrix after removing missing values
cor(imp.knn.all[,1:8],use="pairwise.complete.obs")
```

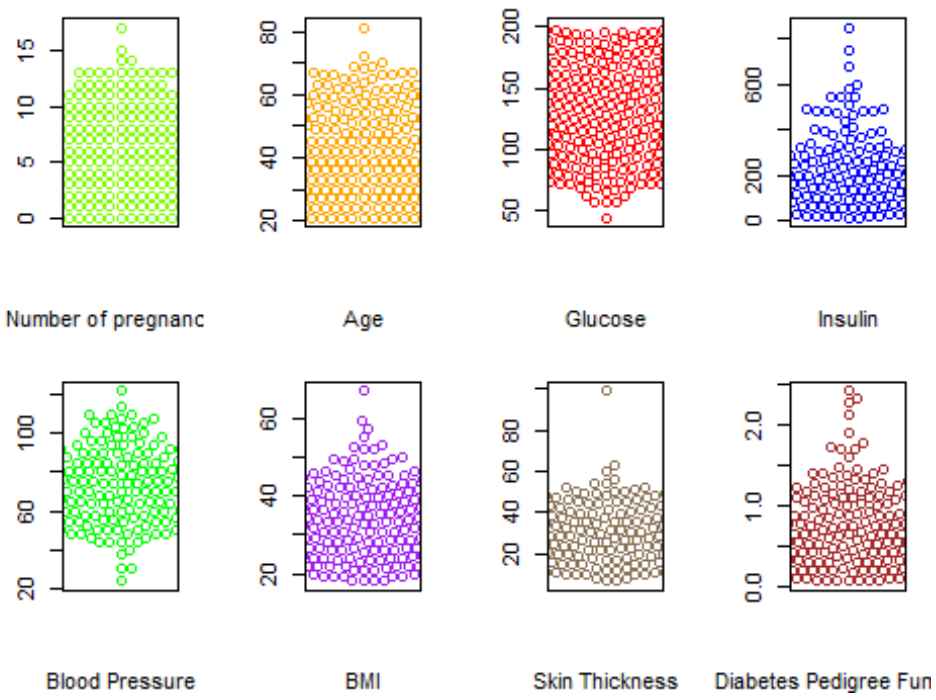
```
##                           Pregnancies   Glucose BloodPressure SkinThickness
## Pregnancies                1.00000000 0.1300741  0.2140764696     0.1320222
## Glucose                    0.13007413 1.0000000  0.2289397099     0.2317676
## BloodPressure              0.21407647 0.2289397  1.0000000000     0.2269567
## SkinThickness              0.13202219 0.2317676  0.2269567112     1.0000000
## Insulin                    0.10082471 0.6129553  0.1549335461     0.2515871
## BMI                        0.02209358 0.2368455  0.2944102934     0.6648612
## DiabetesPedigreeFunction -0.03352267 0.1384564 -0.0008954921     0.1290416
## Age                        0.54434123 0.2688504  0.3336544468     0.1606280
##                              Insulin        BMI DiabetesPedigreeFunction
## Pregnancies                0.1008247 0.02209358            -0.0335226730
## Glucose                    0.6129553 0.23684548             0.1384564487
## BloodPressure              0.1549335 0.29441029            -0.0008954921
## SkinThickness              0.2515871 0.66486115             0.1290415648
## Insulin                    1.0000000 0.28764155             0.1550272700
## BMI                        0.2876415 1.00000000             0.1535030251
## DiabetesPedigreeFunction 0.1550273 0.15350303             1.0000000000
## Age                        0.2763533 0.02718476             0.0335613124
##                                  Age
## Pregnancies              0.54434123
## Glucose                  0.26885045
## BloodPressure            0.33365445
## SkinThickness            0.16062798
## Insulin                  0.27635325
## BMI                      0.02718476
## DiabetesPedigreeFunction 0.03356131
## Age                      1.00000000
```

```
#Matrix of scatterplot with coefficients of correlation
pairs.panels(imp.knn.all[,-9],
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
)
```


```
cor.matrix <- cor(imp.knn.all[,1:8],use="pairwise.complete.obs")
corrplot(cor.matrix, order="hclust")

## Warning in corrplot(cor.matrix, order = "hclust"): Not been able to calcul
ate
## text margin, please try again with a clean new empty window using {plot.ne
w();
## dev.off()} or reduce tl.cex

#High correlation
highCorr <- findCorrelation(cor.matrix,cutoff=0.75)
length(highCorr)

## [1] 0

# Near Zero variance
nearZeroVar(diabetes)

## integer(0)
```

```
par(mfrow = c(2,4))
```

```
beeswarm(imp.knn.all$Pregnancies, xlab="Number of pregnancies", col="chartreuse")
beeswarm(imp.knn.all$Age, xlab="Age", col="orange")
beeswarm(imp.knn.all$Glucose, xlab="Glucose", col="red")
beeswarm(imp.knn.all$Insulin, xlab="Insulin", col="blue")
beeswarm(imp.knn.all$BloodPressure, xlab="Blood Pressure", col="green")
beeswarm(imp.knn.all$BMI, xlab="BMI", col="purple")
beeswarm(imp.knn.all$SkinThickness, xlab="Skin Thickness", col="burlywood4")
beeswarm(imp.knn.all$DiabetesPedigreeFunction, xlab="Diabetes Pedigree Function", col="brown")
```




```
#skewness


skew.all <- apply(imp.knn.all[,1:8],2,function(x) skewness(x,na.rm=T))
skew.all
```

```
##              Pregnancies                  Glucose            BloodPressure
##                0.8981549                0.5317852                0.1538784
##            SkinThickness                  Insulin                      BMI
##                0.6832310                2.2504549                0.6048157
## DiabetesPedigreeFunction                      Age
##                1.9124179                1.1251880
```

```r
trans <- BoxCoxTrans(imp.knn.all$Age)
head(imp.knn.all$Age) #

## [1] 50 31 32 21 33 30

diabetes.new <- predict(trans,head(imp.knn.all$Age))
skewness(diabetes.new)

## [1] -0.4192027

trans <- BoxCoxTrans(imp.knn.all$Insulin)
head(imp.knn.all$Insulin) #

## [1] 254.18622  71.23043 204.61273  94.00000 168.00000 106.27184

diabetes.new <- predict(trans,head(imp.knn.all$Insulin))
skewness(diabetes.new)

## [1] -0.004154655

trans <- BoxCoxTrans(imp.knn.all$DiabetesPedigreeFunction)
head(imp.knn.all$DiabetesPedigreeFunction) #

## [1] 0.627 0.351 0.672 0.167 2.288 0.201

diabetes.new <- predict(trans,head(imp.knn.all$DiabetesPedigreeFunction))
skewness(diabetes.new)

## [1] 0.4105096
```

VIII. Appendix 2: Models Implementation in R

```
###### Libraries and fucntions #######
library(tree)

## Warning: package 'tree' was built under R version 3.6.3

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.6.3

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

library(readxl)

## Warning: package 'readxl' was built under R version 3.6.3

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.6.3

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

library(lattice)
library(leaps)

## Warning: package 'leaps' was built under R version 3.6.3

library(boot)

##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma

library(bestglm)

## Warning: package 'bestglm' was built under R version 3.6.3

library(dummies)

## dummies-1.5.6 provided by Decision Patterns

library(MASS)
library(DAAG)
```

```
## Warning: package 'DAAG' was built under R version 3.6.3

##
## Attaching package: 'DAAG'

## The following object is masked from 'package:MASS':
##
##     hills

library(caret)

## Warning: package 'caret' was built under R version 3.6.3

library(glmnet)

## Warning: package 'glmnet' was built under R version 3.6.3

## Loading required package: Matrix

## Loaded glmnet 4.0-2

library(plotROC)

## Warning: package 'plotROC' was built under R version 3.6.3

library(ROCR)

## Warning: package 'ROCR' was built under R version 3.6.3

library(pROC)

## Warning: package 'pROC' was built under R version 3.6.3

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following object is masked from 'package:plotROC':
##
##     ggroc

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(class)
library(splines)
library(Hmisc)

## Warning: package 'Hmisc' was built under R version 3.6.3

## Loading required package: survival
```

```
##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## The following object is masked from 'package:DAAG':
##
##     lung

## The following object is masked from 'package:boot':
##
##     aml

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

library(DMwR)

## Warning: package 'DMwR' was built under R version 3.6.3

## Loading required package: grid

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

library(corrplot)

## Warning: package 'corrplot' was built under R version 3.6.3

## corrplot 0.84 loaded

library(e1071)

## Warning: package 'e1071' was built under R version 3.6.3

##
## Attaching package: 'e1071'

## The following object is masked from 'package:Hmisc':
##
##     impute

#Loading the dataset
diabetes <- read.csv("C:/Users/safaa/OneDrive/Desktop/Math748/data/diabetes.c
sv")
```

```r
#transform 0 to NA
diabetes$Glucose[diabetes$Glucose==0]=NA
diabetes$BloodPressure[diabetes$BloodPressure==0]=NA
diabetes$SkinThickness[diabetes$SkinThickness==0]=NA
diabetes$Insulin[diabetes$Insulin==0]=NA
diabetes$BMI[diabetes$BMI==0]=NA
sum(is.na(diabetes))
```

```
## [1] 652
```

```r
#transform NA to nearest value
imp.knn.all <- knnImputation(diabetes[,!names(diabetes) %in% "Outcome"])
sum(is.na(imp.knn.all))
```

```
## [1] 0
```

```r
diabetes[,1:8]=imp.knn.all

#changing the variable Outcome to factors with 2 levels
y<-as.factor(diabetes$Outcome)
diab <- data.frame(diabetes[,1:8], y)
#checking the summary of the data
summary(diab) #Diag is unbalanced
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.:22.31
##  Median : 3.000   Median :117.0   Median : 72.00   Median :29.00
##  Mean   : 3.845   Mean   :121.7   Mean   : 72.31   Mean   :29.03
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:35.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0780           Min.   :21.00
##  1st Qu.: 91.32   1st Qu.:27.50   1st Qu.:0.2437           1st Qu.:24.00
##  Median :135.57   Median :32.15   Median :0.3725           Median :29.00
##  Mean   :153.88   Mean   :32.43   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:189.82   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##  y
##  0:500
##  1:268
##
##
##
##
```

```r
#Exploring Statistical Learning Methods
set.seed(447)
#setting up data frame to use bestglm (also to use for upsampling )
X=diab
```

```r
y=X$y#creating y vector and matrix of predictor vars
X$y=NULL
Xy<-data.frame(X, y=y)
summary(Xy) #the data is imbalanced
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.:22.31
##  Median : 3.000   Median :117.0   Median : 72.00   Median :29.00
##  Mean   : 3.845   Mean   :121.7   Mean   : 72.31   Mean   :29.03
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:35.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin            BMI        DiabetesPedigreeFunction      Age
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0780           Min.   :21.00
##  1st Qu.: 91.32   1st Qu.:27.50   1st Qu.:0.2437           1st Qu.:24.00
##  Median :135.57   Median :32.15   Median :0.3725           Median :29.00
##  Mean   :153.88   Mean   :32.43   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:189.82   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##  y
##  0:500
##  1:268
##
##
##
##
```

```r
#Using UpSample to balance the class in response variable
diab.balance<-upSample(x=X,y=y) #now 1000 observations
summary(diab.balance) #Diag is now balanced 50/50
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.:102.0   1st Qu.: 65.00   1st Qu.:23.00
##  Median : 3.000   Median :123.0   Median : 72.00   Median :30.00
##  Mean   : 4.071   Mean   :126.7   Mean   : 72.81   Mean   :29.64
##  3rd Qu.: 6.000   3rd Qu.:148.0   3rd Qu.: 80.00   3rd Qu.:35.09
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin            BMI        DiabetesPedigreeFunction      Age
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0780           Min.   :21.00
##  1st Qu.: 99.96   1st Qu.:28.19   1st Qu.:0.2457           1st Qu.:25.00
##  Median :145.74   Median :32.80   Median :0.3830           Median :31.00
##  Mean   :164.20   Mean   :33.08   Mean   :0.4863           Mean   :34.04
##  3rd Qu.:200.36   3rd Qu.:37.12   3rd Qu.:0.6542           3rd Qu.:42.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##  Class
##  0:500
##  1:500
##
##
```

```
##
##

diab.balance=within(diab.balance, {y = Class}) #Upsample function changed the
name of outcome - changing it back
diab.balance=diab.balance[,-c(9)]

#######################################
# Data Analysis #
#######################################
#Starting with Logistic Regression- using AIC to choose best subset

# setting a training and test set (75-25 split )
set.seed(447)
smp_size <- floor(0.75 * nrow(diab.balance))
train_ind <- sample(seq_len(nrow(diab.balance)), size = smp_size)
train <- diab.balance[train_ind, ] #training set 750 obs
test <- diab.balance[-train_ind, ] #test set 250 obs
dim(test)

## [1] 250   9

#full model with all levels
glmfit<-glm(y~., family = "binomial", data = train)
summary(glmfit) #PRILIM. keep: pregnancies, glucose, BMI, diabetes pedigree f
unction and age

##
## Call:
## glm(formula = y ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9590  -0.7610   0.1994   0.7208   2.2361
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -9.4479783  0.8698754 -10.861  < 2e-16 ***
## Pregnancies               0.1460881  0.0327681   4.458 8.26e-06 ***
## Glucose                   0.0402883  0.0044139   9.127  < 2e-16 ***
## BloodPressure            -0.0102760  0.0084228  -1.220  0.22246
## SkinThickness            -0.0203866  0.0142541  -1.430  0.15265
## Insulin                  -0.0008398  0.0012879  -0.652  0.51435
## BMI                       0.1248839  0.0210243   5.940 2.85e-09 ***
## DiabetesPedigreeFunction  0.8486147  0.2957762   2.869  0.00412 **
## Age                       0.0235053  0.0094427   2.489  0.01280 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 1039.19  on 749  degrees of freedom
## Residual deviance:  728.02  on 741  degrees of freedom
## AIC: 746.02
##
## Number of Fisher Scoring iterations: 5
```

```
pred<-predict(glmfit,test,type ="response")
glm.pred=rep("0",250)
glm.pred[pred>.5]="1"
table(glm.pred,test$y)
```

```
##
## glm.pred   0   1
##        0 106  29
##        1  29  86
```

```
mean(glm.pred!=test$y) #error rate 0.232
```

```
## [1] 0.232
```

```
out<-bestglm(train, IC = "AIC", family = binomial)
```

```
## Morgan-Tatar search since family is non-gaussian.
```

```
out$BestModels
```

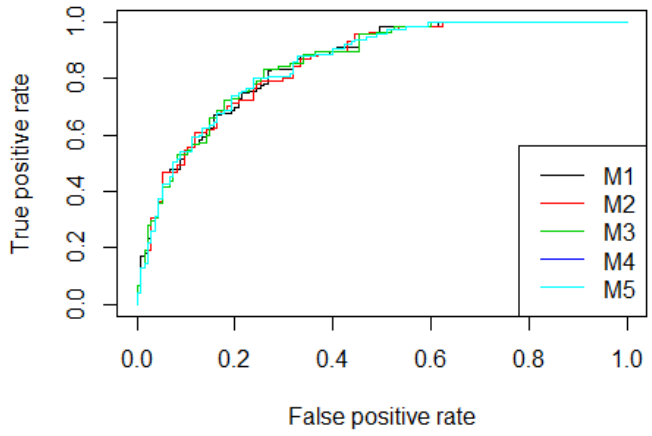
```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1        TRUE    TRUE         FALSE         FALSE   FALSE TRUE
## 2        TRUE    TRUE         FALSE          TRUE   FALSE TRUE
## 3        TRUE    TRUE          TRUE         FALSE   FALSE TRUE
## 4        TRUE    TRUE          TRUE          TRUE   FALSE TRUE
## 5        TRUE    TRUE         FALSE          TRUE    TRUE TRUE
##   DiabetesPedigreeFunction  Age Criterion
## 1                     TRUE TRUE  741.8305
## 2                     TRUE TRUE  741.8570
## 3                     TRUE TRUE  742.3878
## 4                     TRUE TRUE  742.4377
## 5                     TRUE TRUE  743.5169
```

```
#best 5 models
mod1=glm(y~Pregnancies+Glucose+BMI+DiabetesPedigreeFunction+Age, family="bino
mial", data=train)
mod2=glm(y~Pregnancies + Glucose+BMI + DiabetesPedigreeFunction + Age + SkinT
hickness, family="binomial", data=train)
mod3=glm(y~Pregnancies+Glucose+BloodPressure+BMI+DiabetesPedigreeFunction+Age
, family="binomial", data=train)
mod4=glm(y~Pregnancies+Glucose+BloodPressure+SkinThickness+BMI+DiabetesPedigr
eeFunction+Age, family="binomial", data=train)
mod5=glm(y~Pregnancies+Glucose+BMI+DiabetesPedigreeFunction+Age+SkinThickness
+Insulin, family="binomial", data=train)

#generating error rate in model 1
```

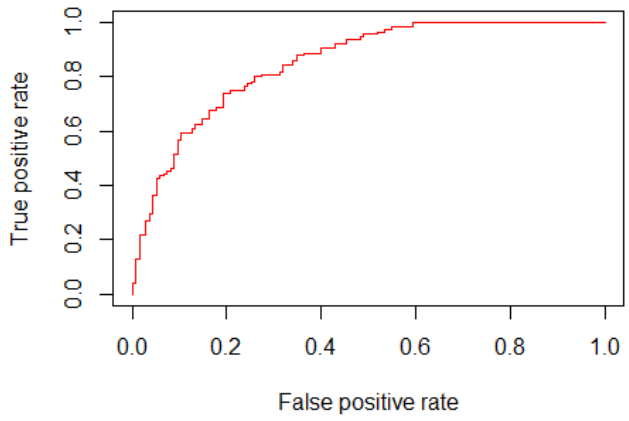
```
pred.mod1<-predict.glm(mod1, test, type= "response")
glm.pred=rep("0",250)
glm.pred[pred.mod1>.5]="1"
table(glm.pred,test$y)

##
## glm.pred   0   1
##        0 103  28
##        1  32  87

mod1.err<-mean(glm.pred!=test$y) #error rate in test data 0.24
mod1.aic<-summary(mod1)$"aic" #AIC 743.8
ROCRpred.mod1<-prediction(pred.mod1, test$y)
ROCRperf.mod1<-performance(ROCRpred.mod1, 'tpr','fpr') #ROC Curve
auc.mod1 = as.data.frame(performance(ROCRpred.mod1, 'auc')@y.values) #AUC=0.8
54

#generating error rate in model 2
pred.mod2<-predict.glm(mod2, test, type= "response")
glm.pred=rep("0",250)
glm.pred[pred.mod2>.5]="1"
table(glm.pred,test$y)

##
## glm.pred   0   1
##        0 103  29
##        1  32  86

mod2.err<-mean(glm.pred!=test$y) #error rate in test data 0.244
mod2.aic<-summary(mod2)$"aic" #743.857
ROCRpred.mod2<-prediction(pred.mod2, test$y)
ROCRperf.mod2<-performance(ROCRpred.mod2, 'tpr','fpr') #ROC Curve
auc.mod2 = as.data.frame(performance(ROCRpred.mod2, 'auc')@y.values) #AUC 0.8
50

#generating error rate in model 3
pred.mod3<-predict.glm(mod3, test, type= "response")
glm.pred=rep("0",250)
glm.pred[pred.mod3>.5]="1"
table(glm.pred,test$y)

##
## glm.pred   0   1
##        0 104  28
##        1  31  87

mod3.err<-mean(glm.pred!=test$y) #error rate in test data 0.236
mod3.aic<-summary(mod3)$"aic" #AIC 744.38
ROCRpred.mod3<-prediction(pred.mod3, test$y)
ROCRperf.mod3<-performance(ROCRpred.mod3, 'tpr','fpr') #ROC Curve
auc.mod3 = as.data.frame(performance(ROCRpred.mod3, 'auc')@y.values) # AUC=0.
```

```
856

#generating error rate in model 4
pred.mod4<-predict.glm(mod4, test, type= "response")
glm.pred=rep("0",250)
glm.pred[pred.mod4>.5]="1"
table(glm.pred,test$y)

##
## glm.pred   0   1
##        0 106  29
##        1  29  86

mod4.err<-mean(glm.pred!=test$y) #error rate in test data 0.232
mod4.aic<-summary(mod4)$"aic" #AIC 744.43
ROCRpred.mod4<-prediction(pred.mod4, test$y)
ROCRperf.mod4<-performance(ROCRpred.mod4, 'tpr','fpr') #ROC Curve
auc.mod4 = as.data.frame(performance(ROCRpred.mod4, 'auc')@y.values) # AUC 0.
854

#generating error rate in model 5
pred.mod5<-predict.glm(mod4, test, type= "response")
glm.pred=rep("0",250)
glm.pred[pred.mod5>.5]="1"
table(glm.pred,test$y)

##
## glm.pred   0   1
##        0 106  29
##        1  29  86

mod5.err<-mean(glm.pred!=test$y) #error rate in test data 0.232
mod5.aic<-summary(mod5)$"aic" #AIC 745.52
ROCRpred.mod5<-prediction(pred.mod5, test$y)
ROCRperf.mod5<-performance(ROCRpred.mod5, 'tpr','fpr') #ROC Curve
auc.mod5 = as.data.frame(performance(ROCRpred.mod5, 'auc')@y.values) # AUC 0.
854

#creating a table of classifcation error rate,
model.err<-data.frame(error_rate=c(mod1.err,mod2.err,mod3.err,mod4.err,mod5.e
rr),
             aic=c(mod1.aic,mod2.aic,mod3.aic,mod4.aic,mod5.aic),
             AUC = c(auc.mod1[1,1],auc.mod2[1,1],auc.mod3[1,1],auc.mod4[1,1],
auc.mod5[1,1]))
rownames(model.err)<-c("Model 1","Model 2","Model 3","Model 4","Model 5")
#model 5 has low error rate with highest AIC and highest AUC
#creating a plot of the ROC curves for the models.
plot(ROCRperf.mod1,col=1)
plot(ROCRperf.mod2,col= 2, add = TRUE)
plot(ROCRperf.mod3,col= 3, add = TRUE)
plot(ROCRperf.mod4,col= 4, add = TRUE)
```

```
plot(ROCRperf.mod5,col= 5, add = TRUE)
legend("bottomright", c("M1","M2","M3","M4","M5"), lty=1,
       col = c(1:5), bty="o")
```




```
model.err

##         error_rate      aic       AUC
## Model 1      0.240 743.8305 0.8542351
## Model 2      0.244 743.8570 0.8508857
## Model 3      0.236 744.3878 0.8559098
## Model 4      0.232 744.4377 0.8542351
## Model 5      0.232 745.5169 0.8542351

#GRAPGHING
ROCRpred<-prediction(pred, test$y)
ROCRperf<-performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf,col="red",text.adj = c(-0.2,1.7))
```



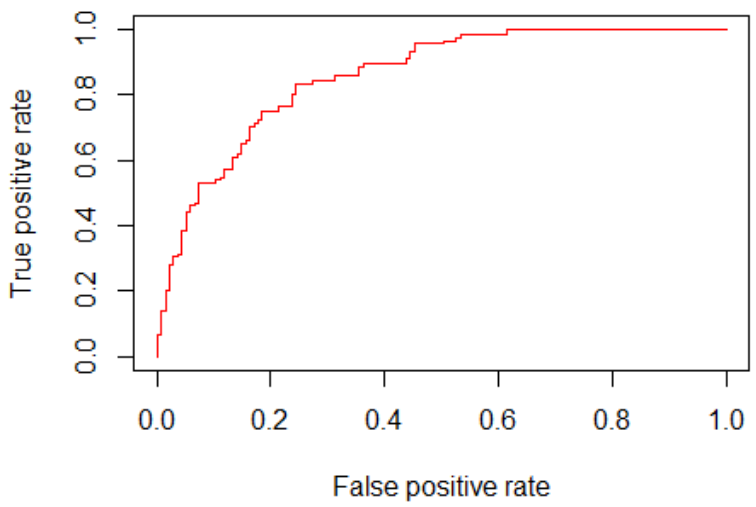
```r
auc = performance(ROCRpred, 'auc')@y.values # 0.848

#using STEP AIC to choose model
stepAIC(glmfit, direction = "both")
```

```
## Start:  AIC=746.02
## y ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin +
##     BMI + DiabetesPedigreeFunction + Age
##
##                           Df Deviance    AIC
## - Insulin                  1   728.44 744.44
## - BloodPressure            1   729.52 745.52
## <none>                         728.02 746.02
## - SkinThickness            1   730.03 746.03
## - Age                      1   734.28 750.28
## - DiabetesPedigreeFunction 1   736.65 752.65
## - Pregnancies              1   748.96 764.96
## - BMI                      1   766.80 782.80
## - Glucose                  1   834.22 850.22
##
## Step:  AIC=744.44
## y ~ Pregnancies + Glucose + BloodPressure + SkinThickness + BMI +
##     DiabetesPedigreeFunction + Age
##
##                           Df Deviance    AIC
## - BloodPressure            1   729.86 743.86
## - SkinThickness            1   730.39 744.39
## <none>                         728.44 744.44
## + Insulin                  1   728.02 746.02
## - Age                      1   734.33 748.33
## - DiabetesPedigreeFunction 1   736.88 750.88
## - Pregnancies              1   749.31 763.31
## - BMI                      1   767.02 781.02
## - Glucose                  1   871.74 885.74
##
## Step:  AIC=743.86
## y ~ Pregnancies + Glucose + SkinThickness + BMI + DiabetesPedigreeFunction
+
##     Age
##
##                           Df Deviance    AIC
## - SkinThickness            1   731.83 743.83
## <none>                         729.86 743.86
## + BloodPressure            1   728.44 744.44
## + Insulin                  1   729.52 745.52
## - Age                      1   734.62 746.62
## - DiabetesPedigreeFunction 1   739.41 751.41
## - Pregnancies              1   750.74 762.74
## - BMI                      1   767.03 779.03
## - Glucose                  1   871.82 883.82
```

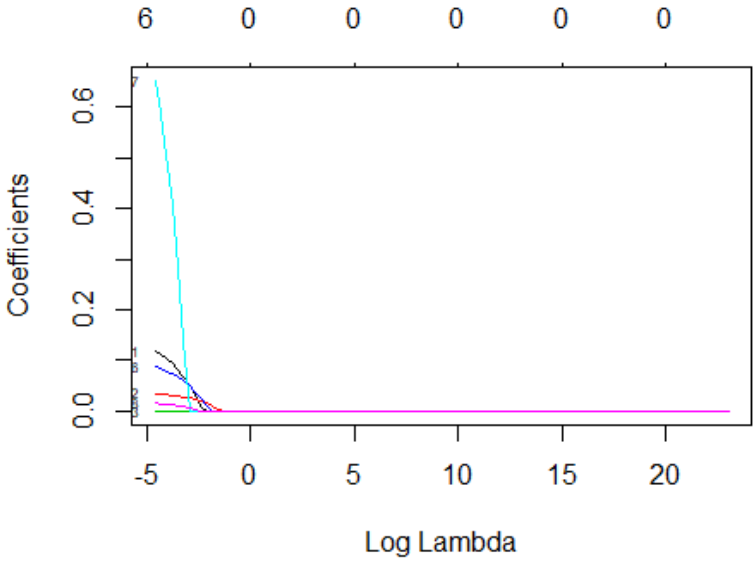
```
##
## Step:  AIC=743.83
## y ~ Pregnancies + Glucose + BMI + DiabetesPedigreeFunction +
##     Age
##
##                            Df Deviance    AIC
## <none>                          731.83 743.83
## + SkinThickness             1   729.86 743.86
## + BloodPressure             1   730.39 744.39
## + Insulin                   1   731.54 745.54
## - Age                       1   736.21 746.21
## - DiabetesPedigreeFunction  1   741.51 751.51
## - Pregnancies               1   751.52 761.52
## - BMI                       1   782.62 792.62
## - Glucose                   1   871.93 881.93

##
## Call:  glm(formula = y ~ Pregnancies + Glucose + BMI + DiabetesPedigreeFun
ction +
##     Age, family = "binomial", data = train)
##
## Coefficients:
##              (Intercept)               Pregnancies                   Gluco
se
##                  -9.58212                   0.13949                   0.037
66
##                      BMI  DiabetesPedigreeFunction                       A
ge
##                   0.09921                   0.87853                   0.018
51
##
## Degrees of Freedom: 749 Total (i.e. Null);  744 Residual
## Null Deviance:       1039
## Residual Deviance: 731.8     AIC: 743.8

glm1<-glm(y ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin
+
            BMI + DiabetesPedigreeFunction + Age ,family = "binomial",data=di
ab.balance) #drop skinthickness
glm11<-glm(y ~ Pregnancies + Glucose + BloodPressure + Insulin + BMI + Diabet
esPedigreeFunction +
             Age,family = "binomial",data=diab.balance) #drop Insulin
glm2<-glm(y ~ Pregnancies + Glucose + BloodPressure + BMI + DiabetesPedigreeF
unction +
            Age,family = "binomial",data=diab.balance)

summary(glm2)#AIC=989.95

##
## Call:
```

```
## glm(formula = y ~ Pregnancies + Glucose + BloodPressure + BMI +
##     DiabetesPedigreeFunction + Age, family = "binomial", data = diab.balan
ce)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -3.12451  -0.74657  -0.05197   0.71629   2.19919
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -8.996389   0.708844 -12.692  < 2e-16 ***
## Pregnancies               0.130455   0.027936   4.670 3.02e-06 ***
## Glucose                   0.039267   0.003223  12.183  < 2e-16 ***
## BloodPressure            -0.011091   0.007248  -1.530  0.12595
## BMI                       0.099254   0.013328   7.447 9.55e-14 ***
## DiabetesPedigreeFunction  0.943195   0.256842   3.672  0.00024 ***
## Age                       0.017734   0.008389   2.114  0.03452 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1386.29  on 999  degrees of freedom
## Residual deviance:  967.95  on 993  degrees of freedom
## AIC: 981.95
##
## Number of Fisher Scoring iterations: 5

pred2<-predict(glm2,test,type ="response")
glm.pred=rep("0",250)
glm.pred[pred2>.5]="1"
table(glm.pred,test$y)

##
## glm.pred   0   1
##        0 106  29
##        1  29  86

mean(glm.pred!=test$y) #error rate 48.5%

## [1] 0.232

table(test$y, pred2> 0.5)

##
##     FALSE TRUE
##   0   106   29
##   1    29   86

ROCRpred2<-prediction(pred2, test$y)
ROCRperf2<-performance(ROCRpred2, 'tpr','fpr')
```

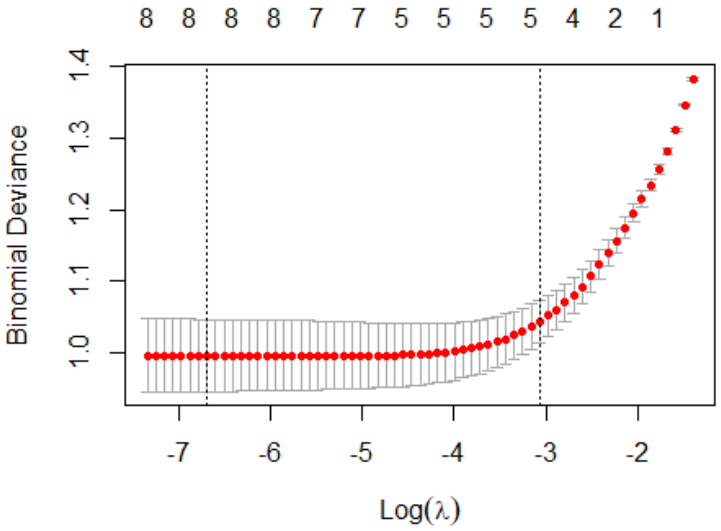
```
auc2 = performance(ROCRpred2, 'auc')@y.values #0.848
plot(ROCRperf2,col="red",text.adj = c(-0.2,1.7))
```



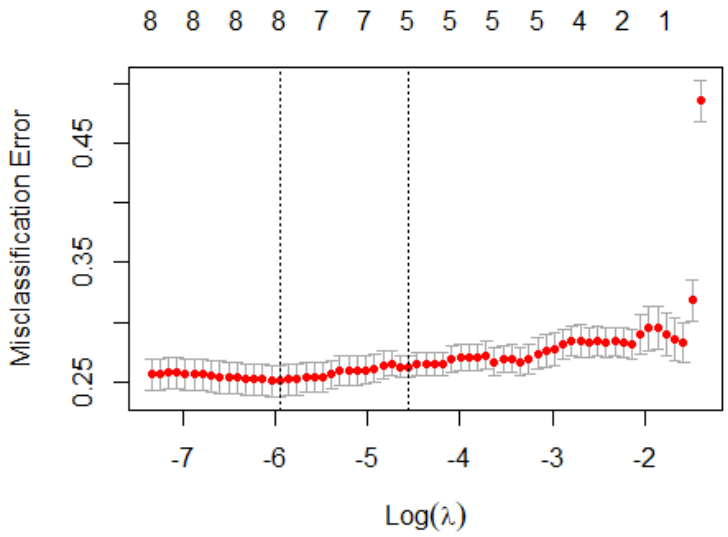

```
##################
# The Lasso ####
##################

set.seed(447) #setting seed
x <- model.matrix(y ~.,diab.balance)[, -1]
#creating a model matrix including only the factors

y <-diab.balance$y
#setting y (response variable)
set.seed(447)
smpsize <- floor(0.75 * nrow(diab.balance))
train_ind <- sample(seq_len(nrow(diab.balance)), size = smpsize)
xtrain <- x[train_ind, ] #training set
xtest <- x[-train_ind, ] #test set

y.train=y[train_ind] #creating y from train
y.test = y[-train_ind]# creating y from test

grid=10^seq(10,-2,length=100) #grid of lambda

lasso.mod=glmnet(xtrain,y.train,alpha=1,family = "binomial",lambda=grid)
#alpha =1 is the lasso penalty.
plot(lasso.mod, xvar="lambda",label = TRUE) #so many parameters, hard to inte
rpret. use CV to choose best lambda
```



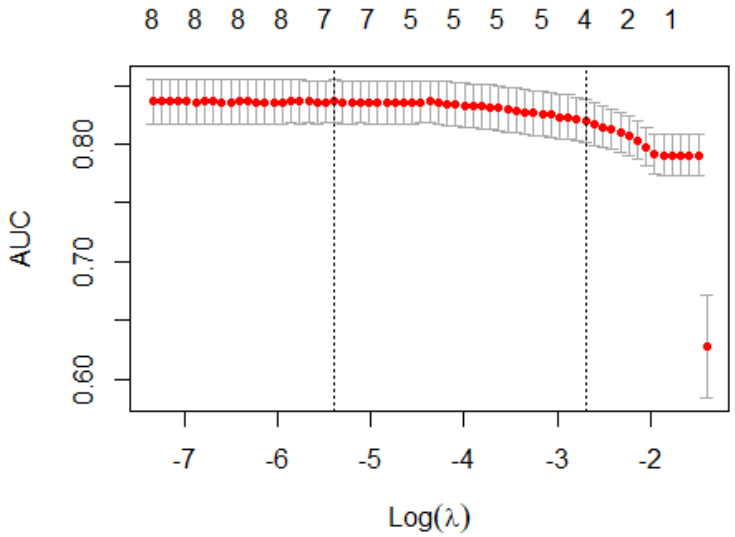

```
#Using Cross Validation to choose the best lambda (3 ways:deviance,classifcat
ion error rate & auc)
set.seed(447)
cv.glmmod <- cv.glmnet(xtrain, y.train, alpha=1,family = "binomial")
cv.glmmod1 <- cv.glmnet(xtrain, y.train, alpha=1,family =
                        "binomial",type.measure = "class")
cv.glmmod2 <- cv.glmnet(xtrain, y.train, alpha=1,family =
                        "binomial",type.measure = "auc")
plot(cv.glmmod) #binomial deviance
```




```
plot(cv.glmmod1) #classification error
```

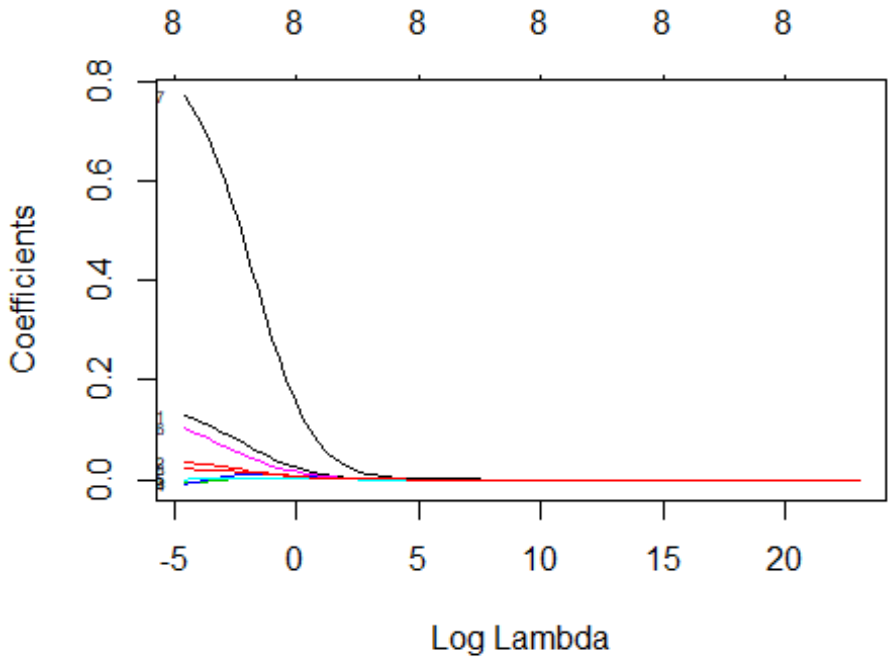



```
plot(cv.glmmod2) #AUC
```



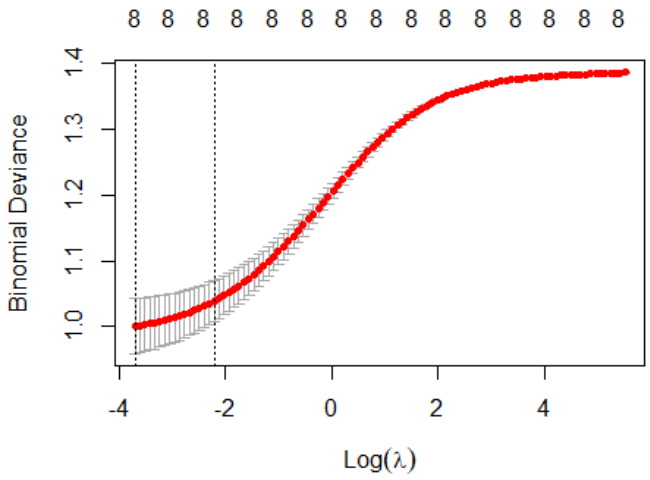

```
best.lambda<- cv.glmmod$lambda.1se
best.lambda1<- cv.glmmod1$lambda.1se
best.lambda2<- cv.glmmod2$lambda.1se

bestlam <- c(best.lambda,best.lambda1,best.lambda2)
pred.lasso=predict(lasso.mod,s=best.lambda1,newx=xtest,type = "class")
lasso.err<-mean(pred.lasso!=y.test) #error rate in test data
out = glmnet(x,y,alpha=1,family = "binomial")
lasso.coef = predict(out,type = "coefficients", s = best.lambda1)
lasso.coef
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                       1
## (Intercept)                -8.3377157098
## Pregnancies                 0.1076750328
## Glucose                     0.0356094328
## BloodPressure              -0.0002986872
## SkinThickness               .
## Insulin                     .
## BMI                         0.0816623442
## DiabetesPedigreeFunction    0.7237228688
## Age                         0.0116039836

#classification error is 0.256

#########################
# RIDGE REGRESSION ####
#########################

ridge.mod=glmnet(xtrain,y.train,alpha=0,family = "binomial",lambda=grid)
#alpha =0 is the ridge penalty.
plot(ridge.mod, xvar="lambda",label = TRUE) #so many parameters, hard to inte
rpret. use CV to choose best lambda
```

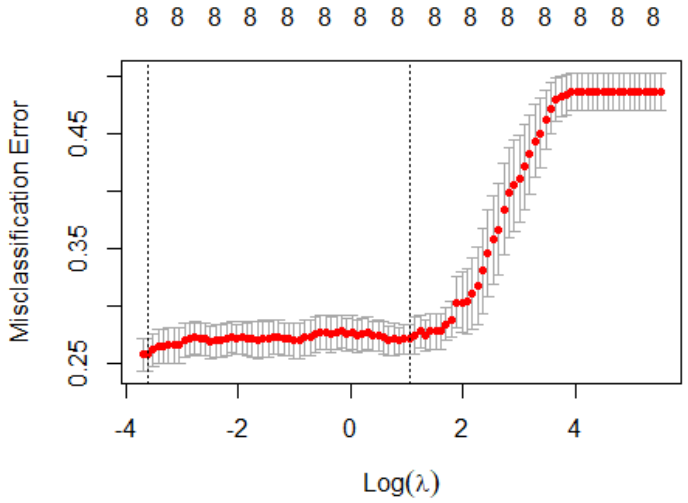



```
#Using Cross Validation to choose the best lambda (3 ways:deviance,classifcat
ion error rate & auc)
set.seed(447)
cv.ridmod <- cv.glmnet(xtrain, y.train, alpha=0,family = "binomial")
```

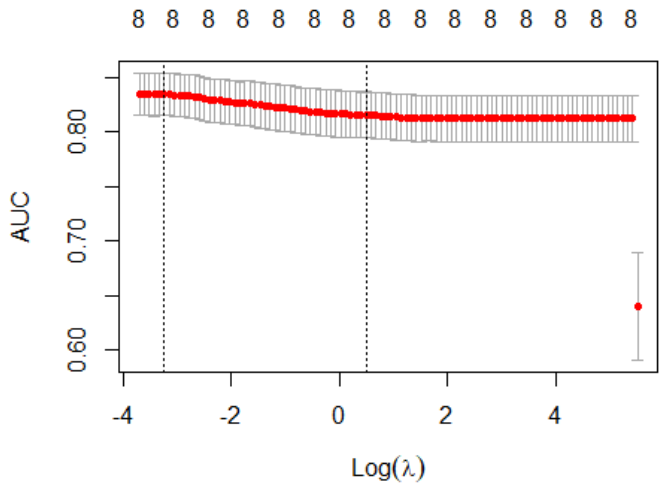
```
cv.ridmod1 <- cv.glmnet(xtrain, y.train, alpha=0,family =
                        "binomial",type.measure = "class")
cv.ridmod2 <- cv.glmnet(xtrain, y.train, alpha=0,family =
                        "binomial",type.measure = "auc")
plot(cv.ridmod) #binomial deviance
```




```
plot(cv.ridmod1) #classification error
```

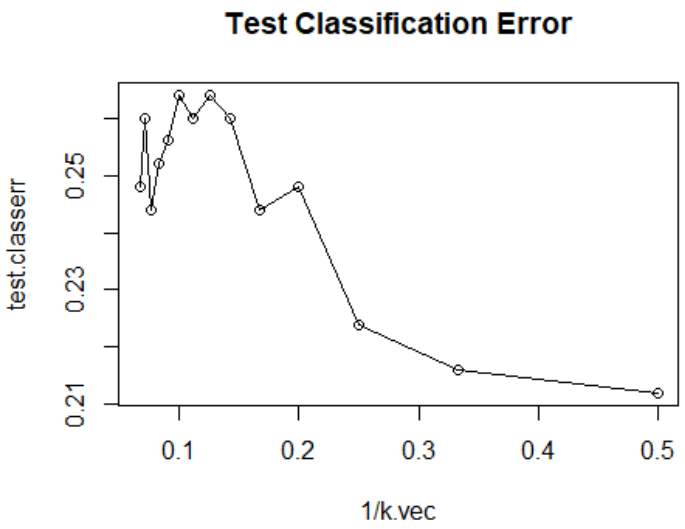



```
plot(cv.ridmod2) #AUC
```




```
best.lam<- cv.ridmod$lambda.1se
best.lam1<- cv.ridmod1$lambda.1se
best.lam2<- cv.ridmod2$lambda.1se

bestlam.rid <- c(best.lam,best.lam1,best.lam2)

pred.ridge=predict(ridge.mod,s=best.lam1,newx=xtest,type = "class")
ridge.err<-mean(pred.ridge!=y.test) #error rate in test data
out = glmnet(x,y,alpha=0,family = "binomial")
ridge.coef = predict(out,type = "coefficients", s = best.lam1)
ridge.coef

## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                      1
## (Intercept)               -1.0425994834
## Pregnancies                0.0095627100
## Glucose                    0.0024678661
## BloodPressure              0.0017228322
## SkinThickness              0.0040945822
## Insulin                    0.0004982991
## BMI                        0.0067726201
## DiabetesPedigreeFunction   0.0721609093
## Age                        0.0030366941

#classification error is 0.252
#########################

# KNN ####
#########################

#using the training and test data from lasso and ridge
set.seed(447)
```

```
knn.pred=knn(xtrain,xtest,y.train,k=1)
mean(y.test!=knn.pred) # error rate 0.172

## [1] 0.176

knn.pred1=knn(xtrain,xtest,y.train,k=5)
mean(y.test!=knn.pred1) # error rate 0.204

## [1] 0.256

knn.pred2=knn(xtrain,xtest,y.train,k=10)
mean(y.test!=knn.pred2) # error rate 0.248

## [1] 0.256

#Using CV to find best K
k.vec=2:15
test.classerr=rep(0,length(k.vec))
for (i in 1:length(k.vec))
{
  k=k.vec[i]
  knn.pred=knn(xtrain,xtest,y.train,k=k)
  test.classerr[i]=mean(y.test!=knn.pred)
}
plot(1/k.vec,test.classerr,type="o",main="Test Classification Error") #best i
s 1/kvec = 0.25 or k = 4
```

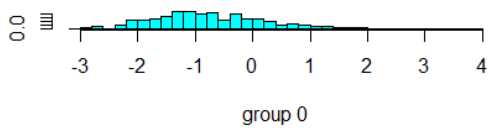


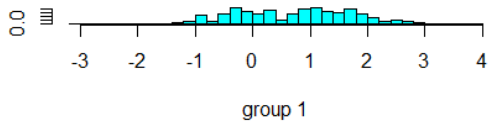

```
knn.predbest=knn(xtrain,xtest,y.train,k=4)
mean(y.test!=knn.predbest) # error rate 0.196

## [1] 0.236

########################################
# Linear Discriminant Analysis #######
```

```r
#######################################
#Using validation set method
smp_size <- floor(0.75 * nrow(diab.balance))
train_ind <- sample(seq_len(nrow(diab.balance)), size = smp_size)
train <- diab.balance[train_ind, ] #training set
test <- diab.balance[-train_ind, ] #test set
lda.fit=lda(y~.,data=train) #lda fit
lda.fit
```

```
## Call:
## lda(y ~ ., data = train)
##
## Prior probabilities of groups:
##     0     1
## 0.508 0.492
##
## Group means:
##   Pregnancies  Glucose BloodPressure SkinThickness  Insulin      BMI
## 0    3.430446 110.4162      70.52756      27.50002 131.5466 30.98324
## 1    4.739837 144.2905      74.80672      32.20528 201.6960 35.41789
##   DiabetesPedigreeFunction      Age
## 0                0.4308793 31.60892
## 1                0.5519973 37.36856
##
## Coefficients of linear discriminants:
##                                  LD1
## Pregnancies                0.0641419952
## Glucose                    0.0289917255
## BloodPressure             -0.0024241117
## SkinThickness             -0.0050270015
## Insulin                   -0.0002949292
## BMI                        0.0646269227
## DiabetesPedigreeFunction   0.6065579743
## Age                        0.0188487401
```

```r
plot(lda.fit)
```





```r
lda.pred=predict(lda.fit, test)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```r
lda.class=lda.pred$class # the predicted class for each observation in test d
ata
table(lda.class,test$y)
```

```
##
## lda.class  0  1
##         0 99 49
##         1 20 82
```

```r
mean(lda.class!=test$y) # really high error rate 0.256
```

```
## [1] 0.276
```

```r
####################################
# Quadratic Discriminant Analysis ##
####################################
qda.fit=qda(y~.,data=train)
qda.fit
```

```
## Call:
## qda(y ~ ., data = train)
##
## Prior probabilities of groups:
##     0     1
## 0.508 0.492
##
## Group means:
##   Pregnancies  Glucose BloodPressure SkinThickness  Insulin      BMI
## 0    3.430446 110.4162      70.52756      27.50002 131.5466 30.98324
## 1    4.739837 144.2905      74.80672      32.20528 201.6960 35.41789
##   DiabetesPedigreeFunction      Age
## 0                0.4308793 31.60892
## 1                0.5519973 37.36856
```

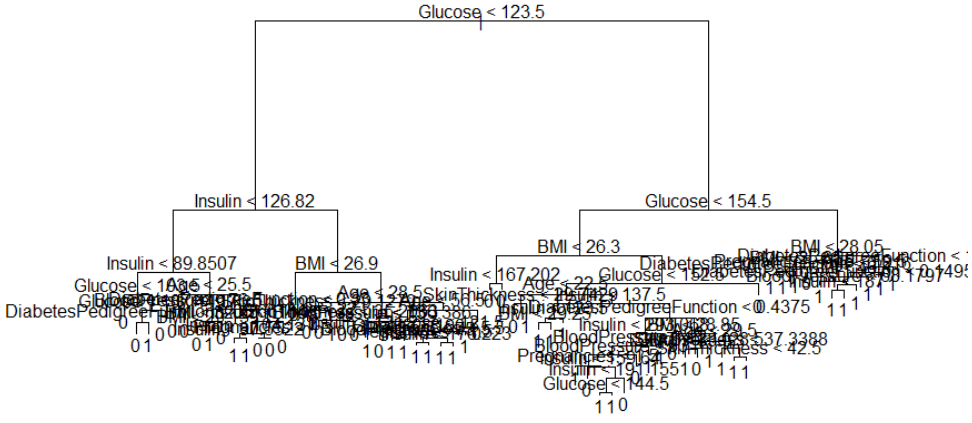
```r
qda.class=predict(qda.fit,test)$class
table(qda.class,test$y)
```

```
##
## qda.class  0  1
##         0 95 51
##         1 24 80
```

```r
mean(qda.class!=test$y) #error rate 0.276
```

```
## [1] 0.3
```

```
######################
#Decision Trees ####
######################
set.seed(1)
tree.diab=tree(y~.,data = train,split = "gini")
#type="class" return the actual class prediction
summary(tree.diab)

##
## Classification tree:
## tree(formula = y ~ ., data = train, split = "gini")
## Number of terminal nodes:  71
## Residual mean deviance:  0.5079 = 344.9 / 679
## Misclassification error rate: 0.1187 = 89 / 750

tree.pred=predict(tree.diab,test,type="class")
table(tree.pred,test$y)

##
## tree.pred  0  1
##         0 94 45
##         1 25 86

mean(tree.pred!=test$y) #test class error rate 0.22

## [1] 0.28

plot(tree.diab)
text(tree.diab,pretty=0)
```




```
#pruning the tree
set.seed(447)
#cv.tree selects the optimal level of tree complexity
```

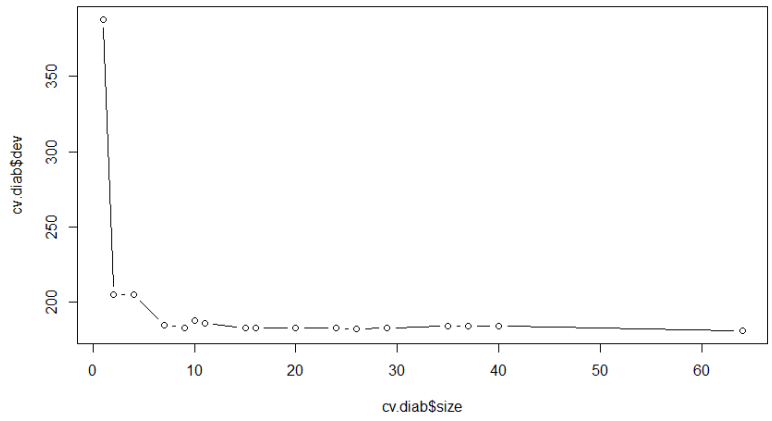
```r
#FUN=prune.misclass means that we want the
#classfication error to guide the CV
#default seting is deviance
cv.diab=cv.tree(tree.diab,FUN=prune.misclass)
names(cv.diab)
```

```
## [1] "size"   "dev"    "k"      "method"
```

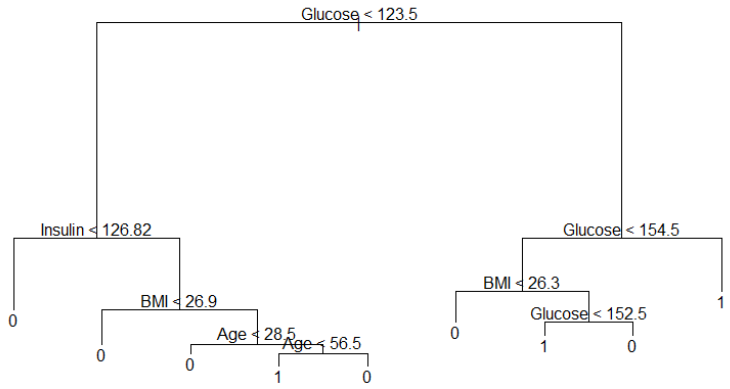
```r
cv.diab
```

```
## $size
##  [1] 71 43 40 38 37 28 22 19 14 11  9  2  1
##
## $dev
##  [1] 182 183 183 183 182 184 182 182 180 180 191 191 385
##
## $k
##  [1]        -Inf   0.0000000   0.3333333   0.5000000   1.0000000   1.66666
67
##  [7]   2.0000000   2.3333333   3.0000000   4.0000000   5.0000000   5.85714
29
## [13] 165.0000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
#dev: CV error rate
#size: number of terminal nodes of each tree
#k: complexity parameter (corresponds to alpha)
par(mfrow=c(1,1))
plot(cv.diab$size,cv.diab$dev,type="b")
```



```
#with 9 terminal nodes with lowest CV error
prune.diab=prune.misclass(tree.diab,best=9)
summary(prune.diab)

##
## Classification tree:
## snip.tree(tree = tree.diab, nodes = c(52L, 53L, 108L, 109L, 110L,
## 7L, 2L, 12L, 111L))
## Variables actually used in tree construction:
## [1] "Glucose"       "Pregnancies"   "Age"           "SkinThickness"
## [5] "BMI"
## Number of terminal nodes:  9
## Residual mean deviance:  1.007 = 745.9 / 741
## Misclassification error rate: 0.2173 = 163 / 750

plot(prune.diab)
text(prune.diab,pretty=0)
```




```
tree.pred=predict(prune.diab,test,type="class")
table(tree.pred,test$y)

##
## tree.pred  0  1
##         0 85 32
##         1 34 99

mean(tree.pred!=test$y) #24.8% test err

## [1] 0.264

#a larger pruned tree (size=8) next smallest size
prune.diab2=prune.misclass(tree.diab,best=8)
summary(prune.diab2)
```

```
##
## Classification tree:
## snip.tree(tree = tree.diab, nodes = c(52L, 53L, 108L, 109L, 110L,
## 7L, 2L, 12L, 111L))
## Variables actually used in tree construction:
## [1] "Glucose"       "Pregnancies"   "Age"           "SkinThickness"
## [5] "BMI"
## Number of terminal nodes:  9
## Residual mean deviance:  1.007 = 745.9 / 741
## Misclassification error rate: 0.2173 = 163 / 750
```

```
plot(prune.diab2)
text(prune.diab2,pretty=10)
```

```
tree.pred2=predict(prune.diab2,test,type="class")
table(tree.pred2,test$y)
```

```
##
## tree.pred2  0  1
##          0 85 32
##          1 34 99
```

```
mean(tree.pred2!=test$y) #same test err
```

```
## [1] 0.264
```

```
##############################
#Bagging and Random Forests #
##############################
#BAGGING
bag.diab=randomForest(y~.,data=train,importance=TRUE, mtry=9)
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within
valid
## range
```

```
bag.diab
```

```
##
## Call:
##  randomForest(formula = y ~ ., data = train, importance = TRUE,      mtry
= 9)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 15.6%
## Confusion matrix:
##     0   1 class.error
```

```
## 0 303  78   0.2047244
## 1  39 330   0.1056911
```

```
yhat.bag1= predict(bag.diab,test,type = "class")
table(yhat.bag1,test$y)
```

```
##
## yhat.bag1   0   1
##         0  99  24
##         1  20 107
```

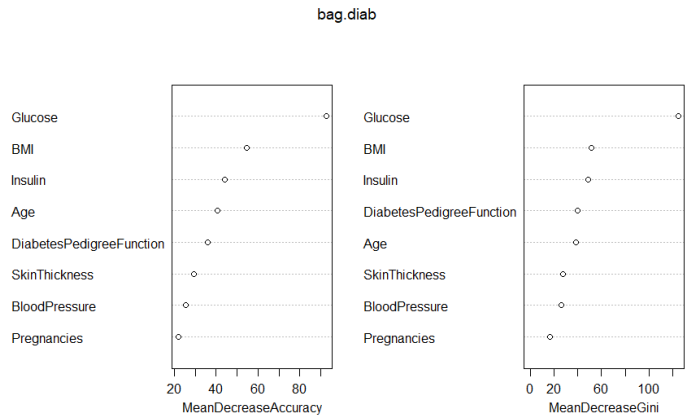
```
mean(yhat.bag1!=test$y) #very low test error 0.144
```

```
## [1] 0.176
```

```
importance(bag.diab) # importance of each predictor
```

```
##                                0        1 MeanDecreaseAccuracy
## Pregnancies                12.795172 24.71718              25.82413
## Glucose                    36.646500 79.48631              80.63636
## BloodPressure               3.759584 27.06284              23.28631
## SkinThickness               6.783669 28.63964              27.58456
## Insulin                    11.083175 48.02972              47.95741
## BMI                        15.214295 53.28156              49.01349
## DiabetesPedigreeFunction    9.277494 37.55030              35.98086
## Age                        15.494973 50.78887              50.04008
##                           MeanDecreaseGini
## Pregnancies                       18.50305
## Glucose                          110.94092
## BloodPressure                     22.81725
## SkinThickness                     25.87641
## Insulin                           64.11448
## BMI                               50.82649
## DiabetesPedigreeFunction          37.97966
## Age                               43.31149
```

```
varImpPlot(bag.diab)
```



```r
#RANDOM FOREST
fit <- randomForest(y ~ ., data=train, importance = TRUE)
print(fit) # view results
```

```
##
## Call:
##  randomForest(formula = y ~ ., data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 15.33%
## Confusion matrix:
##     0   1 class.error
## 0 303  78   0.2047244
## 1  37 332   0.1002710
```

```r
summary(fit)
```

```
##                 Length Class  Mode
## call               4   -none- call
## type               1   -none- character
## predicted        750   factor numeric
## err.rate        1500   -none- numeric
## confusion          6   -none- numeric
## votes           1500   matrix numeric
## oob.times        750   -none- numeric
## classes            2   -none- character
## importance        32   -none- numeric
## importanceSD      24   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y                750   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```r
yhat.rf = predict(fit,newdata=test)
mean(yhat.rf!=test$y) #very low test error 14.4%
```

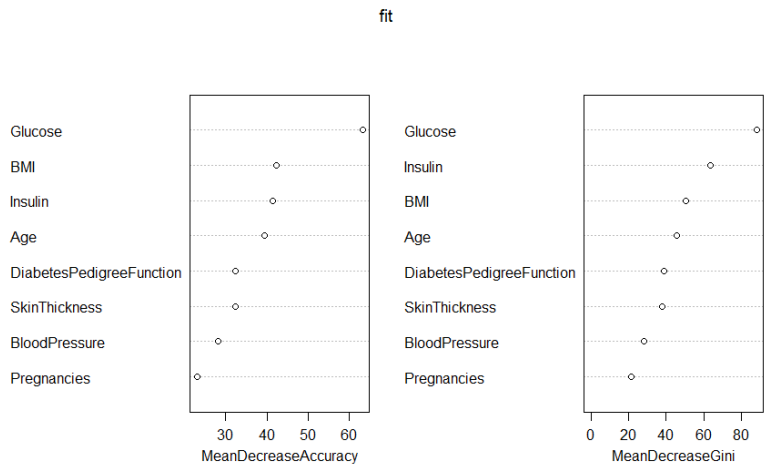
```
## [1] 0.176
```

```r
importance(fit) # importance of each predictor
```

```
##                                0        1 MeanDecreaseAccuracy
## Pregnancies             8.940485 26.84071              26.24094
## Glucose                28.316648 61.33780              59.47383
## BloodPressure           3.549854 28.47378              25.85128
```

```
## SkinThickness              7.288188 35.63409              33.31724
## Insulin                   13.431094 44.04563              44.64224
## BMI                       10.990598 44.01556              43.67595
## DiabetesPedigreeFunction   5.946409 39.61031              36.42389
## Age                       15.238199 42.02155              43.99150
##                          MeanDecreaseGini
## Pregnancies                         22.11694
## Glucose                             85.18324
## BloodPressure                       26.12792
## SkinThickness                       37.11760
## Insulin                             68.29572
## BMI                                 52.66643
## DiabetesPedigreeFunction            37.68087
## Age                                 45.13229
```

```
varImpPlot(fit)
```




```
#SVM
library(e1071)
set.seed(1)
tune.out=tune(svm, y~., data=train, kernel="radial", ranges=list(cost=c(0.1,1
,10,100,1000),gamma=c(0.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     4
##
## - best performance: 0.1973333
##
```

```
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.2786667 0.06043464
## 2  1e+00   0.5 0.2146667 0.06665185
## 3  1e+01   0.5 0.2160000 0.07769519
## 4  1e+02   0.5 0.2186667 0.06538018
## 5  1e+03   0.5 0.2186667 0.06538018
## 6  1e-01   1.0 0.3346667 0.15414599
## 7  1e+00   1.0 0.2146667 0.06605647
## 8  1e+01   1.0 0.2120000 0.06635483
## 9  1e+02   1.0 0.2120000 0.06635483
## 10 1e+03   1.0 0.2120000 0.06635483
## 11 1e-01   2.0 0.5266667 0.05342585
## 12 1e+00   2.0 0.1986667 0.06782694
## 13 1e+01   2.0 0.2013333 0.06724196
## 14 1e+02   2.0 0.2013333 0.06724196
## 15 1e+03   2.0 0.2013333 0.06724196
## 16 1e-01   3.0 0.5266667 0.05342585
## 17 1e+00   3.0 0.2000000 0.05296167
## 18 1e+01   3.0 0.2000000 0.05896222
## 19 1e+02   3.0 0.2000000 0.05896222
## 20 1e+03   3.0 0.2000000 0.05896222
## 21 1e-01   4.0 0.5266667 0.05342585
## 22 1e+00   4.0 0.1973333 0.04523737
## 23 1e+01   4.0 0.2000000 0.04868645
## 24 1e+02   4.0 0.2000000 0.04868645
## 25 1e+03   4.0 0.2000000 0.04868645

table(true=test$y, pred=predict(tune.out$best.model,newdata=test))

##     pred
## true   0   1
##    0 119   0
##    1  54  77

#15.6% misclassification rate in test data
```