BA with R

PREDICT THE ONSET OF DIABETES BASED ON DIAGNOSTIC MEASURES

GROUP 6

Hanisha Reddy Mittapelli
Pragathi Prasanna
Prateek Nayan
Priyanka Prakash Babu
Sirija Reddy Salimadugu

Under guidance of :
Prof. Jianqing Chen

The University of Texas at Dallas

Naveen Jindal School of Management

Spring 2020

Contents

1. Objective

The objective of the dataset is to diagnostically predict whether a patient has diabetes, based on certain diagnostic measurements included in the dataset. Increasing the accuracy in detecting the diabetes of a patient by using classification techniques is the main goal of this project.

2. Introduction

The goal for this project is to predict (classify) Diabetes in individual women based on several medical condition and clinical tests. Diabetes is a very common metabolic disease. Generally, a person is suffering from diabetes, when blood sugar levels are above normal (4.4 to 6.1 mmol/L). Pancreas present in the human body produces insulin, a hormone that is responsible to help glucose reach each cell of the body. A diabetic patient essentially has low production of insulin or their body is not able to use the insulin well. There are three main types of diabetes, viz. Type 1, Type 2 and Gestational. Type 1 – The disease manifests as an autoimmune disease occurring at a very young age of below 20 years. In this type of diabetes, the pancreatic cells that produce insulin have been destroyed. Type 2 - Diabetes is in the state when the various organs of the body become insulin resistant, and this increases the demand for insulin. At this point, the pancreas doesn't make the required amount of insulin. Gestational diabetes tends to occur in pregnant women, as the pancreas doesn't make enough insulin.

Undiagnosed diabetes may result in very high blood sugar levels referred to as hyperglycemia which can lead to complications like diabetic retinopathy, nephropathy, neuropathy, cardiac stroke and foot ulcer. So, early detection of diabetes is very important to improve the quality of life of patients and enhancement of their life expectancy. Models used in this project would help in predicting (classifying) Diabetes in individual women. This modelling will help medical industry to diagnose diabetes with less efforts and more reliability.

3. Problem Definition

The dataset that is obtained from Kaggle is originally from the *National Institute of Diabetes and Digestive and Kidney Diseases*. The objective of this dataset is to predict if a patient has diabetes or no. The dataset consists of several medical predictor variables and one target variable, which is the outcome variable. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on. We applied several data cleaning techniques on the dataset, which showed that dataset is almost cleaned from the source. The dataset has 8 defining attributes and 1 outcome (as diabetes YES or NO). We have divided 60% as training dataset and 40% data as testing dataset. We found algorithms which work best with the available dataset.

4. The Data Collection Process

4.1 Types of data we considered:

We considered three datasets before we could finalize on the one, we have used. There were a lot of factors that led us to choose this dataset. When we considered the first dataset which had 50 variables and several thousand rows, there were columns that seemed unrelated to the prediction or out of context to help us know if the patient had diabetes or not. The second dataset which had 900 variables and millions of rows, had too many variables filled with dirty data, which seemed impossible to sort and arrive at a clean structure. Finally, we came across a dataset that seemed apt for determining the outcome whether a patient has diabetes or not. This dataset has 9 variables and 700 rows. Among all these datasets, majorly the column datatype was Boolean and numeric. Some columns were in the form of factors and integers.

4.2 Choosing data sets:

With the first dataset, we tried to clean and performed preprocessing on it. We were unable to decide on the predictors as it contained too many vague columns like NULL values, special characters and missing/unknown values. On the other hand, we tried to omit the columns only to end up with some of the unrelated parameters. After removing all the NULL values, we got just 111 rows as clean data output. This led us to discard the dataset.

In case of the second one, filled with too many variables, had null values at most of the significant parameters that would help us in determining the outcome. We performed basic clean up like omitting the NULL values and Normalizing the dataset. Pre-processing became a tedious task with variables as much as 900. So, this prompted us to discard this dataset to move ahead with another dataset.

In the third one, we obtained almost clean data except with some missing values and it required very less effort as it consisted of 700 plus records with 9 variables. So, we proceeded with this dataset.

4.3 Cleaning the dataset:

Pre-processing of data is an important set that helps to look into the various aspects of data and to further perform exploratory analysis. In this dataset, there are 6 columns of Integer type and 2 are of numeric type, and 1 is of factor data type.

From the previous analysis some patients have missing data for some of the features. Some analytics algorithms don't work well when the data is missing so we must find a solution to clean the data we have. The easiest option could be to eliminate all those patients with null/zero values, but in this way, we would eliminate a lot of important data.

Another option is to calculate the median value for a specific column and substitute that value everywhere (in the same column) we have zero or null. We haven't transformed all the columns, because for some values it can make sense to be zero (like "Number of times pregnant"). This is nothing but imputing.

4.4 Difficulties we experience:

With the various pre-processing techniques available, we can say that the data cleaning process will be easy but when we have a look at the dataset, the real challenge is posted. Though we could just omit the records that are null or impute using the various methods available, it loses on the whole point of coming to the outcome. In the case of the first two datasets we faced this issue.
1st data set : We worked on 50 variable data set. It also consists of missing, unknown values and special characters. We found that this dataset is mostly unreliable because when we perform *NA.ommit*, we just got 111 rows.
2nd data set : We also worked on 900 variable data set, which was huge in size. We were unable to understand the meaning of each column as Data Dictionary was not provided.

With the last dataset that we came across, we required not much elimination of the null records and the variables pertaining to what are some of the important aspects in real life.

5. Dataset Information

We have taken this data from Kaggle. The data set consists of 9 variables out of which 8 independent variables are different medical predictors and the target variable is Outcome. Description for all of the variables is given below

Data Dictionary:

Sl. no.	Attribute Name	Description	Type of Attribute
1	Pregnant	Number of times pregnant	Integer
2	Glucose	Plasma glucose concentration a 2 hour in an oral glucose tolerance test	Integer
3	Diastolic	(Blood Pressure) Diastolic blood pressure (mm Hg)	Integer
4	Triceps	(Skin Thickness) Triceps skin fold thickness (mm)	Integer
5	Insulin	2-Hour serum insulin (mu U/ml)	Integer
6	BMI	Body mass index (weight in kg/ (height in m)^2)	Numeric
7	Diabetes	Diabetes pedigree function	Numeric
8	Age	Age (years)	Integer
9	Test	(Outcome) Class variable (0 or 1)	Factor w/2 levels

6. Splitting the Dataset:

To address the overfitting problem, we divide the entire dataset into two partitions : a training partition and a validation partition. We develop our model using only one of the partitions and then we use the trained model to make predictions on other partition and compare the predicted value with the pre-assigned label.

6.1 Training Partition:

The largest partition that has data used to build the various models is training partition. We generally use one training partition to develop multiple models.

6.2 Validation Partition:

The validation partition is used to analyze the predictive performance of each model which can be used to compare different models. The validation partition may be used in an automated fashion to tune and improve the model in some algorithms like classification and regression trees, k-nearest neighbors.

Splitting the dataset is a very important step for supervised machine learning models. Basically, we are going to use the first part to train the model, then we use the trained model to make predictions on new data (which is the test dataset, not part of the training set) and compare the predicted value with the pre-assigned label.

7. Comparing Multiple Algorithms

To know which algorithm will work better with our dataset, we will have to compare a few and choose the one with "best score"

We created a list of algorithms, calculated their scores. Finally compared these multiple algorithms to pick the algorithm with best score.

For this purpose, we performed the following:

- We need to consider high accuracy to identify best working algorithm.
- We need to take care of the False Negative ratio.
- The algorithm which predicted as Negative (No Diabetes) for the individual women, it turned out to be positive (YES Diabetes) is called False negative.
- Sensitivity is percentage of women that the classifier labeled as Positive are actually positive. [True Positive/ (True Positive +False Positive). Sensitivity is a true positive recognition
- Sensitivity is inversely proportional to False Negative. Higher the sensitivity, lower the False negative value. In this case we need to minimize false negative

Models:

7.1 PCA (Principle Component Analysis):

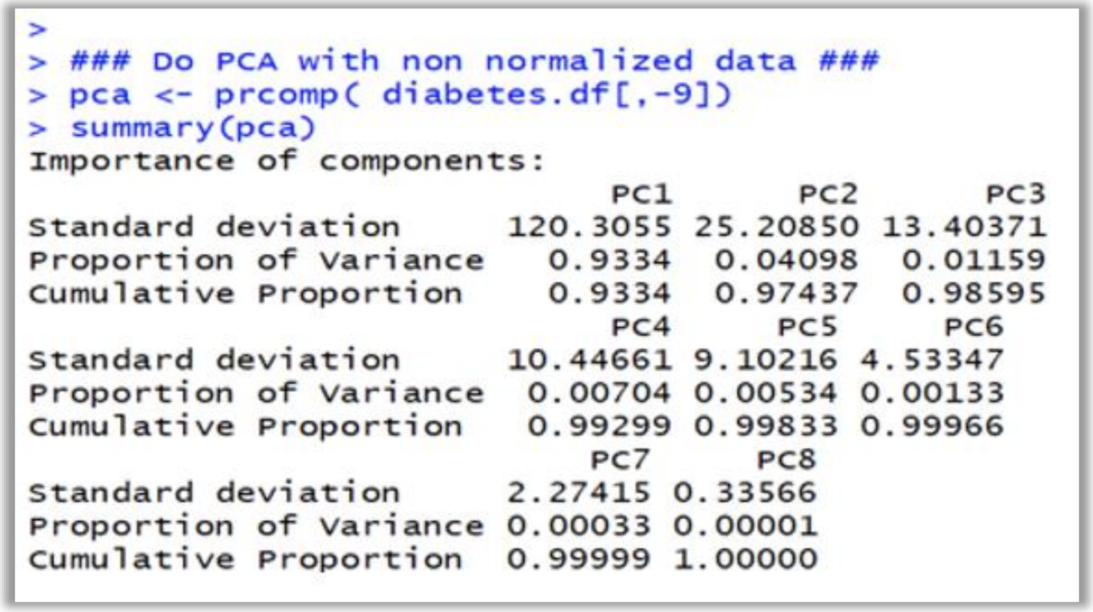
Principal components analysis (PCA) is a useful method for dimension reduction, especially when the number of variables is large. PCA is especially valuable when we have subsets of measurements that are measured on the same scale and are highly correlated. In that case, it provides a few variables (often as few as three) that are weighted linear combinations of the original variables, and that retain the majority of the information of the full original set.

This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.

The main goal of PCA is to get smaller number of numerical variables that contain most of the information. PCA is intended for use with numerical variables. For categorical variables, other methods such as correspondence analysis are more suitable.

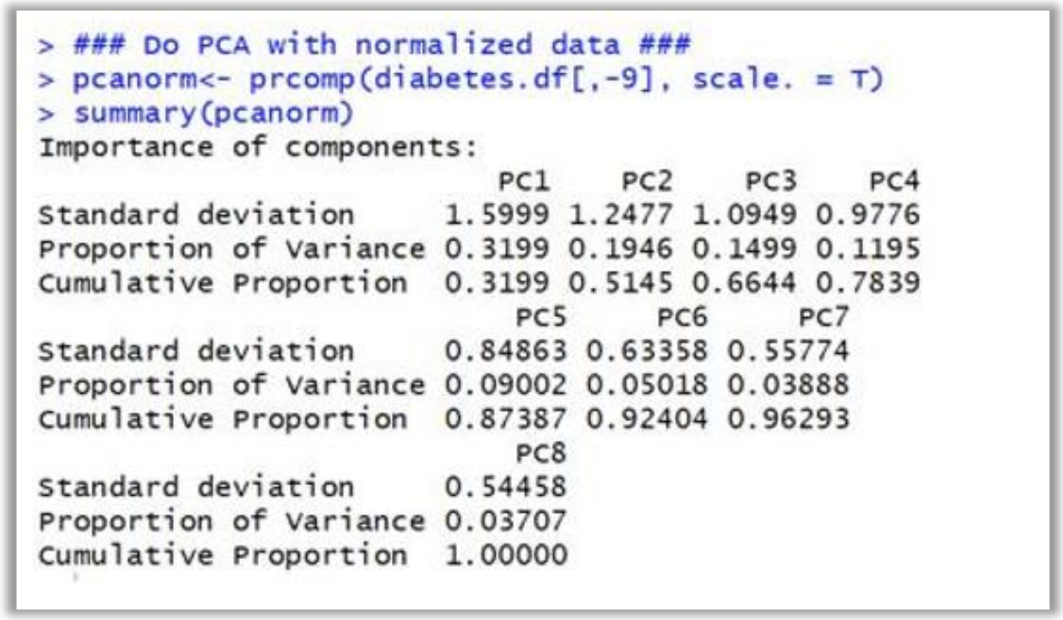
Below is the output of the PCA performed on our dataset:

Principle Component Analysis on the non-normalized dataset:



```
>
> ### Do PCA with non normalized data ###
> pca <- prcomp( diabetes.df[,-9])
> summary(pca)
Importance of components:
                            PC1      PC2      PC3
Standard deviation     120.3055 25.20850 13.40371
Proportion of Variance   0.9334  0.04098  0.01159
Cumulative Proportion    0.9334  0.97437  0.98595
                            PC4     PC5     PC6
Standard deviation     10.44661 9.10216 4.53347
Proportion of Variance  0.00704 0.00534 0.00133
Cumulative Proportion   0.99299 0.99833 0.99966
                           PC7     PC8
Standard deviation     2.27415 0.33566
Proportion of Variance 0.00033 0.00001
Cumulative Proportion  0.99999 1.00000
```

Principle Component Analysis on the normalized dataset:


```
> ### Do PCA with normalized data ###
> pcanorm<- prcomp(diabetes.df[,-9], scale. = T)
> summary(pcanorm)
Importance of components:
                          PC1    PC2    PC3    PC4
Standard deviation     1.5999 1.2477 1.0949 0.9776
Proportion of Variance 0.3199 0.1946 0.1499 0.1195
Cumulative Proportion  0.3199 0.5145 0.6644 0.7839
                           PC5     PC6     PC7
Standard deviation     0.84863 0.63358 0.55774
Proportion of Variance 0.09002 0.05018 0.03888
Cumulative Proportion  0.87387 0.92404 0.96293
                           PC8
Standard deviation     0.54458
Proportion of Variance 0.03707
Cumulative Proportion  1.00000
```

From the output of performing Principle Component Analysis on the non-normalized data , it can be analyzed that variation from all the principal components are contributing

This tells us that there is no scope to remove the variables and hence every variable is to be considered.

7.2 Logistic Regression

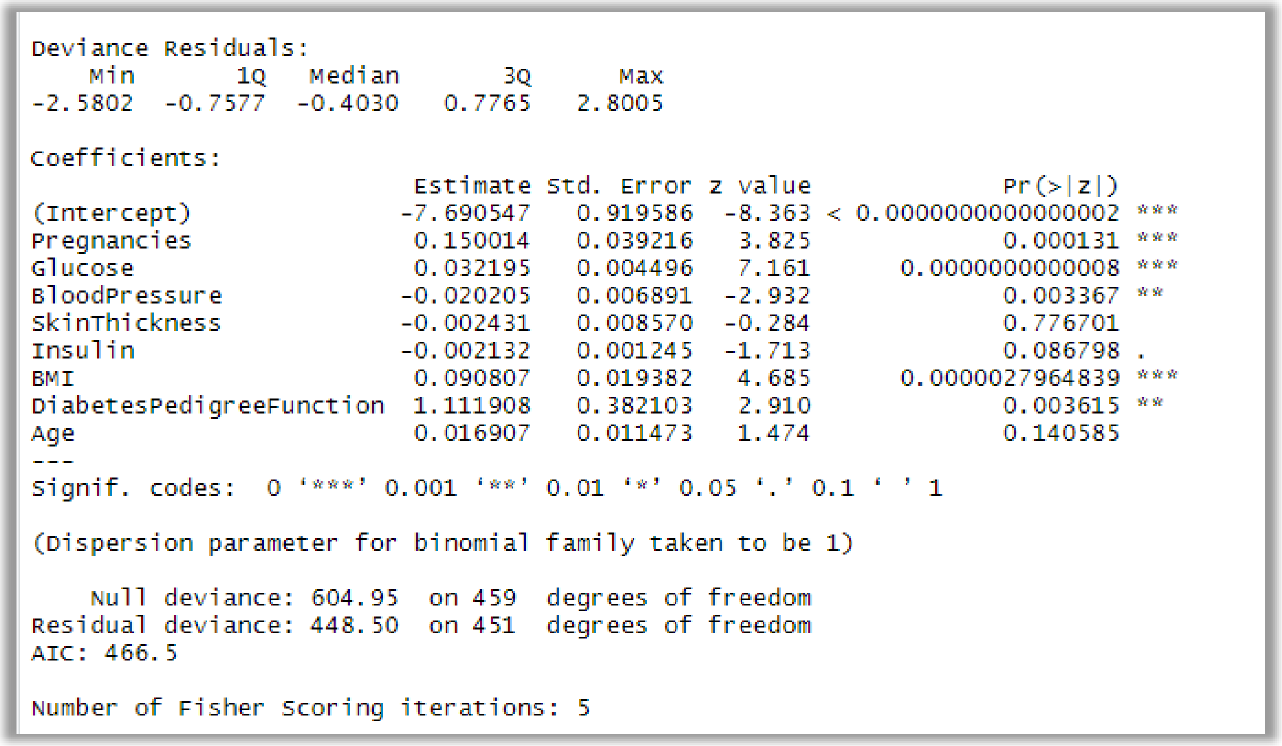
Logistic regression can be used for classifying a new record, where its class is unknown, into one of the classes, based on the values of its predictor variables (called classification). Logistic Regression is a classification algorithm. It is used to estimate binary values like 0/1, yes/no, true/false based on given set of independent variables. It predicts the probability of occurrence of an event by fitting data to a logit function. Hence, it is also known as logit regression. Since, it predicts the probability, its output values lie between 0 and 1.

In logistic regression, we take two steps:

- The first step yields estimate of the probabilities of belonging to each class.
- In the next step, we use a cutoff value on these probabilities in order to classify each case into one of the classes.

Output for logistic regression:

Training Partition:



```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5802  -0.7577  -0.4030   0.7765   2.8005

Coefficients:
                          Estimate Std. Error z value           Pr(>|z|)
(Intercept)              -7.690547   0.919586  -8.363 < 0.0000000000000002 ***
Pregnancies               0.150014   0.039216   3.825           0.000131 ***
Glucose                   0.032195   0.004496   7.161     0.0000000000008 ***
BloodPressure            -0.020205   0.006891  -2.932           0.003367 **
SkinThickness            -0.002431   0.008570  -0.284           0.776701
Insulin                  -0.002132   0.001245  -1.713           0.086798 .
BMI                       0.090807   0.019382   4.685     0.0000027964839 ***
DiabetesPedigreeFunction  1.111908   0.382103   2.910           0.003615 **
Age                       0.016907   0.011473   1.474           0.140585
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 604.95  on 459  degrees of freedom
Residual deviance: 448.50  on 451  degrees of freedom
AIC: 466.5

Number of Fisher Scoring iterations: 5
```

Interpretation of logistic regression:

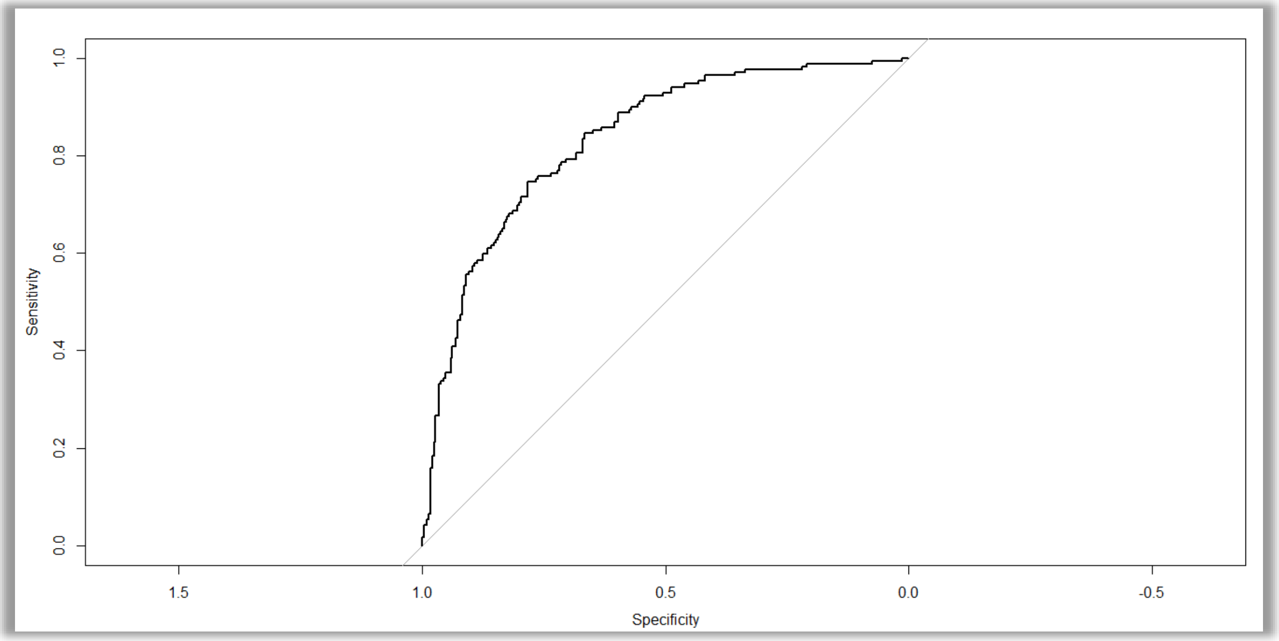
From the above output the model based on all predictors has estimated the logistic equation

log(p/1-p) = - 7.690547 + 0.1540014*Pregnancies + 0.032195*Glucose – 0.020205*BloodPressure - 0.002431*SkinThickness – 0.002132*Insulin + 0.090807*BMI + 1.111908* DiabetesPedigreeFunction + 0.016907*age

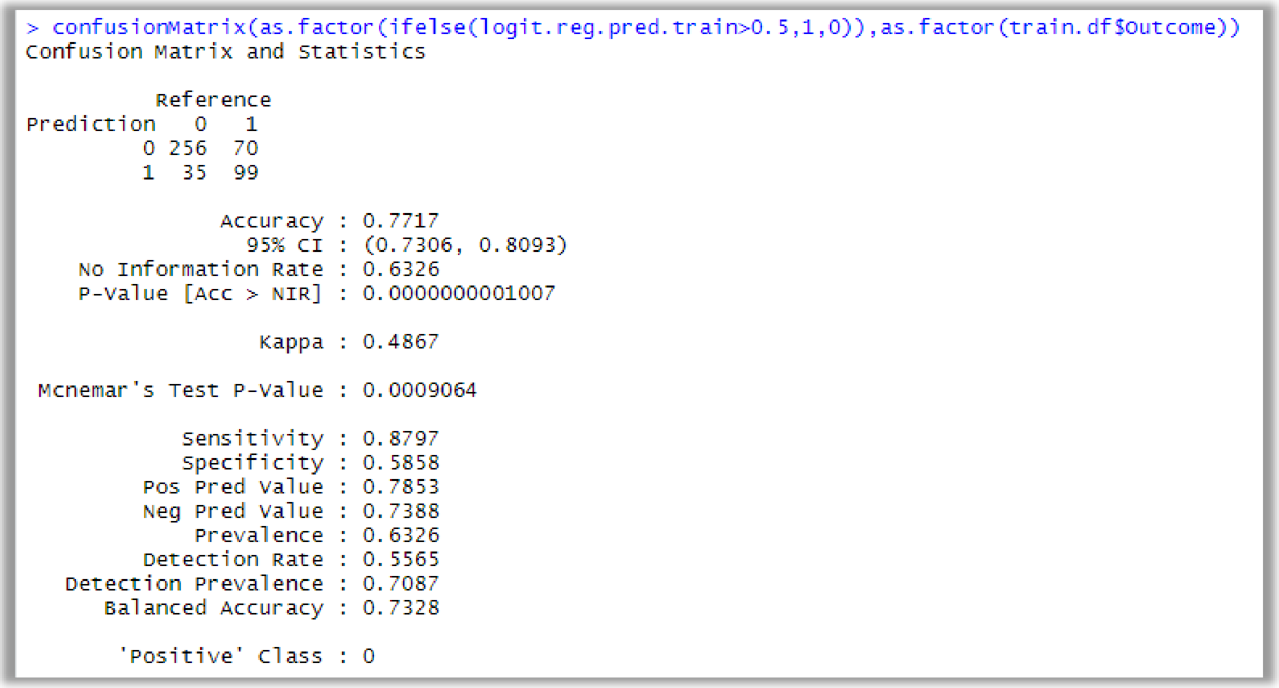
where p is probability of having diabetes

For the continuous predictors, positive coefficients indicate that a higher value on that predictor is associated with a higher probability of detecting as having a diabetes. Similarly, negative coefficients mean that a higher value on that predictor is associated with a lower probability of detecting as having a diabetes. As we can see, BMI, DiabetesPedigreeFunction and glucose are factors which are highly associated with having a diabetes.

Lift Chart:

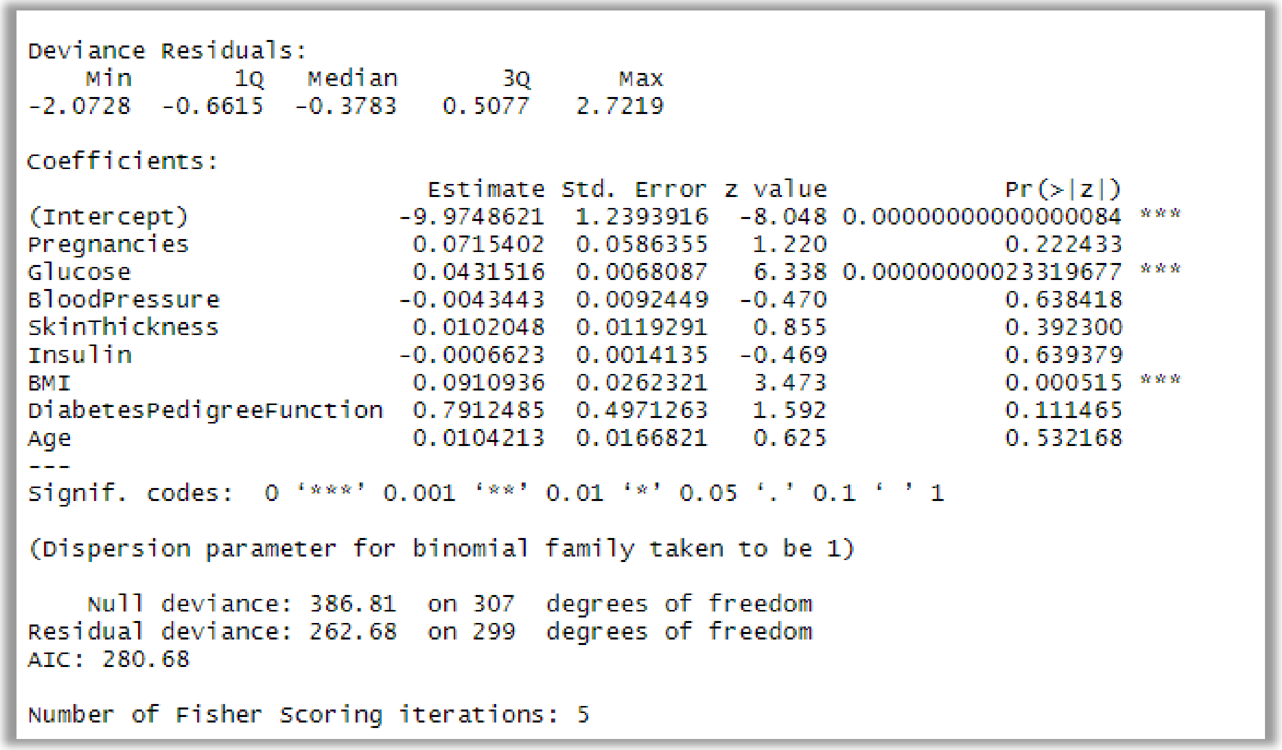


Confusion Matrix:



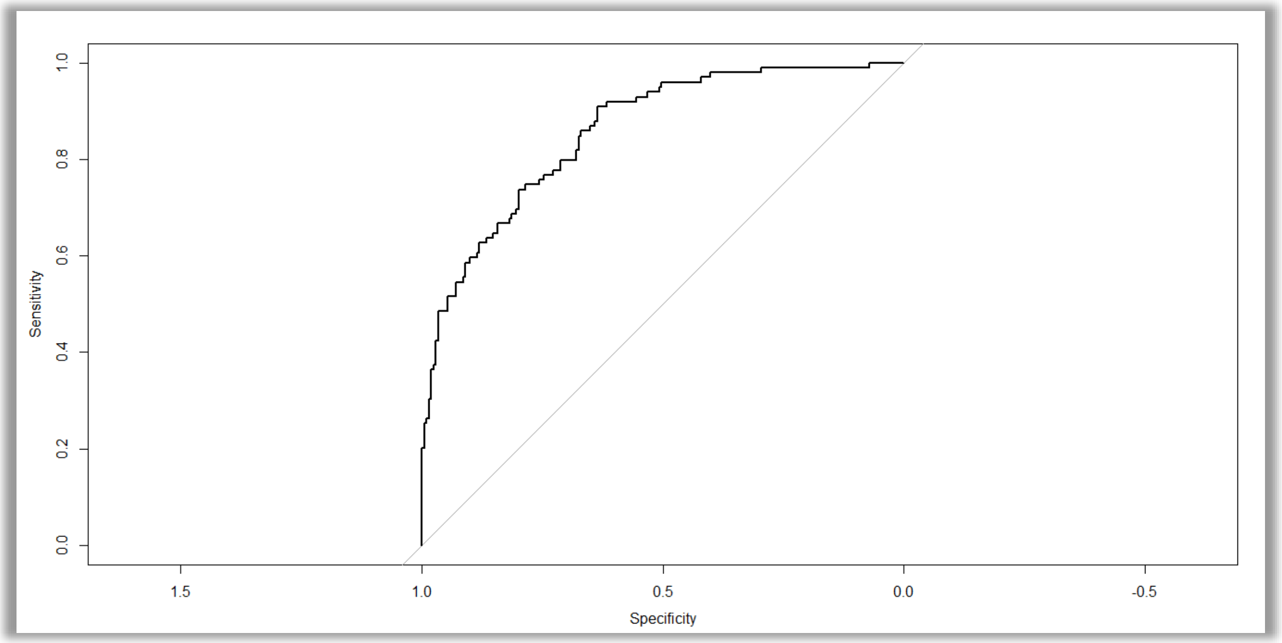
```
> confusionMatrix(as.factor(ifelse(logit.reg.pred.train>0.5,1,0)),as.factor(train.df$Outcome))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 256  70
         1  35  99

               Accuracy : 0.7717
                 95% CI : (0.7306, 0.8093)
    No Information Rate : 0.6326
    P-Value [Acc > NIR] : 0.0000000001007

                  Kappa : 0.4867

 Mcnemar's Test P-Value : 0.0009064

            Sensitivity : 0.8797
            Specificity : 0.5858
         Pos Pred Value : 0.7853
         Neg Pred Value : 0.7388
             Prevalence : 0.6326
         Detection Rate : 0.5565
   Detection Prevalence : 0.7087
      Balanced Accuracy : 0.7328

       'Positive' Class : 0
```

Validation Partition:

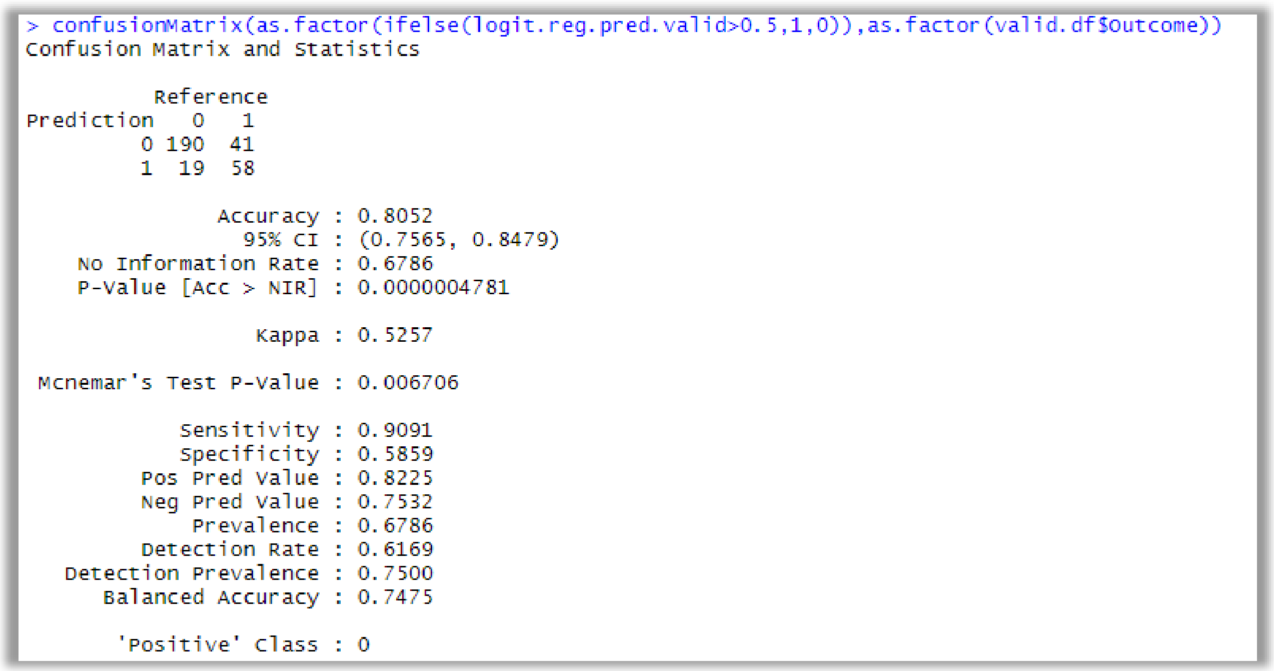


```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0728  -0.6615  -0.3783   0.5077   2.7219

Coefficients:
                           Estimate Std. Error z value            Pr(>|z|)
(Intercept)              -9.9748621  1.2393916  -8.048 0.00000000000000084 ***
Pregnancies               0.0715402  0.0586355   1.220            0.222433
Glucose                   0.0431516  0.0068087   6.338 0.00000000023319677 ***
BloodPressure            -0.0043443  0.0092449  -0.470            0.638418
SkinThickness             0.0102048  0.0119291   0.855            0.392300
Insulin                  -0.0006623  0.0014135  -0.469            0.639379
BMI                       0.0910936  0.0262321   3.473            0.000515 ***
DiabetesPedigreeFunction  0.7912485  0.4971263   1.592            0.111465
Age                       0.0104213  0.0166821   0.625            0.532168
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 386.81  on 307  degrees of freedom
Residual deviance: 262.68  on 299  degrees of freedom
AIC: 280.68

Number of Fisher Scoring iterations: 5
```

Lift Chart:



Confusion Matrix:


```
> confusionMatrix(as.factor(ifelse(logit.reg.pred.valid>0.5,1,0)),as.factor(valid.df$Outcome))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 190  41
         1  19  58

               Accuracy : 0.8052
                 95% CI : (0.7565, 0.8479)
    No Information Rate : 0.6786
    P-Value [Acc > NIR] : 0.0000004781

                  Kappa : 0.5257

 Mcnemar's Test P-Value : 0.006706

            Sensitivity : 0.9091
            Specificity : 0.5859
         Pos Pred Value : 0.8225
         Neg Pred Value : 0.7532
             Prevalence : 0.6786
         Detection Rate : 0.6169
   Detection Prevalence : 0.7500
      Balanced Accuracy : 0.7475

       'Positive' Class : 0
```

7.3 Classification and Regression Tree

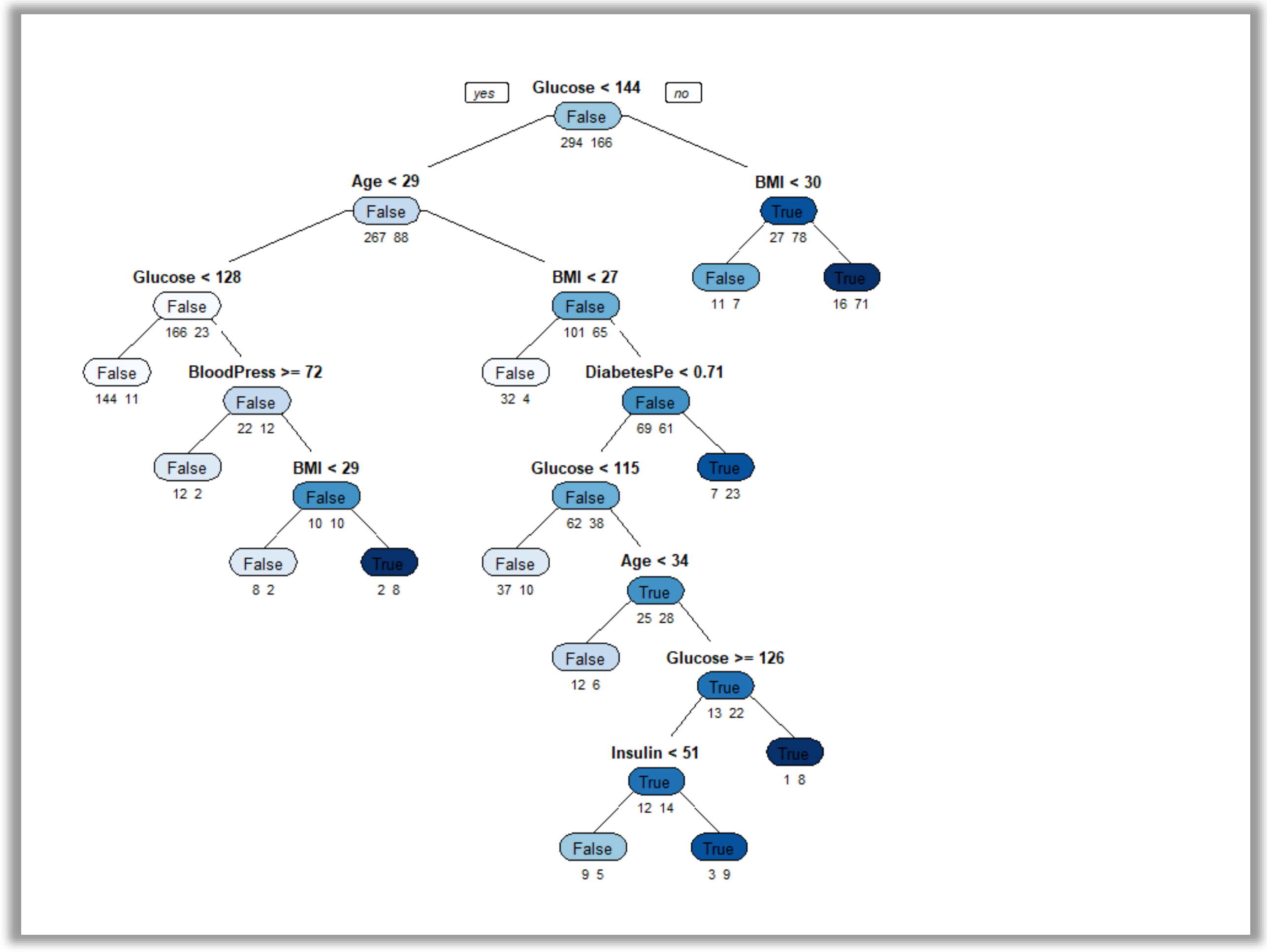
Classification trees are used for classification tasks and regression trees for prediction tasks. In both cases, the algorithm creates binary splits on the predictors that best classify/predict the outcome variable

Tree classification techniques, when they "work" and produce accurate predictions or predicted classifications based on few logical if-then conditions, have several advantages over many of those alternative techniques.

Simplicity of results: In most cases, the interpretation of results summarized in a tree is very simple. This simplicity is useful not only for purposes of rapid classification of new observations (it is much easier to evaluate just one or two logical conditions, than to compute classification scores for each possible group, or predicted values, based on all predictors and using possibly some complex nonlinear model equations), but can also often yield a much simpler model for explaining why observations are classified or predicted in a particular manner.

Tree methods are nonparametric and nonlinear: The results of using tree methods for classification or regression can be summarized in a series of logical if-then conditions (tree nodes). Therefore, there is no implicit assumption that the underlying relationships between the predictor variables and the dependent variable are linear, follow some specific non-linear link function or that they are even monotonic in nature.

OUTPUT – DECISION TREE:



If the condition is satisfied, we go to right, and if not then go to left.

Decide the node based on number of 0s and 1s. Whichever is greater the outcome will be according to that.

From above tree, we can traverse as below:

If Glucose < 155

If Glucose >= 104

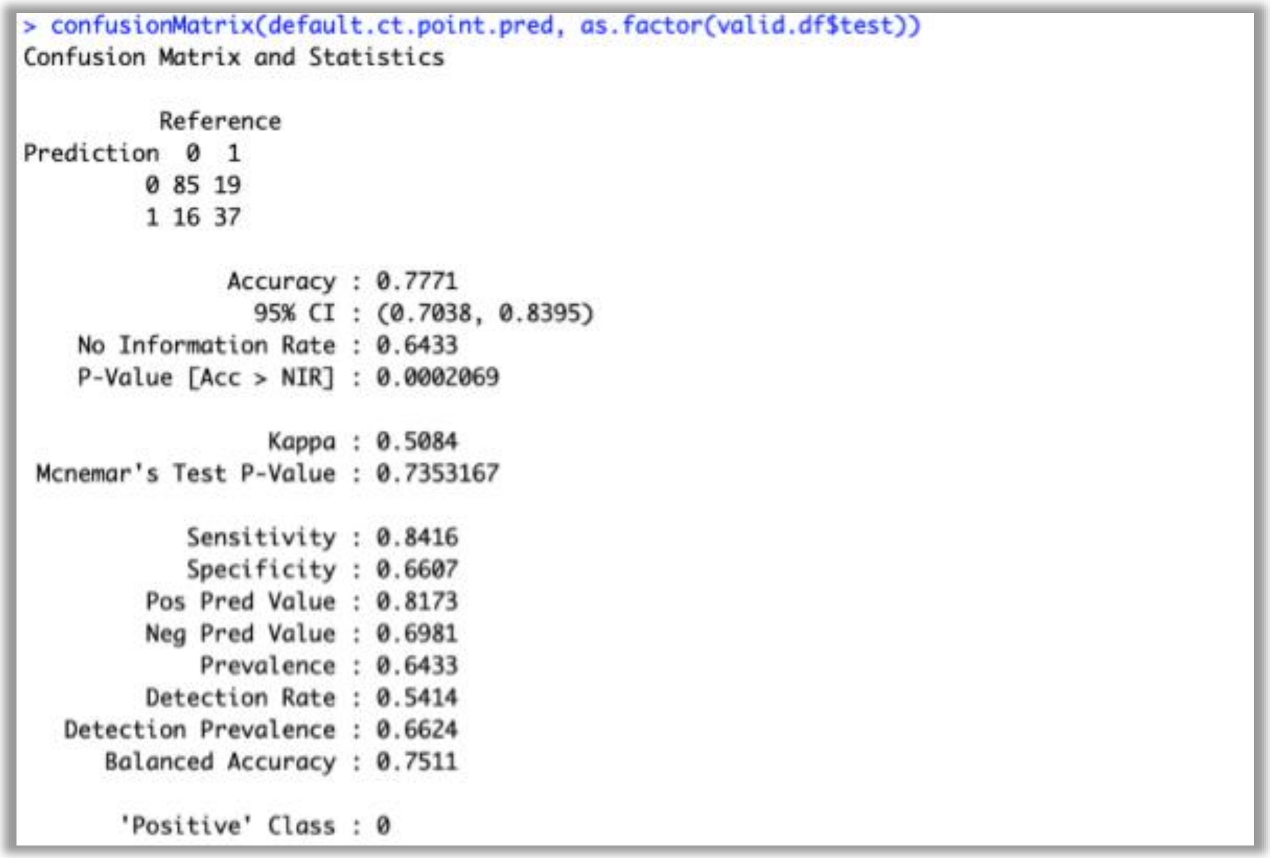
If Age >= 29

If Pregnant >=2

If Diastolic <77

NO Diabetics

OUTPUT OF CONFUSION MATRIX :



```
> confusionMatrix(default.ct.point.pred, as.factor(valid.df$test))
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 85 19
         1 16 37

               Accuracy : 0.7771
                 95% CI : (0.7038, 0.8395)
    No Information Rate : 0.6433
    P-Value [Acc > NIR] : 0.0002069

                  Kappa : 0.5084
 Mcnemar's Test P-Value : 0.7353167

            Sensitivity : 0.8416
            Specificity : 0.6607
         Pos Pred Value : 0.8173
         Neg Pred Value : 0.6981
             Prevalence : 0.6433
         Detection Rate : 0.5414
   Detection Prevalence : 0.6624
      Balanced Accuracy : 0.7511

       'Positive' Class : 0
```

7.4 CLUSTERING

Clustering is a technique of data segmentation that partitions the data into several groups based on their similarity.
Basically, we group the data through a statistical operation. These smaller groups that are formed from the bigger data are known as clusters. These cluster exhibit the following properties:

- They are discovered while carrying out the operation and the knowledge of their number is not known in advance.
- Clusters are the aggregation of similar objects that share common characteristics.

K-Means Clustering

One of the oldest methods of cluster analysis is known as k-means cluster. The first step when using k-means cluster analysis is to specify the number of clusters (k) that will be formed in the final solution.

The algorithm works as below:

- The process begins by choosing k observations to serve as centers for the clusters.
- The distance from each of the other observations is calculated for each of the k clusters, and observations are put in the cluster to which they are the closest.
- After each observation has been put in a cluster, the center of the clusters is recalculated, and every observation is checked to see if it might be closer to a different cluster, now that the centers have been recalculated. The process continues until no observations switch clusters.

Elbow Method:

To determine optimum number of clusters we use elbow method. The elbow method looks at the percentage of variance explained as a function of the number of clusters.We plot percentage of variance explained by the clusters against the number of clusters. We observe that the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the "elbow criterion".

Hierarchical Clustering

In k means clustering, it requires us to specify the number of clusters, and finding the optimal number of clusters can often be hard. Hierarchical clustering is an alternative approach which builds a hierarchy and doesn't require us to specify the number of clusters beforehand. There are two types of hierarchical clustering techniques:

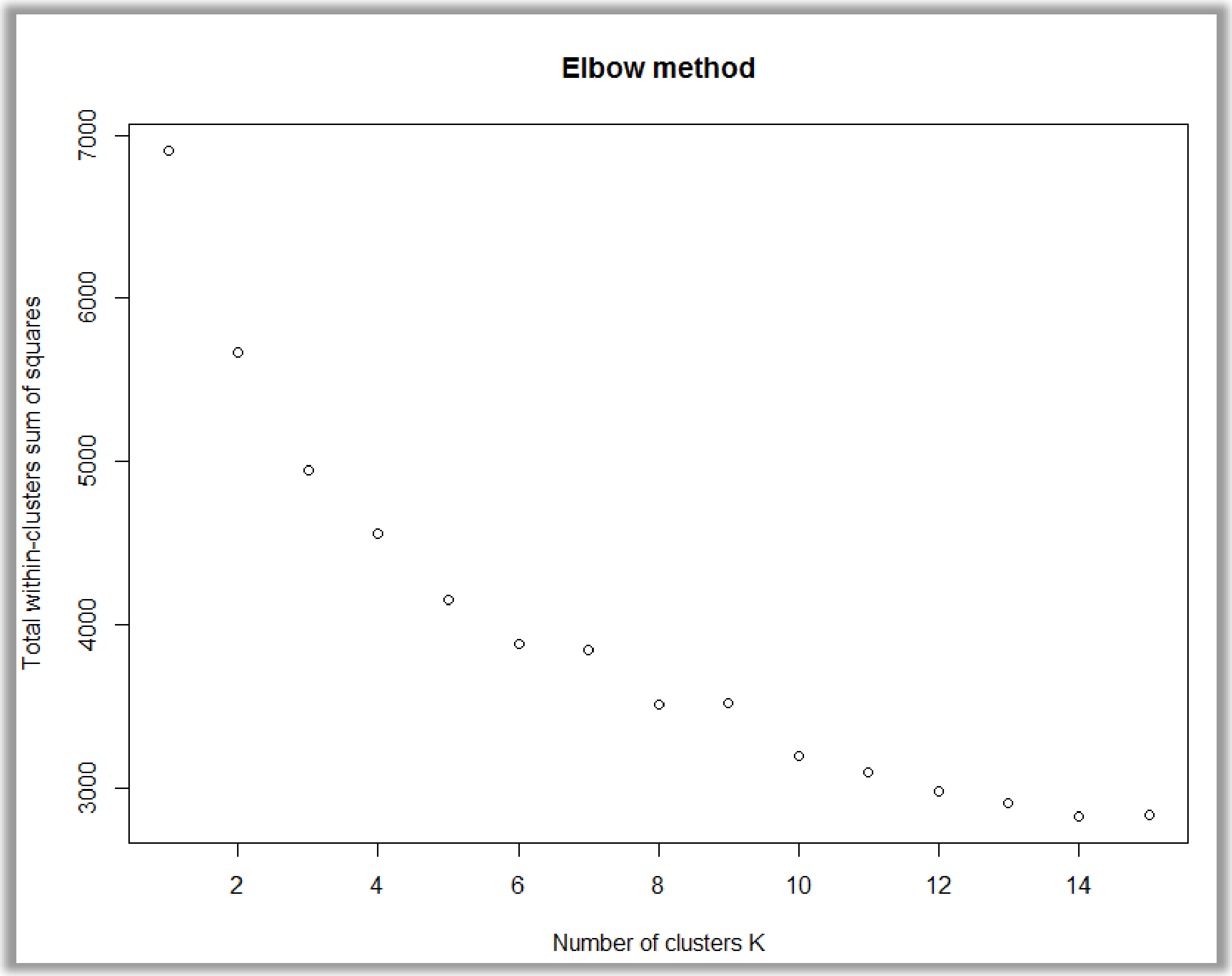
1. **Agglomerative Hierarchical clustering Technique:** In this technique, initially each data point is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster or K clusters are formed.

The algorithm works as follows:

- Compute the proximity matrix
- Let each data point be a cluster
- Repeat: Merge the two closest clusters and update the proximity matrix
- Until only a single cluster remains

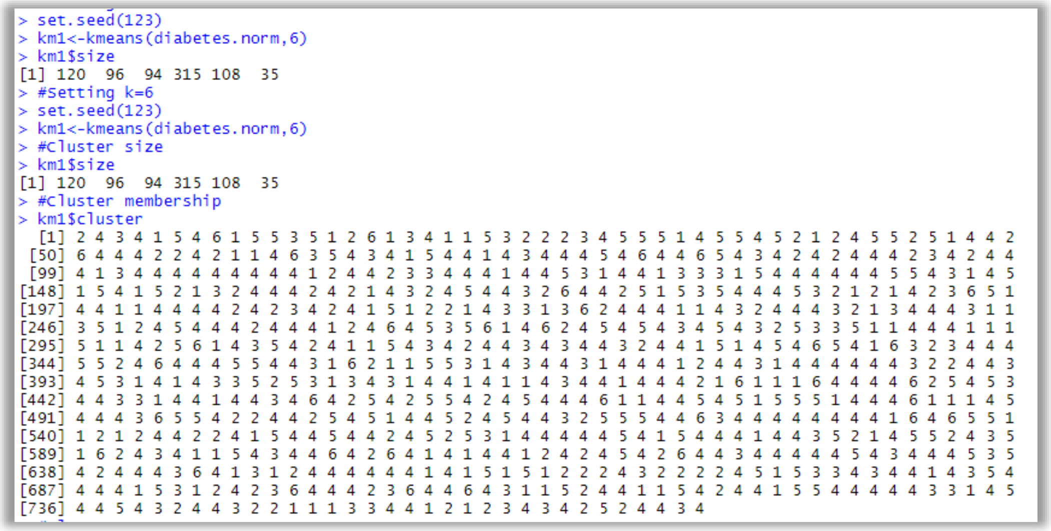
2. **Divisive Hierarchical clustering Technique:** Divisive Hierarchical clustering is exactly the opposite of the Agglomerative Hierarchical clustering. In Divisive Hierarchical clustering, we consider all the data points as a single cluster and in each iteration, we separate the data points from the cluster which are not similar. Each data point which is separated is considered as an individual cluster. In the end, we'll be left with n clusters.

OUTPUT FOR ELBOW METHOD:

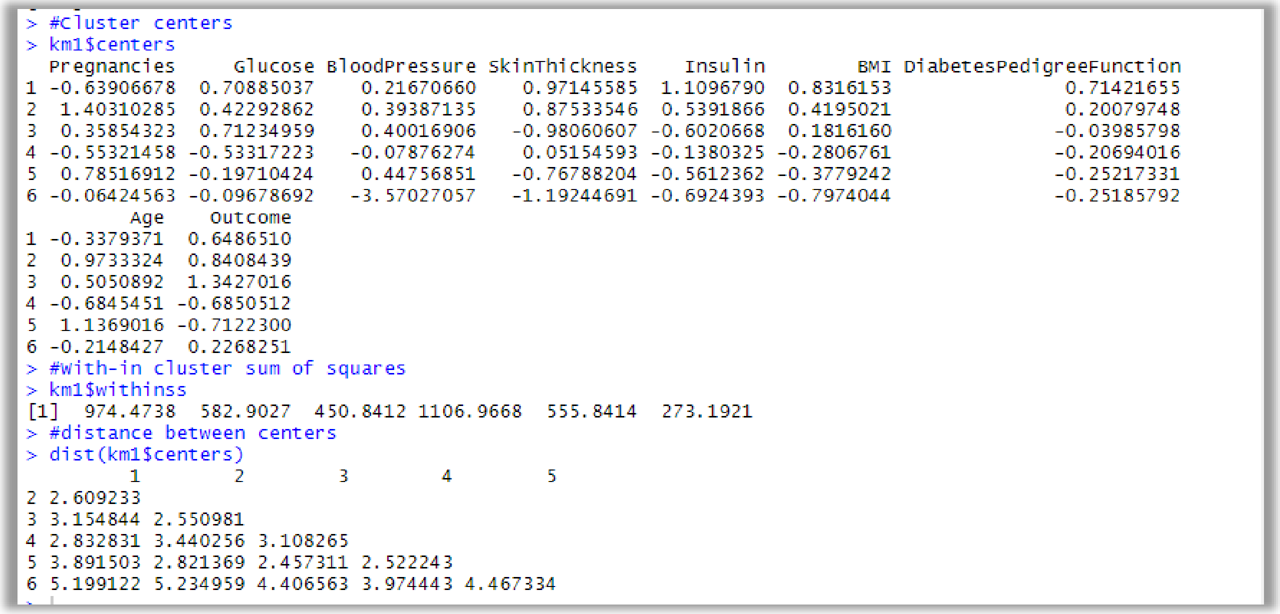


From above graph, for k=6 the between_ss/total_ss ratio tends to change slowly and remain less changing as compared to other. So for this data k=6 should be a good choice for number of clusters.

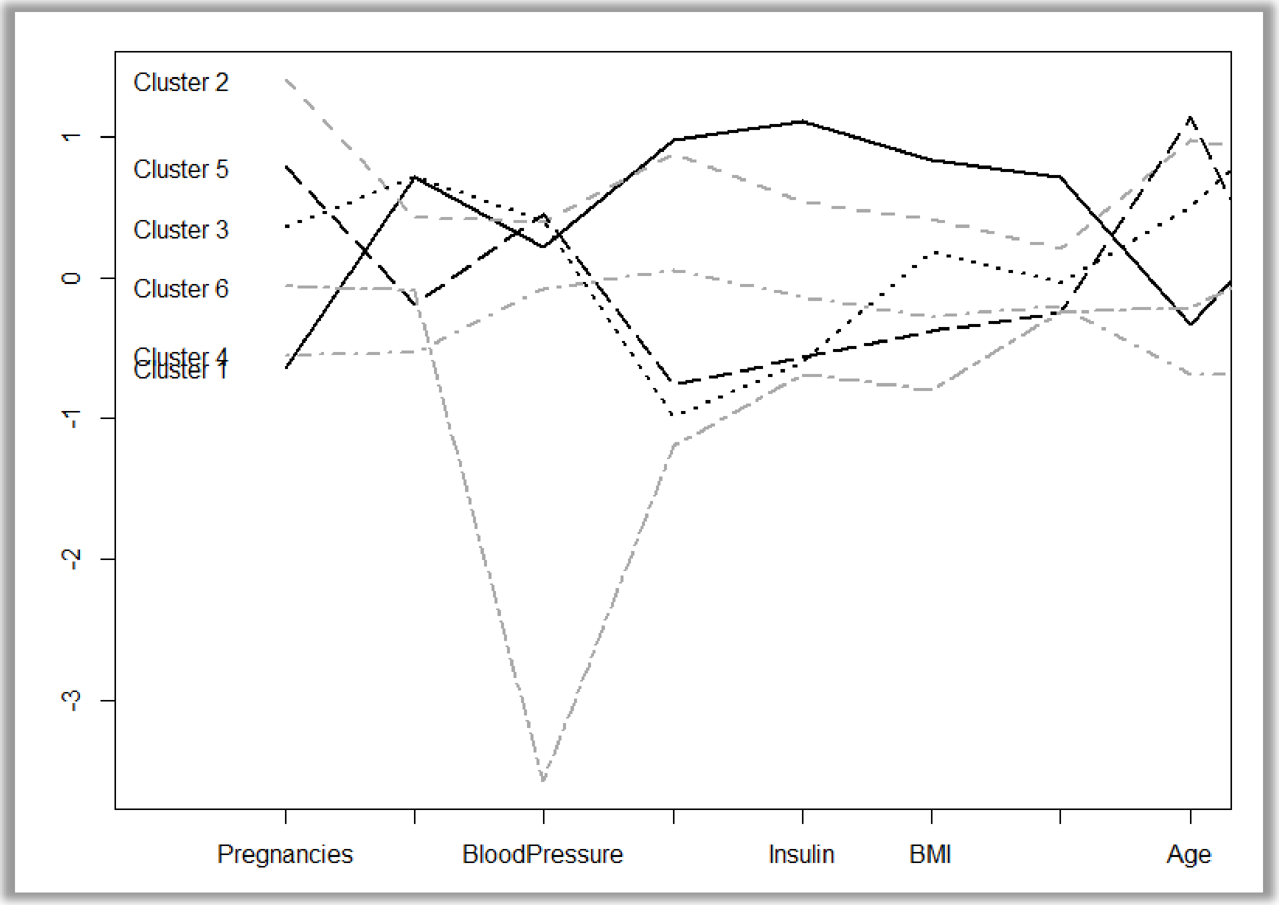
Output for K-Means Clustering:



```
> set.seed(123)
> km1<-kmeans(diabetes.norm,6)
> km1$size
[1] 120  96  94 315 108  35
> #Setting k=6
> set.seed(123)
> km1<-kmeans(diabetes.norm,6)
> #Cluster size
> km1$size
[1] 120  96  94 315 108  35
> #Cluster membership
> km1$cluster
  [1] 2 4 3 4 1 5 4 6 1 5 5 3 5 1 2 6 1 3 4 1 1 5 3 2 2 2 3 4 5 5 5 1 4 5 5 4 5 2 1 2 4 5 5 2 5 1 4 4 2
 [50] 6 4 4 4 2 2 4 2 1 1 4 6 3 5 4 3 4 1 5 4 4 1 4 3 4 4 4 5 4 6 4 4 6 5 4 3 4 2 4 2 4 4 4 2 3 4 2 4 4
 [99] 4 1 3 4 4 4 4 4 4 4 4 4 1 2 4 4 2 3 3 4 4 4 1 4 4 5 3 1 4 4 1 3 3 3 1 5 4 4 4 4 4 4 5 5 4 3 1 4 5
[148] 1 5 4 1 5 2 1 3 2 4 4 4 2 4 2 1 4 3 2 4 5 4 4 3 2 6 4 4 2 5 1 5 3 5 4 4 4 5 3 2 1 2 1 4 2 3 6 5 1
[197] 4 4 1 1 4 4 4 4 2 4 2 3 4 2 4 1 5 1 2 2 1 4 3 3 1 3 6 2 4 4 4 1 1 4 3 2 4 4 4 3 2 1 3 4 4 4 3 1 1
[246] 3 5 1 2 4 5 4 4 4 2 4 4 4 1 2 4 6 4 5 3 5 6 1 4 6 2 4 5 4 5 4 3 4 5 4 3 2 5 3 3 5 1 1 4 4 4 1 1 1
[295] 5 1 1 4 2 5 6 1 4 3 5 4 2 4 1 1 5 4 3 4 2 4 4 3 4 3 4 4 3 2 4 4 1 5 1 4 5 4 6 5 4 1 6 3 2 3 4 4 4
[344] 5 5 2 4 6 4 4 4 5 5 4 4 3 1 6 2 1 1 5 5 3 1 4 3 4 4 3 1 4 4 4 1 2 4 4 3 1 4 4 4 4 4 4 3 2 2 4 4 3
[393] 4 5 3 1 4 1 4 3 3 5 2 5 3 1 3 4 3 1 4 4 1 4 1 1 4 3 4 4 1 4 4 4 2 1 6 1 1 1 6 4 4 4 4 6 2 5 4 5 3
[442] 4 4 3 3 1 4 4 1 4 4 3 4 6 4 2 5 4 2 5 5 4 2 4 5 4 4 4 6 1 1 4 4 5 4 5 1 5 5 5 1 4 4 4 6 1 1 1 4 5
[491] 4 4 4 3 6 5 5 4 2 2 4 4 2 5 4 5 1 4 4 5 2 4 5 4 4 3 2 5 5 5 4 4 6 3 4 4 4 4 4 4 4 4 1 6 4 6 5 5 1
[540] 1 2 1 2 4 4 2 2 4 1 5 4 4 5 4 4 2 4 5 2 5 3 1 4 4 4 4 4 5 4 1 5 4 4 4 1 4 4 3 5 2 1 4 5 5 2 4 3 5
[589] 1 6 2 4 3 4 1 1 5 4 3 4 4 6 4 2 6 4 1 4 1 4 4 1 2 4 2 4 5 4 2 6 4 4 3 4 4 4 4 4 5 4 3 4 4 4 5 3 5
[638] 4 2 4 4 4 3 6 4 1 3 1 2 4 4 4 4 4 4 1 4 1 5 1 5 1 2 2 2 4 3 2 2 2 2 4 5 1 5 3 3 4 3 4 4 1 4 3 5 4
[687] 4 4 4 1 5 3 1 2 4 2 3 6 4 4 4 2 3 6 4 4 6 4 3 1 1 5 2 4 4 1 1 5 4 2 4 4 1 5 5 4 4 4 4 4 3 3 1 4 5
[736] 4 4 5 4 3 2 4 4 3 2 2 1 1 1 3 3 4 4 1 2 1 2 3 4 3 4 2 5 2 4 4 3 4
```

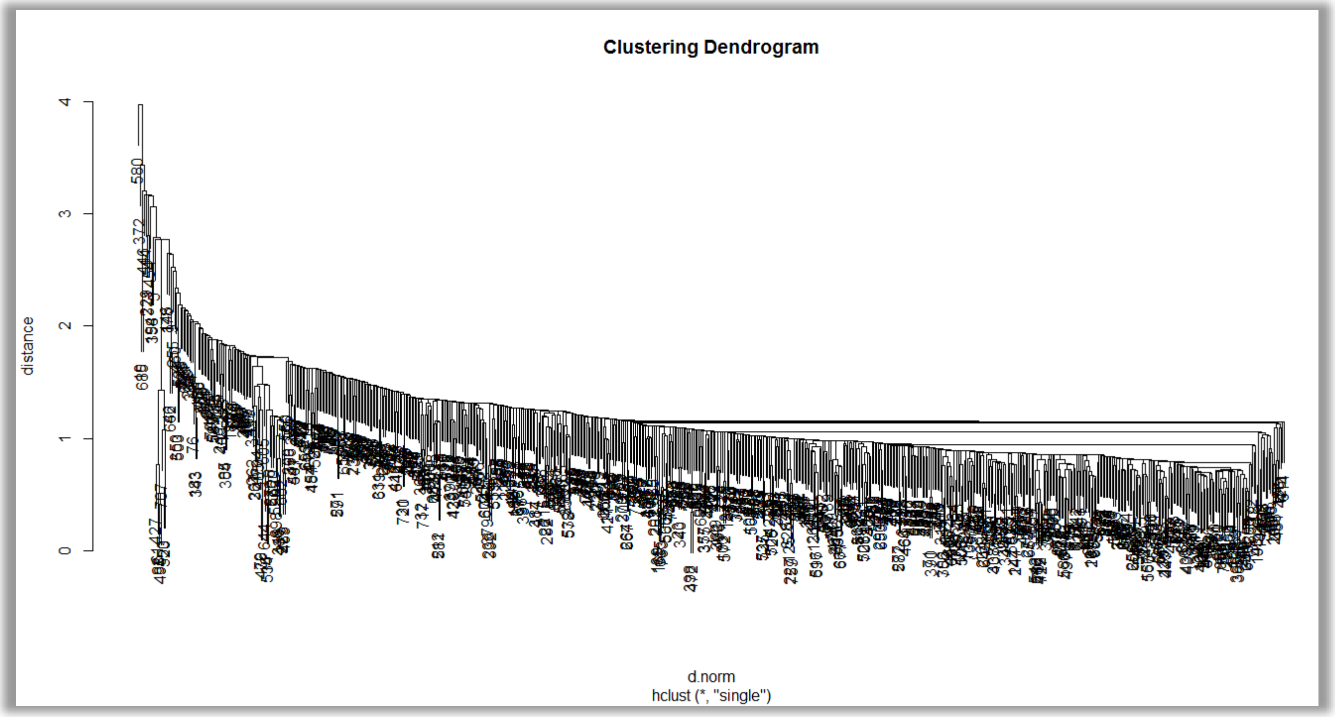


```
> #Cluster centers
> km1$centers
  Pregnancies     Glucose BloodPressure SkinThickness    Insulin        BMI DiabetesPedigreeFunction
1 -0.63906678  0.70885037    0.21670660    0.97145585  1.1096790  0.8316153               0.71421655
2  1.40310285  0.42292862    0.39387135    0.87533546  0.5391866  0.4195021               0.20079748
3  0.35854323  0.71234959    0.40016906   -0.98060607 -0.6020668  0.1816160              -0.03985798
4 -0.55321458 -0.53317223   -0.07876274    0.05154593 -0.1380325 -0.2806761              -0.20694016
5  0.78516912 -0.19710424    0.44756851   -0.76788204 -0.5612362 -0.3779242              -0.25217331
6 -0.06424563 -0.09678692   -3.57027057   -1.19244691 -0.6924393 -0.7974044              -0.25185792
         Age    Outcome
1 -0.3379371  0.6486510
2  0.9733324  0.8408439
3  0.5050892  1.3427016
4 -0.6845451 -0.6850512
5  1.1369016 -0.7122300
6 -0.2148427  0.2268251
> #With-in cluster sum of squares
> km1$withinss
[1]  974.4738  582.9027  450.8412 1106.9668  555.8414  273.1921
> #distance between centers
> dist(km1$centers)
         1        2        3        4        5
2 2.609233
3 3.154844 2.550981
4 2.832831 3.440256 3.108265
5 3.891503 2.821369 2.457311 2.522243
6 5.199122 5.234959 4.406563 3.974443 4.467334
```

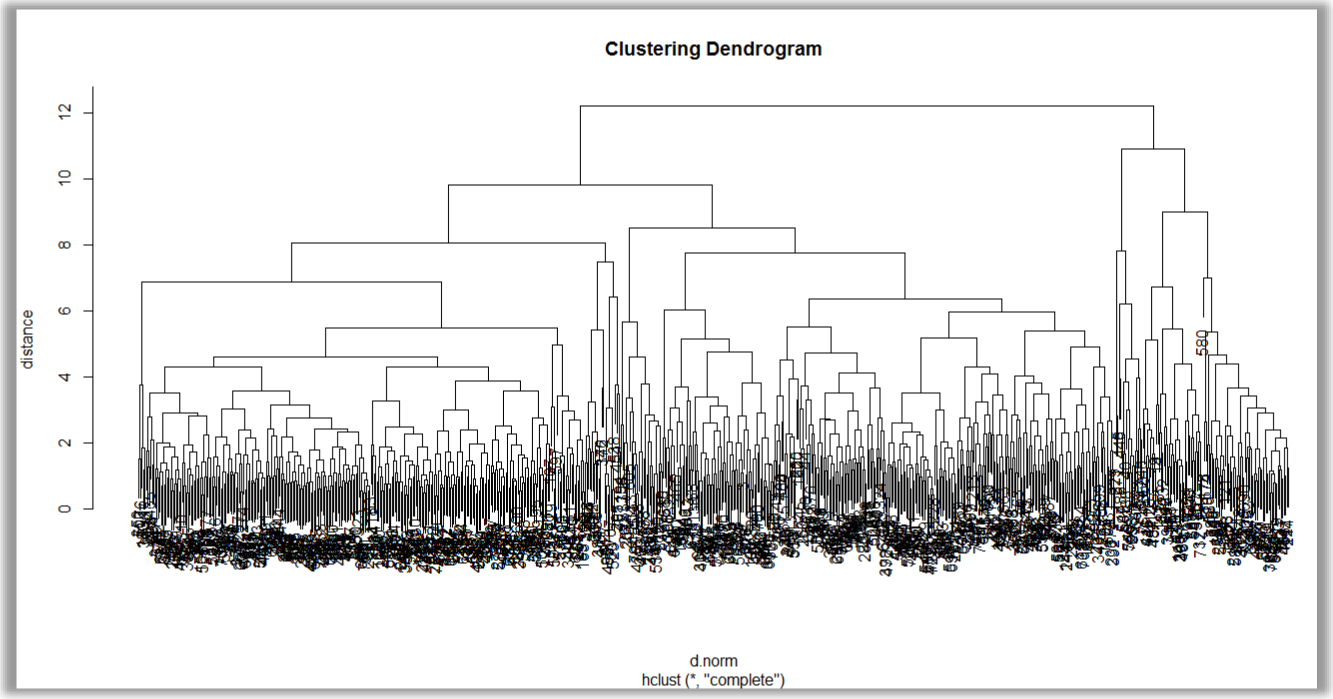


Output for Hierarchical Clustering:

Single Linkage:



Complete Linkage:

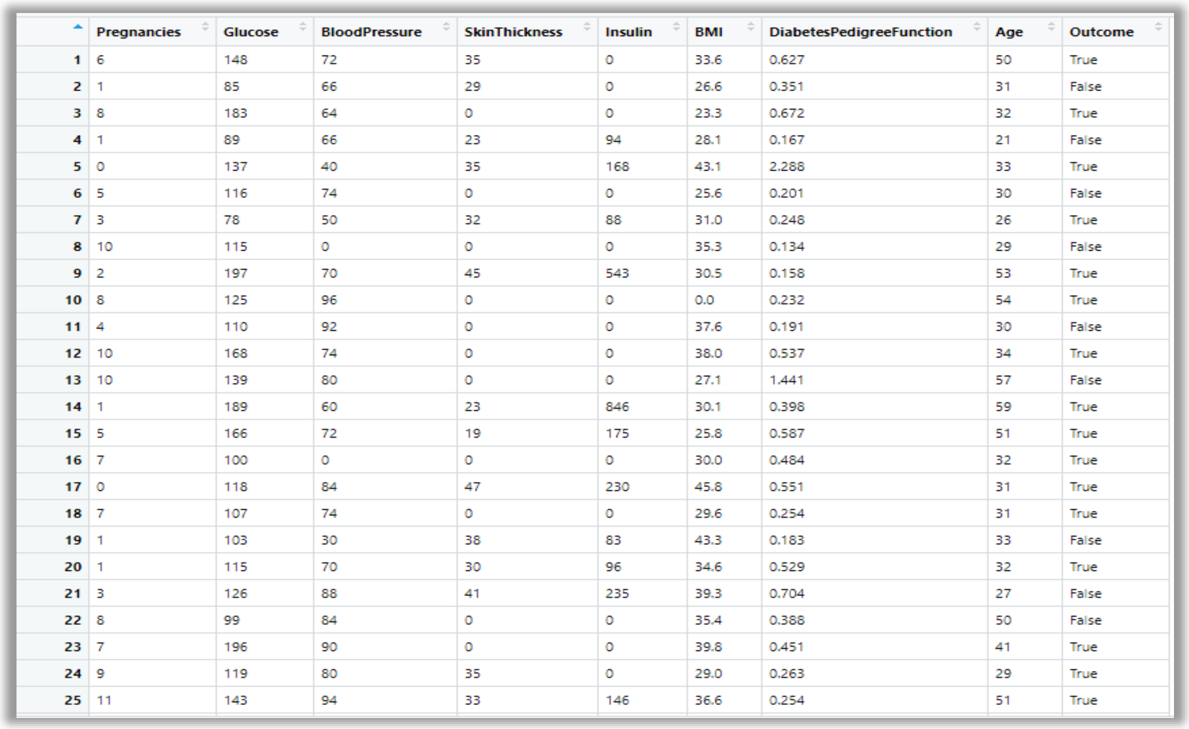


7.5 NAÏVE BAYES

Naive Bayes is a supervised machine learning algorithm that is based on Bayes theorem. We use Bayes theorem to solve classification problems by following a probabilistic approach. It is based on the idea that the predictor variables are independent of each other.
But, in real world problems, we know that the predictor variables are not independent of each other and that there is correlation that exists. The Bayes theorem is based on the principle of conditional probability, which is nothing but the probability of occurrence of an event given on the past event.

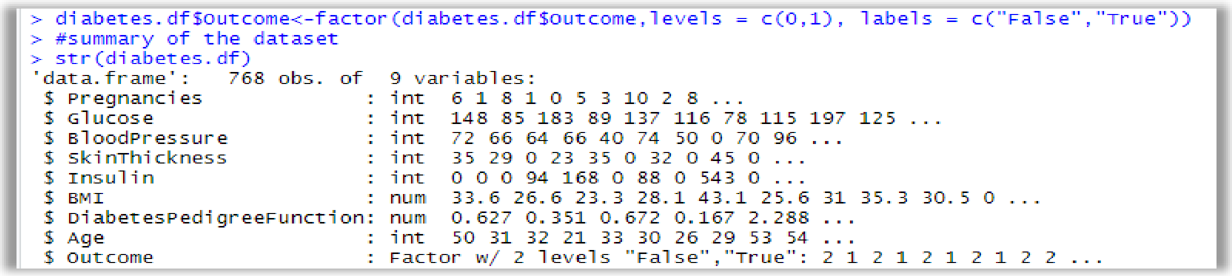
We make use of this algorithm to work on the dataset to predict if the patient is diabetic or normal.

Below is the glimpse of the dataset before we can explore more on it.



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	6	148	72	35	0	33.6	0.627	50	True
2	1	85	66	29	0	26.6	0.351	31	False
3	8	183	64	0	0	23.3	0.672	32	True
4	1	89	66	23	94	28.1	0.167	21	False
5	0	137	40	35	168	43.1	2.288	33	True
6	5	116	74	0	0	25.6	0.201	30	False
7	3	78	50	32	88	31.0	0.248	26	True
8	10	115	0	0	0	35.3	0.134	29	False
9	2	197	70	45	543	30.5	0.158	53	True
10	8	125	96	0	0	0.0	0.232	54	True
11	4	110	92	0	0	37.6	0.191	30	False
12	10	168	74	0	0	38.0	0.537	34	True
13	10	139	80	0	0	27.1	1.441	57	False
14	1	189	60	23	846	30.1	0.398	59	True
15	5	166	72	19	175	25.8	0.587	51	True
16	7	100	0	0	0	30.0	0.484	32	True
17	0	118	84	47	230	45.8	0.551	31	True
18	7	107	74	0	0	29.6	0.254	31	True
19	1	103	30	38	83	43.3	0.183	33	False
20	1	115	70	30	96	34.6	0.529	32	True
21	3	126	88	41	235	39.3	0.704	27	False
22	8	99	84	0	0	35.4	0.388	50	False
23	7	196	90	0	0	39.8	0.451	41	True
24	9	119	80	35	0	29.0	0.263	29	True
25	11	143	94	33	146	36.6	0.254	51	True

We then convert the outcome variable (predictor variable that signifies whether a person is diabetic or normal) to categorical variable.

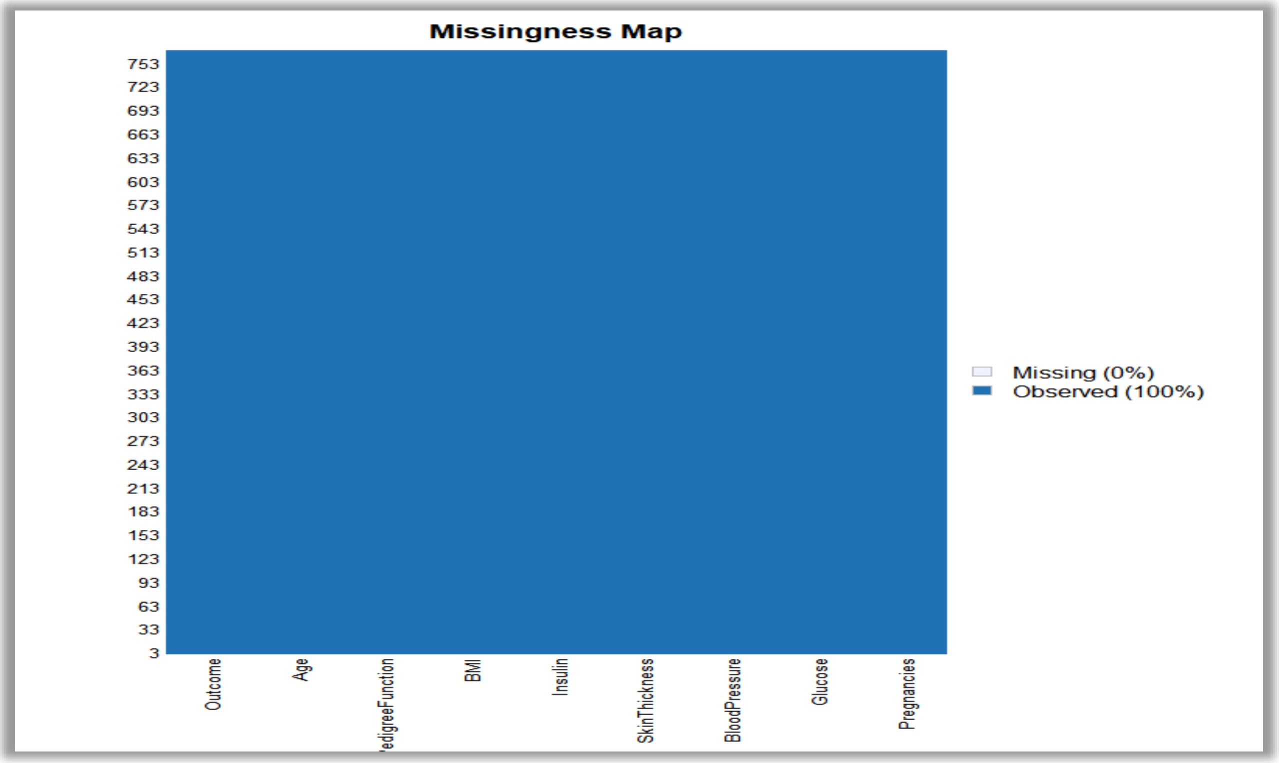


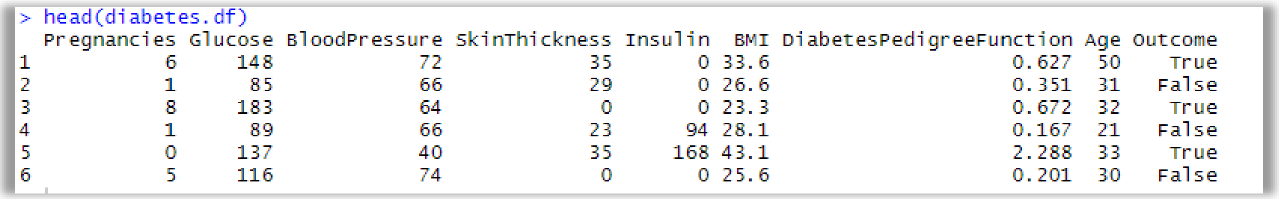
```
> diabetes.df$Outcome<-factor(diabetes.df$Outcome,levels = c(0,1), labels = c("False","True"))
> #summary of the dataset
> str(diabetes.df)
'data.frame':	768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : Factor w/ 2 levels "False","True": 2 1 2 1 2 1 2 1 2 2 ...
```

We can see from the structure of the dataset that the desired variable has been converted into a categorical variable.

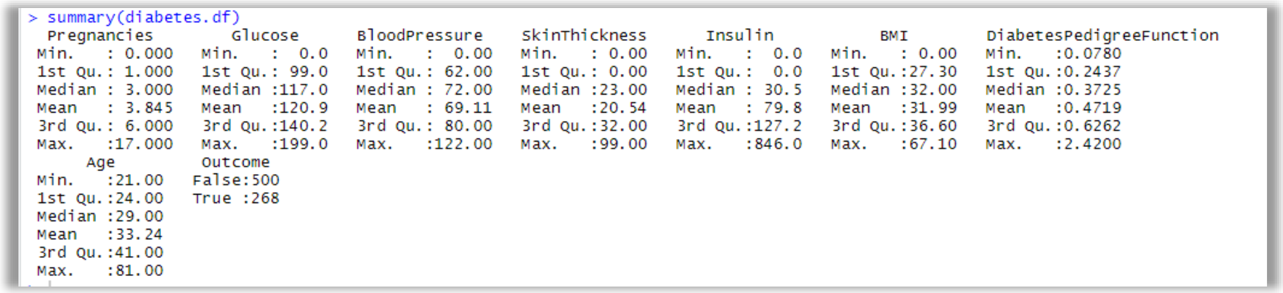
Further, we can check on the summary of the dataset to obtain some insights. We can see that there are no missing values as we have already completed our data cleaning process on the dataset.

OUTPUT : showing the missingness percentage in the dataset





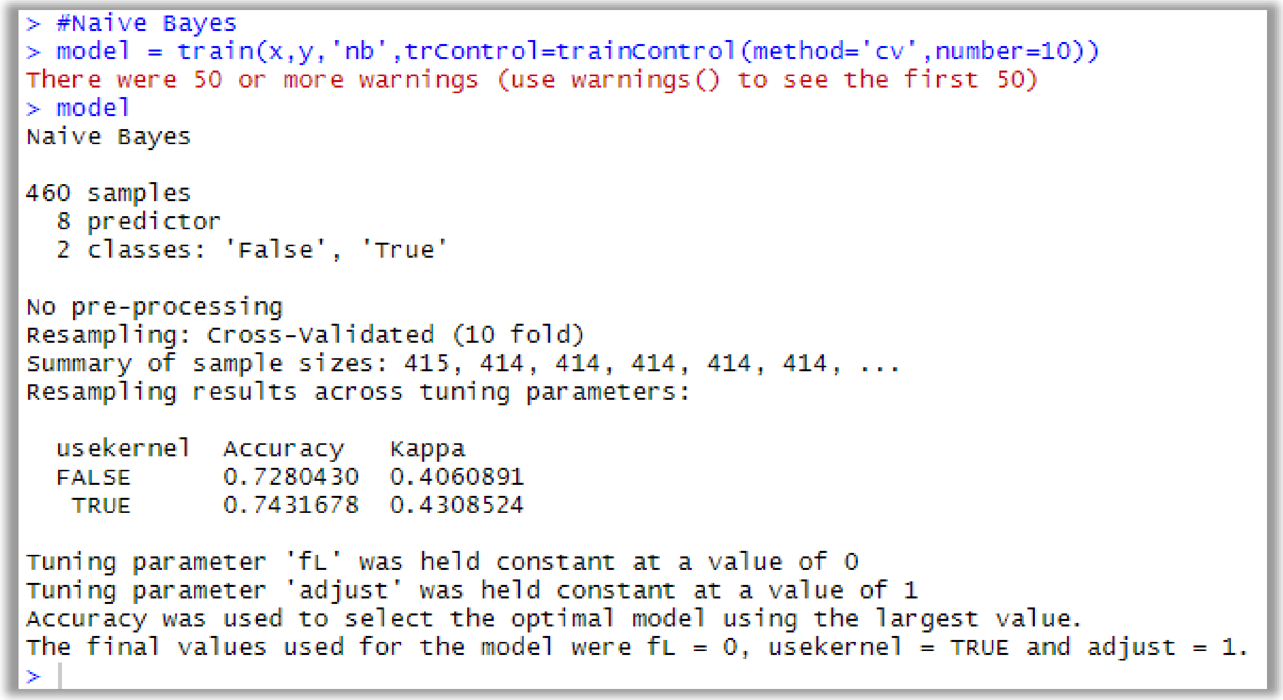
```
> head(diabetes.df)
  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI DiabetesPedigreeFunction Age Outcome
1           6     148            72            35       0 33.6                    0.627  50    True
2           1      85            66            29       0 26.6              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



```
> summary(diabetes.df)
  Pregnancies        Glucose      BloodPressure     SkinThickness      Insulin           BMI        DiabetesPedigreeFunction
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0   Min.   : 0.00   Min.   :0.0780
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00   1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5   Median :32.00   Median :0.3725
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8   Mean   :31.99   Mean   :0.4719
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.0   Max.   :67.10   Max.   :2.4200
      Age          Outcome
 Min.   :21.00   False:500
 1st Qu.:24.00   True :268
 Median :29.00
 Mean   :33.24
 3rd Qu.:41.00
 Max.   :81.00
```

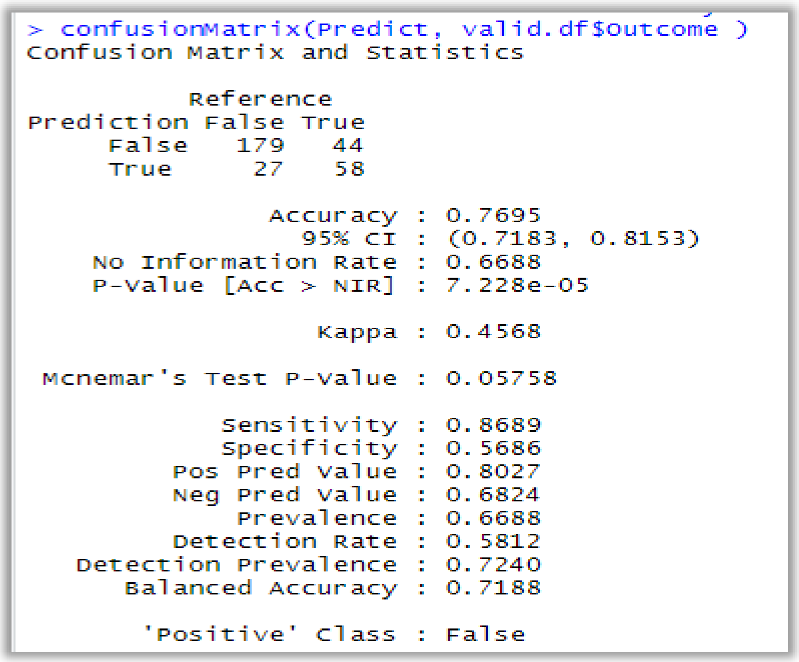
We can splice the data consisting of two part,

1. Training set
2. Test set



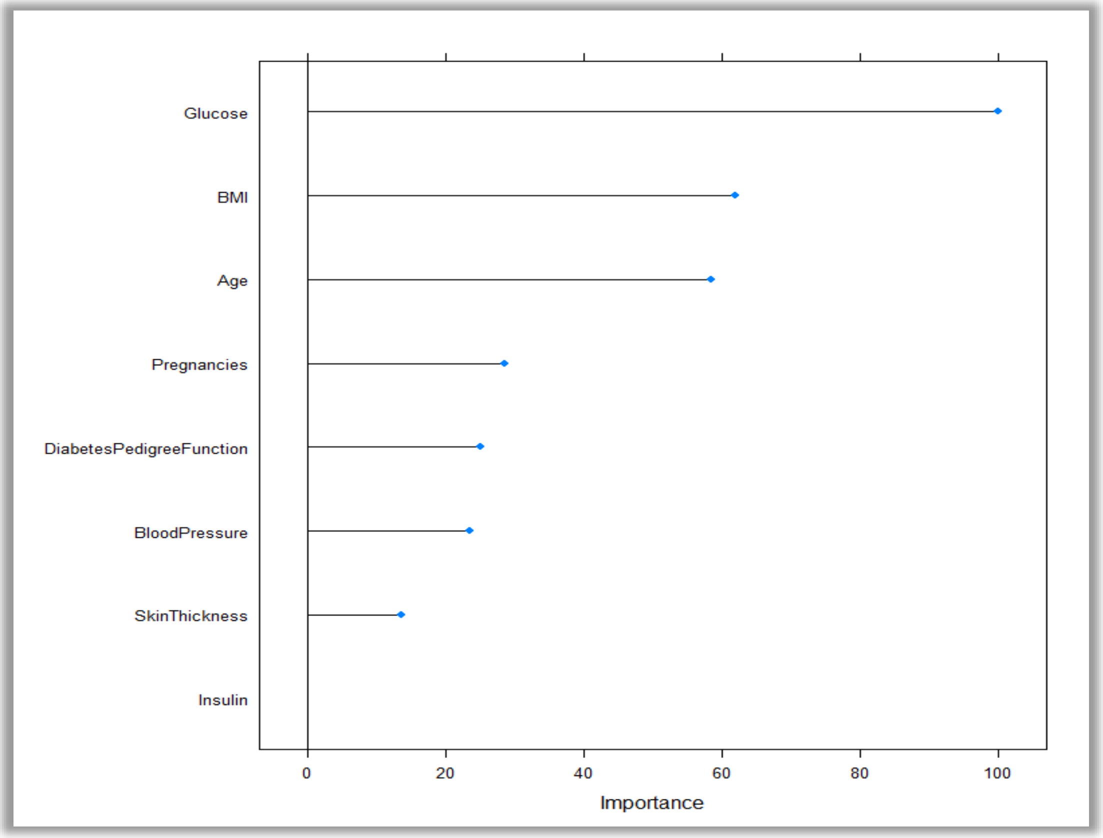
```
> #Naive Bayes
> model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))
There were 50 or more warnings (use warnings() to see the first 50)
> model
Naive Bayes

460 samples
  8 predictor
  2 classes: 'False', 'True'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 415, 414, 414, 414, 414, 414, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.7280430  0.4060891
   TRUE      0.7431678  0.4308524

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
>
```

To check the efficiency of the model, we are now going to run the testing data set on the model, after which we will evaluate the accuracy of the model by using a Confusion matrix.



```
> confusionMatrix(Predict, valid.df$Outcome )
Confusion Matrix and Statistics

          Reference
Prediction False True
     False   179   44
     True     27   58

               Accuracy : 0.7695
                 95% CI : (0.7183, 0.8153)
    No Information Rate : 0.6688
    P-Value [Acc > NIR] : 7.228e-05

                  Kappa : 0.4568

 Mcnemar's Test P-Value : 0.05758

            Sensitivity : 0.8689
            Specificity : 0.5686
         Pos Pred Value : 0.8027
         Neg Pred Value : 0.6824
             Prevalence : 0.6688
         Detection Rate : 0.5812
   Detection Prevalence : 0.7240
      Balanced Accuracy : 0.7188

       'Positive' Class : False
```

OUTPUT : variable importance chart that shows glucose is predominant when compared to others

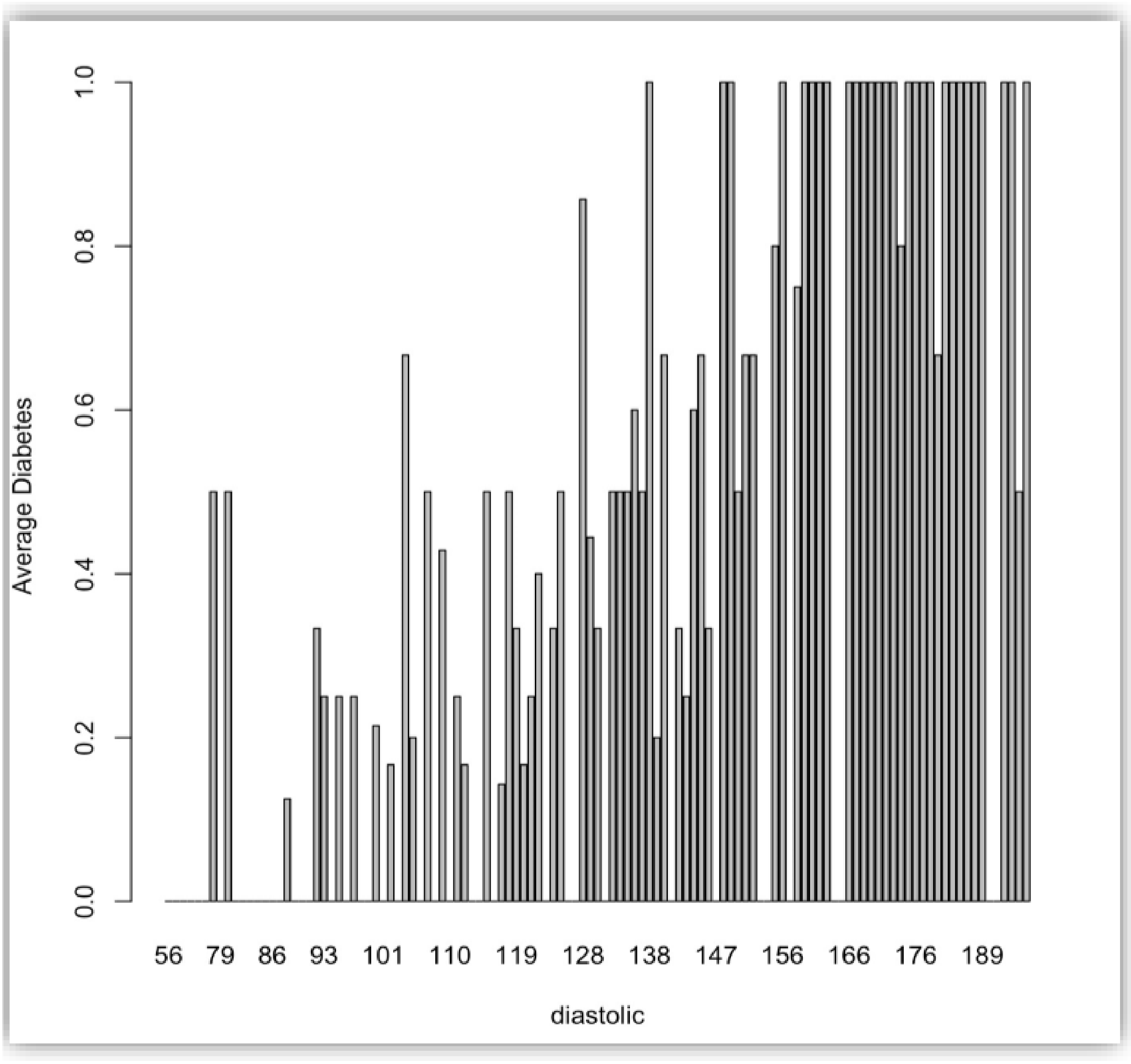


8. Data Visualization

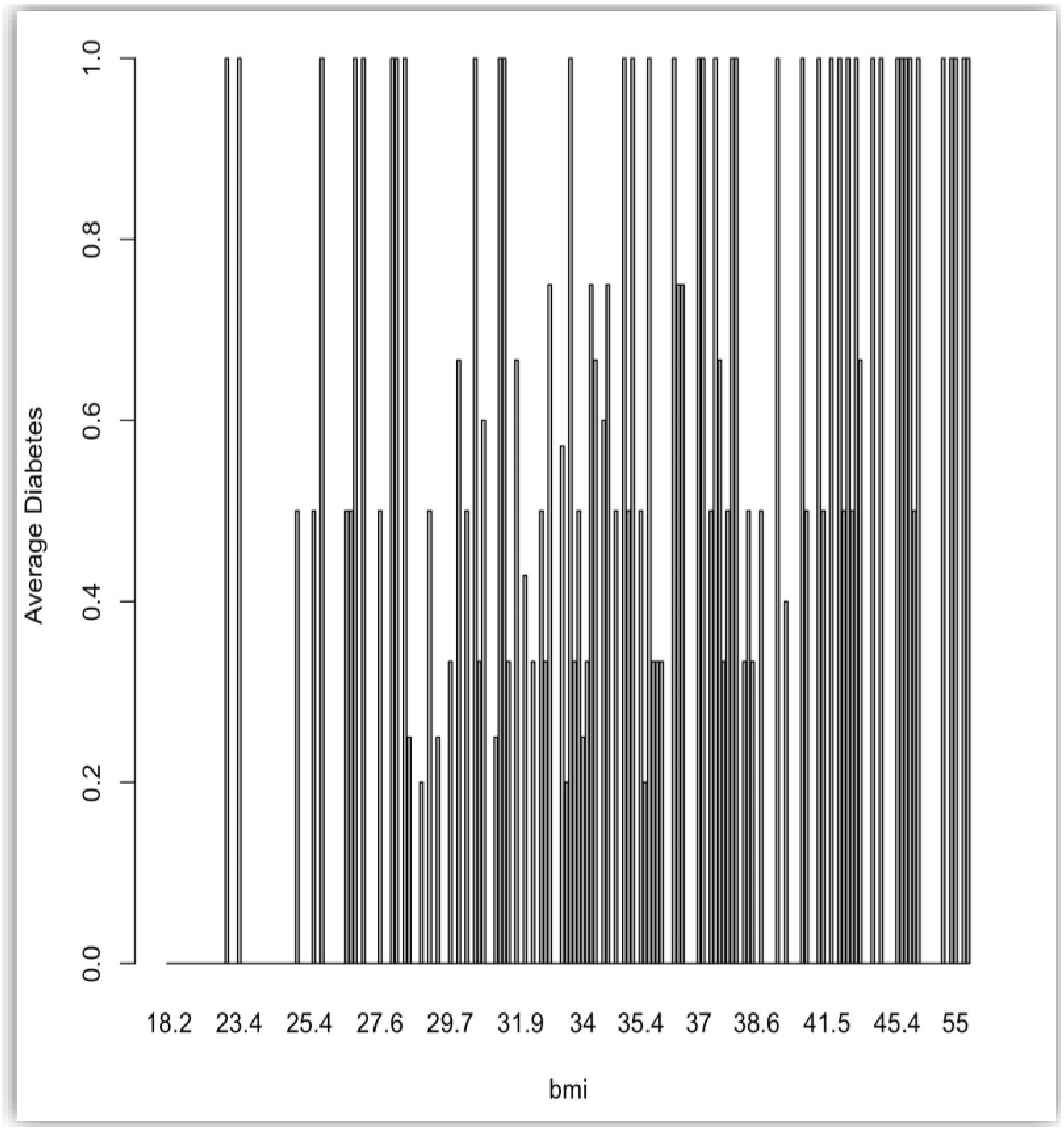
Visualizing the data is an important step of the data analysis. With a graphical visualization of the data we have a better understanding of the various features values distribution: for example, we can understand what's the average age of the people or the average BMI etc...

Bar Plots:

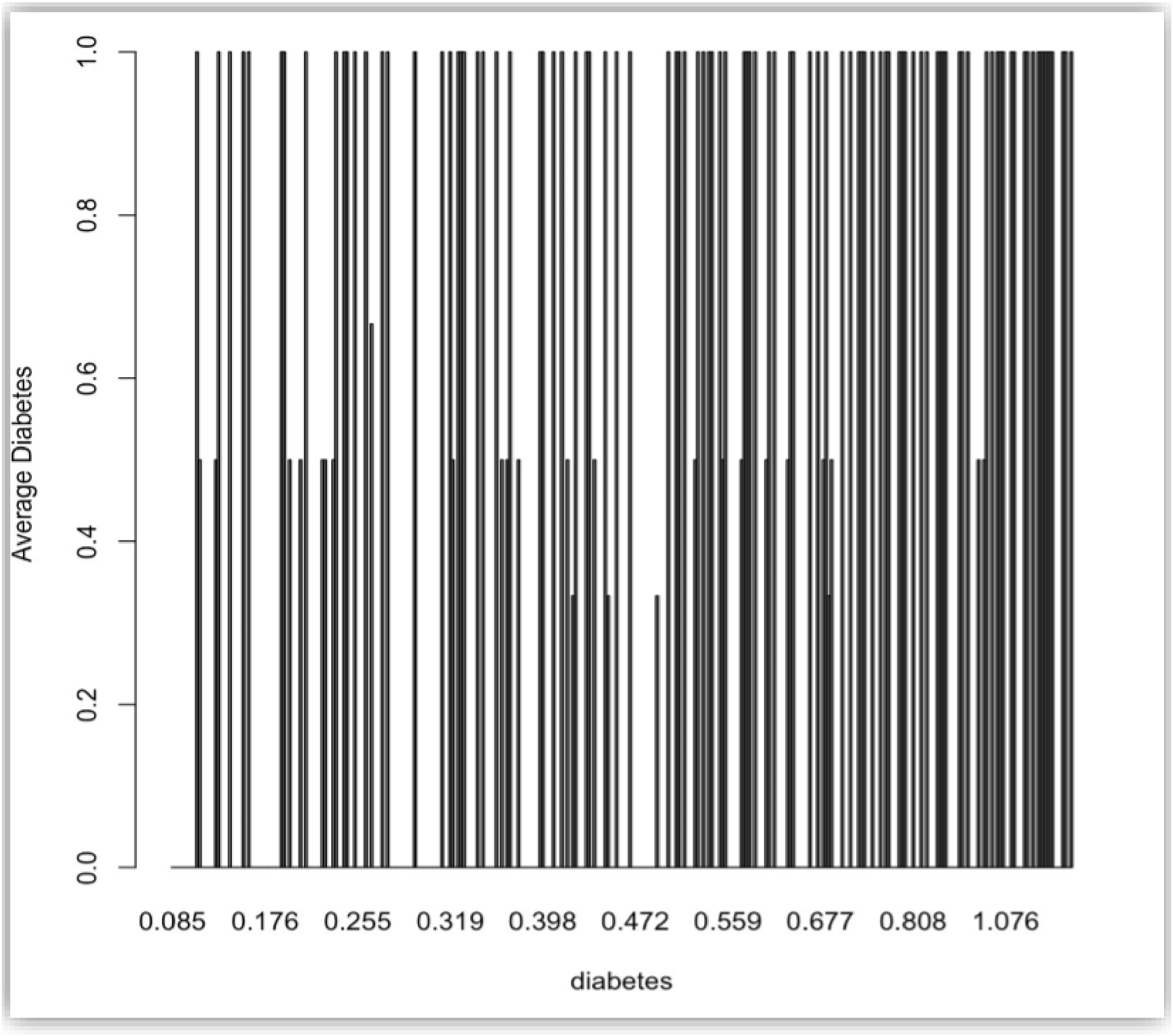
Following chart is showing the average Diastolic level:



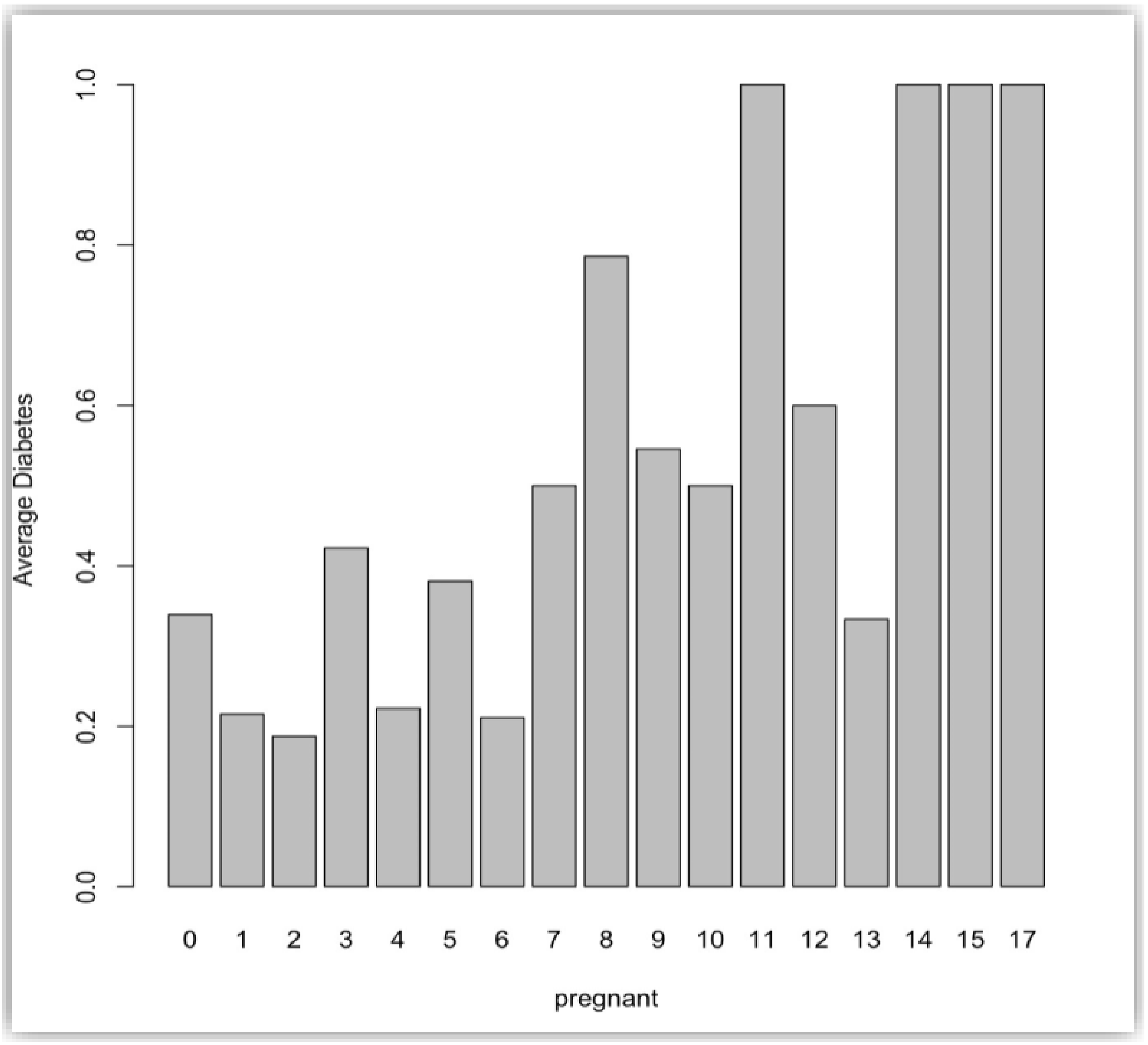
Following chart is showing the average BMI level:



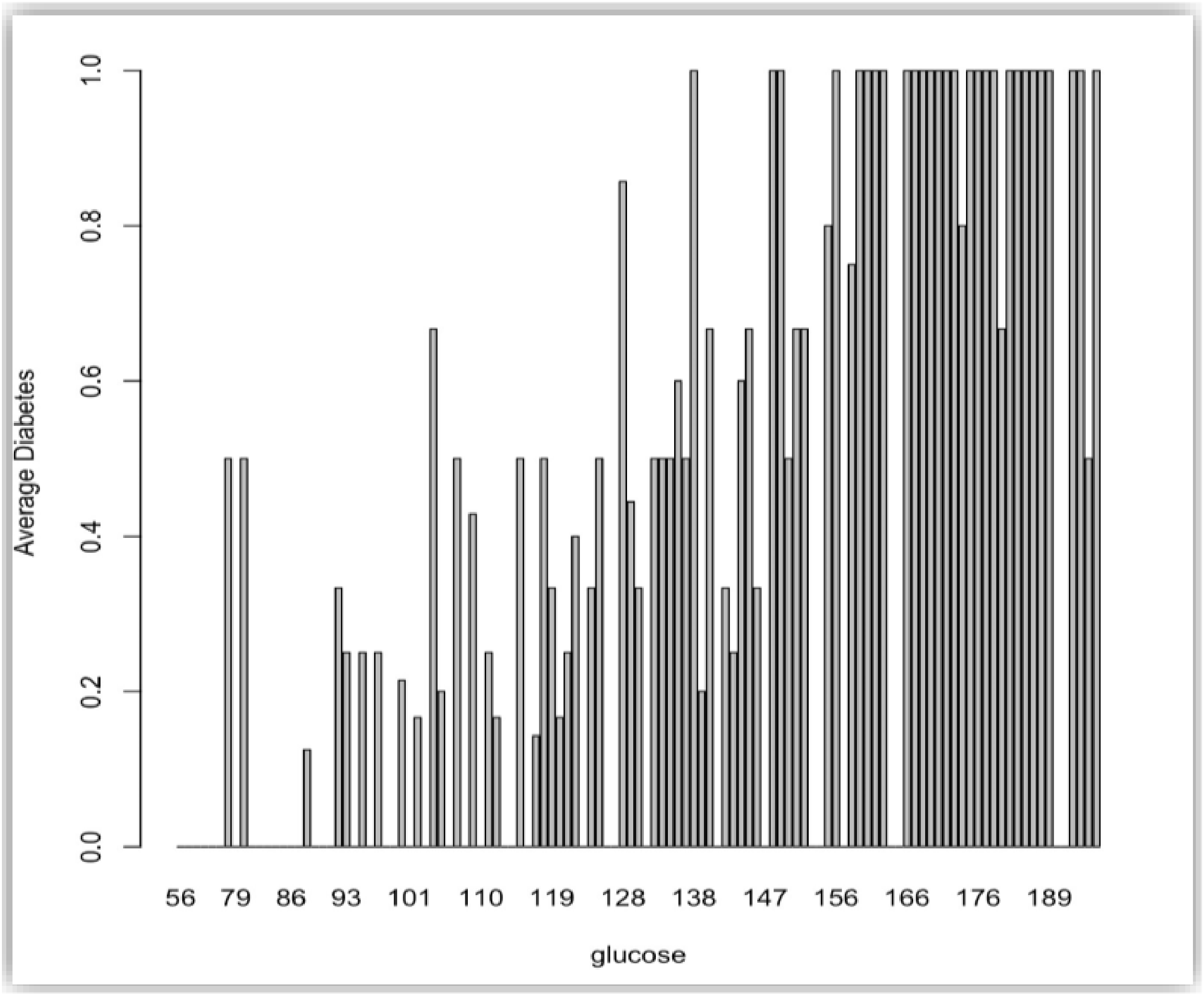
Following chart is showing the average Diabetes level:



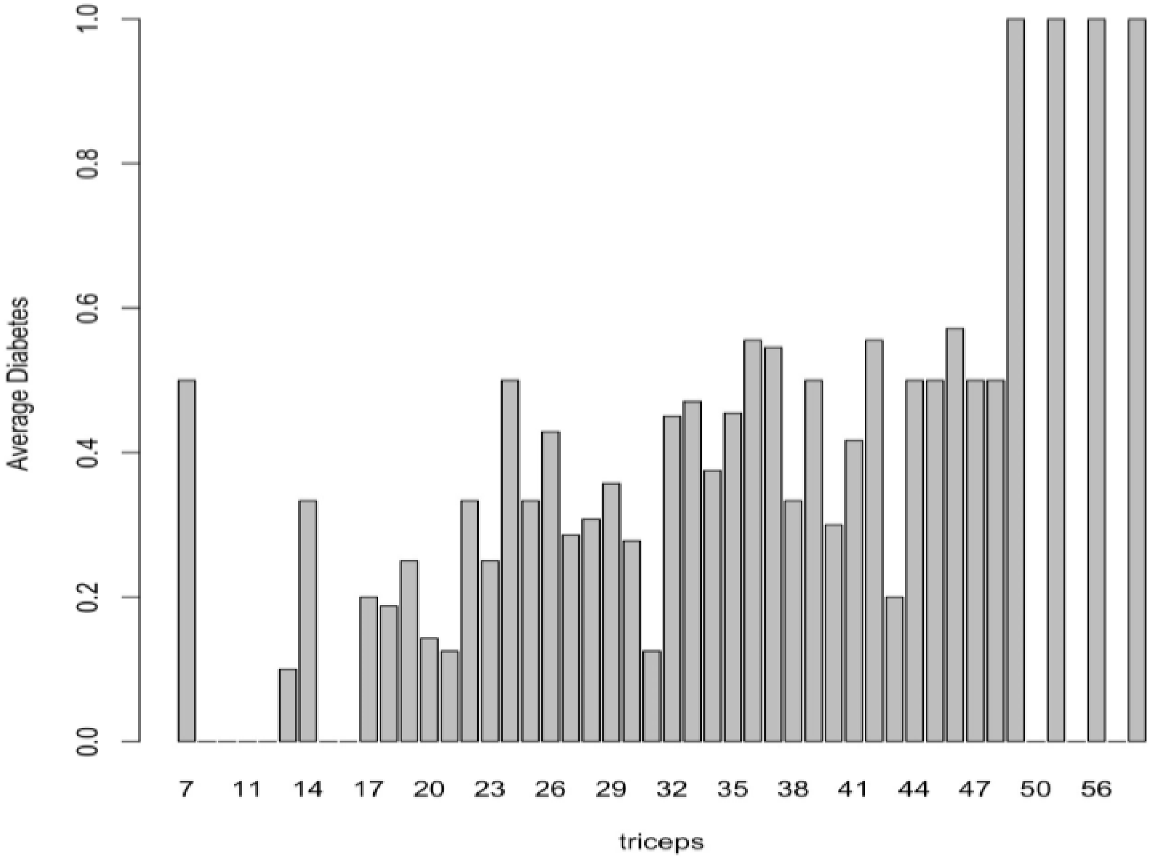
Following chart is showing the average Pregnant level:



Following chart is showing the average Glucose level:

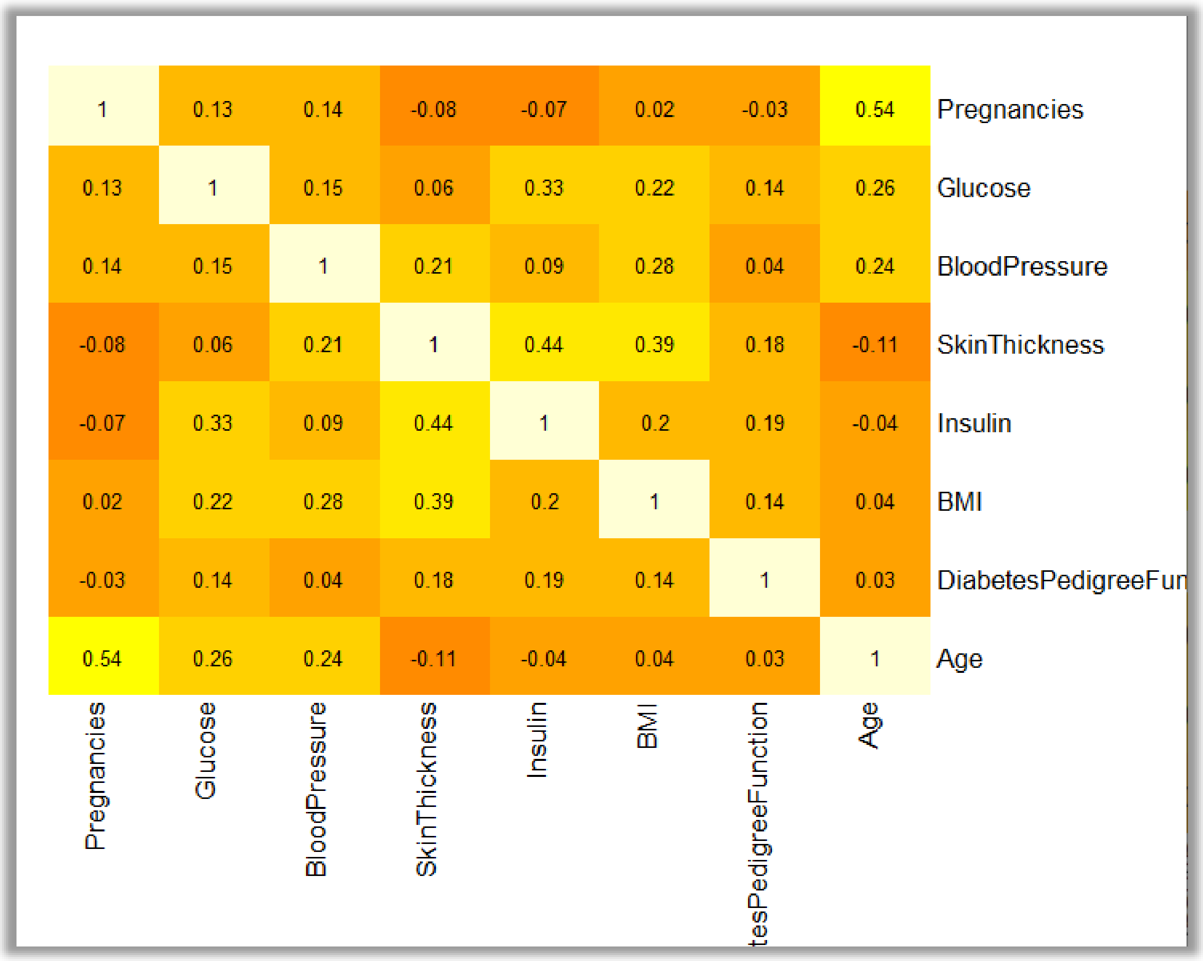


Following chart is showing the average Triceps (Skin Thickness) level:


Average Diabetes
0.0
0.2
0.4
0.6
0.8
1.0
7
11
14
17
20
23
26
29
32
35
38
41
44
47
50
56
triceps

Heatmap:

Heatmaps help to visualize correlation between variables. Darker shades correspond to stronger positive or negative correlation



Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	tesPedigreeFunction	Age	
1	0.13	0.14	-0.08	-0.07	0.02	-0.03	0.54	Pregnancies
0.13	1	0.15	0.06	0.33	0.22	0.14	0.26	Glucose
0.14	0.15	1	0.21	0.09	0.28	0.04	0.24	BloodPressure
-0.08	0.06	0.21	1	0.44	0.39	0.18	-0.11	SkinThickness
-0.07	0.33	0.09	0.44	1	0.2	0.19	-0.04	Insulin
0.02	0.22	0.28	0.39	0.2	1	0.14	0.04	BMI
-0.03	0.14	0.04	0.18	0.19	0.14	1	0.03	DiabetesPedigreeFun
0.54	0.26	0.24	-0.11	-0.04	0.04	0.03	1	Age

From the bellow heatmap, it shows that pregnant variable has stronger positive correlation with age variable and has weak negative correlation with BMI variable. Glucose variable has strong positive correlation with insulin variable and has weak correlation with diabetes variable

9. Validation

k-fold cross validation

Through the analysis we understood that:

1. We should train the model on a large portion of the dataset. Otherwise we'll fail to read and recognize the underlying trend in the data. This will eventually result in a higher bias.

2. We also need a good ratio of testing data points. As we have seen above, less amount of data points can lead to a variance error while testing the effectiveness of the model.

3. We should iterate on the training and testing process multiple times. We should change the train and test dataset distribution. This helps in validating the model effectiveness properly

To implement the K-Fold Validation we perform the below steps:

1. split your entire dataset into k-folds. (In our case k=10)

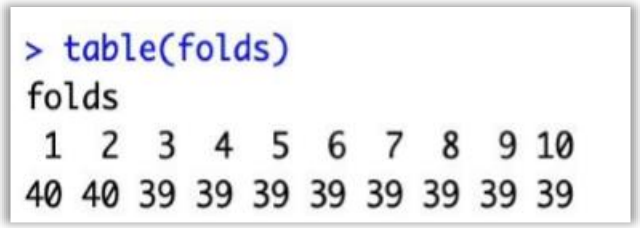
2. For each k-fold in our dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for kth fold

3. Recorded the error we see on each of the predictions

4. Repeat this until each of the k-folds has served as the test set

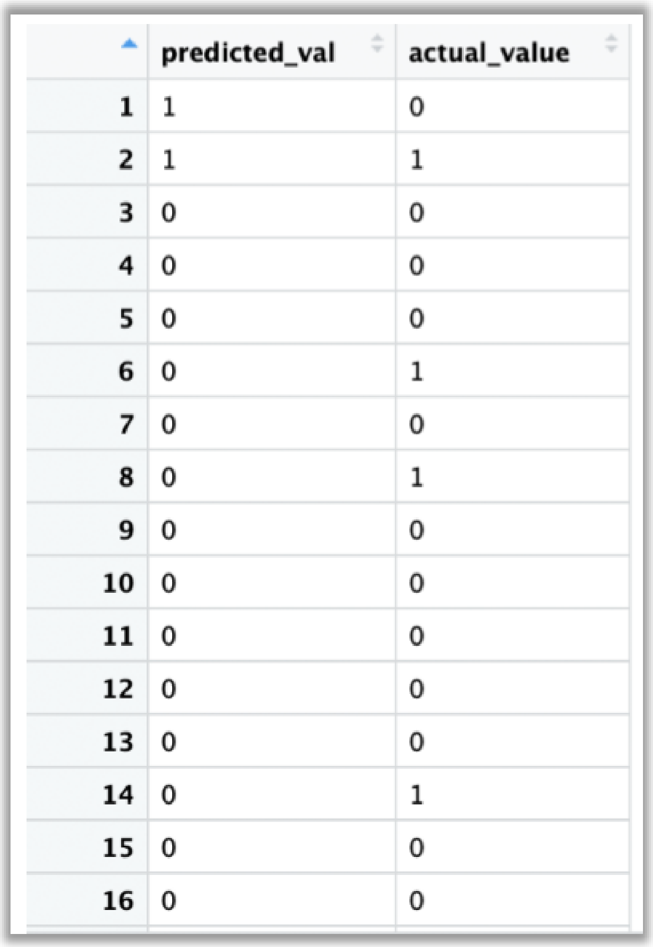
5. The average of our k recorded errors is called the cross-validation error and will serve as our performance metric for the model

Output:



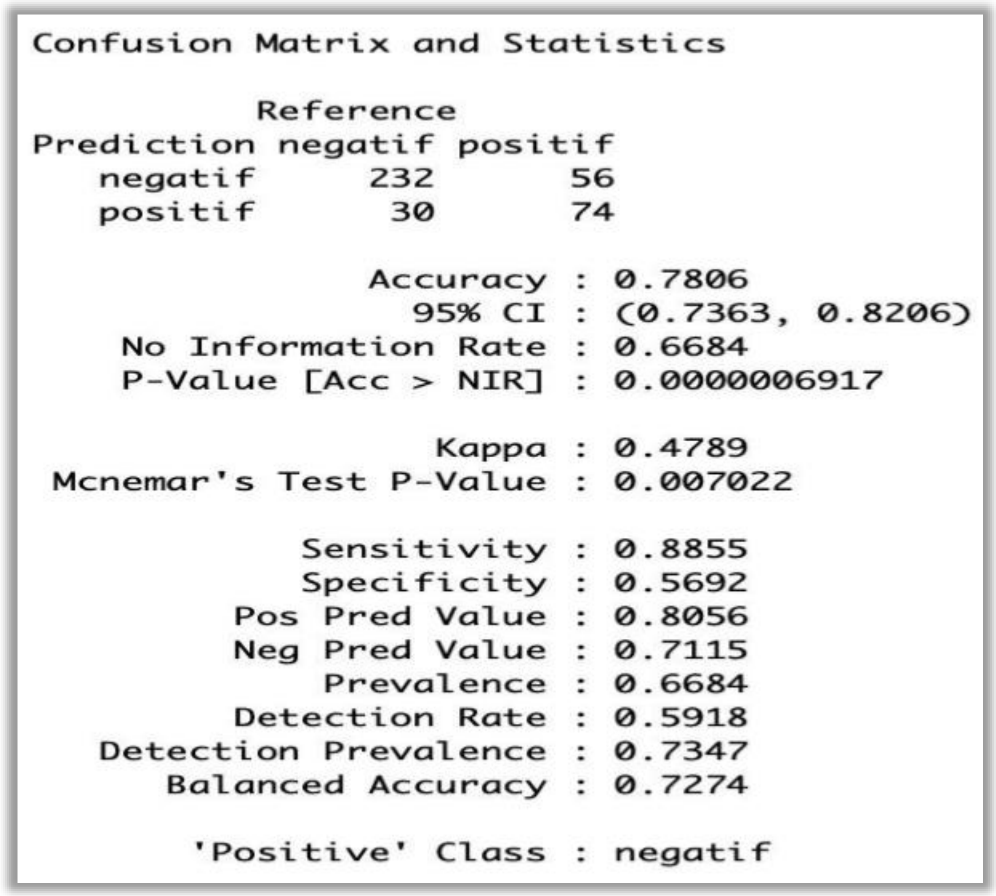
```
> table(folds)
folds
 1  2  3  4  5  6  7  8  9 10
40 40 39 39 39 39 39 39 39 39
```

Output of 10-folds:



	predicted_val	actual_value
1	1	0
2	1	1
3	0	0
4	0	0
5	0	0
6	0	1
7	0	0
8	0	1
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	1
15	0	0
16	0	0

Confusion Matrix:


```
Confusion Matrix and Statistics

          Reference
Prediction negatif positif
   negatif     232      56
   positif      30      74

               Accuracy : 0.7806
                 95% CI : (0.7363, 0.8206)
    No Information Rate : 0.6684
    P-Value [Acc > NIR] : 0.0000006917

                  Kappa : 0.4789
 Mcnemar's Test P-Value : 0.007022

            Sensitivity : 0.8855
            Specificity : 0.5692
         Pos Pred Value : 0.8056
         Neg Pred Value : 0.7115
             Prevalence : 0.6684
         Detection Rate : 0.5918
   Detection Prevalence : 0.7347
      Balanced Accuracy : 0.7274

       'Positive' Class : negatif
```

10. Comparison of Outcomes

No	Model	Accuracy
1	PCA	
2	Naïve Bayes	76.95
3	Logistic Regression	77.17
4	Classification and Regression Tree	73.18
5	10-Fold	78

From the above analysis we can say that the best algorithm to use will be 10-Fold to predict the diabetes.

Sensitivity matters a lot for Diabetes study. We want to minimize false negative as much as possible.

11. Appendix

Below is the code for the analysis

```
library(ggplot2)

library(gains)


### read data

diabetes.df<- read.csv("diabetes.csv")

str(diabetes.df)


###------------Change the datatype to numeric------------####


diabetes.df$pregnant = as.numeric(diabetes.df$pregnant)

diabetes.df$glucose = as.numeric(diabetes.df$glucose)

diabetes.df$diastolic = as.numeric(diabetes.df$diastolic)

diabetes.df$triceps = as.numeric(diabetes.df$triceps)

diabetes.df$insulin = as.numeric(diabetes.df$insulin)

diabetes.df$age = as.numeric(diabetes.df$age)

diabetes.df$test <- as.integer(diabetes.df$test == "positif")

str(diabetes.df)




###---------------------------Heatmap-----------------###


install.packages("gplots") #Install package before running heatmap

library(gplots)


heatmap.2(cor(diabetes.df[, -9]), Rowv = FALSE, Colv = FALSE, dendrogram = "none",
    cellnote = round(cor(diabetes.df[,-9]),2),

    notecol = "black", key = FALSE, trace = "none" , margins = c(10,10))


### Do PCA with non normalized data ###

pca <- prcomp( diabetes.df[,-9])

summary(pca)


### Do PCA with normalized data ###

pcanorm<- prcomp(diabetes.df[,-9], scale. = T) summary(pcanorm)

######## ----------------------Data partitioning------------------- ##########


set.seed(123)

train.index<- sample(c(1:dim(diabetes.df)[1]),dim(diabetes.df)[1]*0.6)

train.df<- diabetes.df[train.index, ] valid.df<- diabetes.df[-train.index, ]



############################NAIVE BAYES#######################

#NAIVE BAYES

#install the required packages

install.packages("Amelia")

install.packages("caret")

install.packages("tinytex")

install.packages("")

install.packages("klaR")


#required libraries

library(Amelia)

library(e1071)

library(caret)
library(tinytex)
library(klaR)

#import the doc
diabetes.df<- read.csv("diabetes.csv")

#setting the outcome variable as categorical
diabetes.df$Outcome<-factor(diabetes.df$Outcome,levels = c(0,1), labels = c("False","True"))
?factor

#summary of the dataset
str(diabetes.df)

#head of the data
head(diabetes.df)

#visualize the missing data
missmap(diabetes.df)

#summary of the dataset
summary(diabetes.df)

#data partitioning
#build model
set.seed(123)
train.index<-sample(c(1:dim(diabetes.df)[1]),dim(diabetes.df)[1]*0.6)
train.df<-diabetes.df[train.index,]
valid.df<-diabetes.df[-train.index,]
```

```
#check dimension of the split
prop.table(table(train.df$Outcome))
prop.table(table(valid.df$Outcome))


#create objects x which holds the predictor variables and y which holds the response variables
x = train.df[,-9]
y = train.df$Outcome

#Naive Bayes
model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))

#Model Evaluation
#Predict testing set
Predict <- predict(model,newdata = valid.df )

#Get the confusion matrix to see accuracy value and other parameter values
confusionMatrix(Predict, valid.df$Outcome )


#checking the importance of the variables
imp<- varImp(model)
plot(imp)



################################lOGISTIC REGRESSION################################

# logistic regression
```

```
library(rpart)

library(rpart.plot)

library(caret)

library(pROC)


#import the data file

diabetes<-read.csv("diabetes.csv")


# partition with random sample

set.seed(1)

dim(diabetes)

train.index<-sample(c(1:dim(diabetes)[1]),dim(diabetes)[1]*0.6)

train.df<-diabetes[train.index,]

valid.df<-diabetes[-train.index,]


# run logistic regression for training data

logit.reg<-glm(Outcome~.,data=train.df,family="binomial")

logit.reg

options(scipen=99)

summary(logit.reg)


# generate confusion matrix for training data

logit.reg.pred.train<-predict(logit.reg,train.df,type = "response")

ifelse(logit.reg.pred.train>0.5,1,0)

confusionMatrix(as.factor(ifelse(logit.reg.pred.train>0.5,1,0)),as.factor(train.df$Outcome))


# run logistic regression for validation data

logit.reg.valid<-glm(Outcome~.,data=valid.df,family="binomial")

logit.reg.valid
```

```
options(scipen=99)

summary(logit.reg.valid)


# generate confusion matrix for validation data

logit.reg.pred.valid<-predict(logit.reg.valid,valid.df,type = "response")

ifelse(logit.reg.pred.valid>0.5,1,0)

confusionMatrix(as.factor(ifelse(logit.reg.pred.valid>0.5,1,0)),as.factor(valid.df$Outcome))


#Lift chart for training data

r.train<-roc(train.df$Outcome,logit.reg.pred.train)

plot.roc(r.train)


#Lift chart for validation data

r.valid<-roc(valid.df$Outcome,logit.reg.pred.valid)

plot.roc(r.valid)



#####################################K-MEANS
CLUSTERING#########################################

#K-MEAN CLUSTERING


#Importing Data

diabetes<-read.csv("diabetes.csv")


#Normalising data

diabetes.norm<-sapply(diabetes,scale)


#determining kvalue
```

```
#elbow method

km_wss <- sapply(1:15, function(k){kmeans(diabetes.norm, k)$tot.withinss})
km_wss
plot(1:15, km_wss, main = "Elbow method",xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")



#Setting k=6
set.seed(123)
km1<-kmeans(diabetes.norm,6)

#Cluster size
km1$size

#Cluster membership
km1$cluster

#Cluster centers
km1$centers

#With-in cluster sum of squares
km1$withinss

#distance between centers
dist(km1$centers)

#Profile plot of centroids
plot(c(0), xaxt = 'n', ylab = "", type = "l",
     ylim = c(min(km1$centers), max(km1$centers)), xlim = c(0, 8))
axis(1, at = c(1:9), labels = names(diabetes))
for (i in c(1:6))
  lines(km1$centers[i,], lty = i, lwd = 2, col = ifelse(i %in% c(1, 3, 5),
                                                       "black", "dark grey"))
text(x = 0.2, y = km1$centers[, 1], labels = paste("Cluster", c(1:6)))



################################HIERARCHIAL
CLUSTERING########################################


#generating dendogram based on single linkage


d.norm<-dist(diabetes.norm,method='euclidean')
hc1<-hclust(d.norm,method = 'single')
plot(hc1,main="Clustering Dendrogram",ylab="distance")

#setting cluster k=6
memb1<- cutree(hc1,k=6)
memb1

#generating dendogram based on complete linkage
hc2<-hclust(d.norm,method = 'complete')
plot(hc2,main="Clustering Dendrogram",ylab="distance")

#setting cluster k=6
memb2<- cutree(hc2,k=6)
memb2
```

```
###########----------------- CLASSIFICATION TREES----------------------------###############

library(rpart)
install.packages("rpart.plot")
library(rpart.plot)
library(caret)

#train.df$Outcome<-as.factor(train.df$test)
default.ct <- rpart(Outcome ~ ., data = train.df, method = "class")
prp(default.ct, type = 1, extra = 1, under = TRUE,, main="Decision Tree",
   split.font = 2, varlen = -10, box.palette = blues9)
default.ct.point.pred <- predict(default.ct,newdata = valid.df,type = "class")
table(default.ct.point.pred, valid.df$Outcome)
confusionMatrix(default.ct.point.pred, as.factor(valid.df$Outcome))
```

R script File:

R_Project.R

12. References

1.Galit Shmueli, Peter Bruce, Inbal Yahav, Nitin Patel, and Kenneth Lichtendahl. Data Mining for Business Analytics: Concepts, Techniques, and Applications in R, 1st edition, 2018. Wiley. ISBN: 978-1-118-87936-8 (Hardcover), ISBN: 978-1- 118-87933-7 (E-Book).

2. https://www.andreagrandi.it

3. http://www.statsoft.com/textbook/classification-and-regression-trees

4. https://www.kaggle.com/

5. https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-crossvalidation-in-python-r/