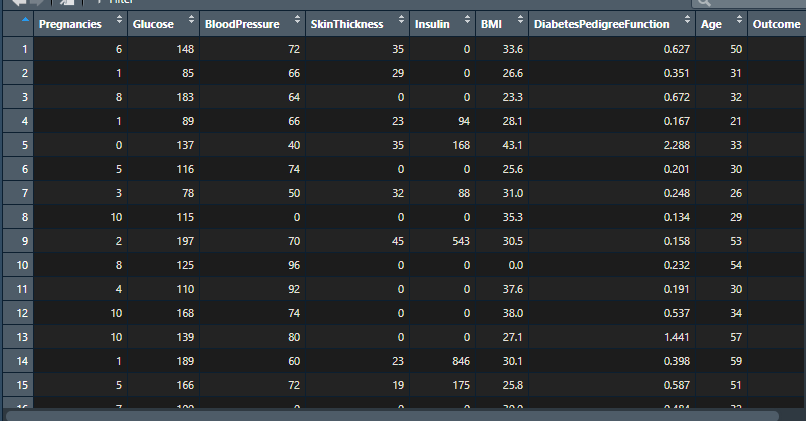
**Mini Project 2- Diabetes Dataset- Logistic Regression**

>> diabetes= read.csv(file.choose())

We uploaded our dataset present in a csv file with the above line of code

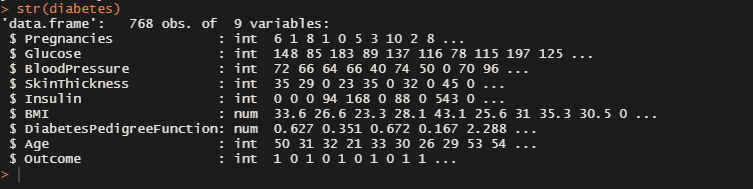
**View()** function visualizes the dataset in a table format, helping the user to have glimpse of the total number of entries that is there in the dataset along with all the columns/variables of the dataset.



>>str(diabetes) >>dim(diabetes) >>names(diabetes)

We also used few basic operational functions to get some knowledge/overview of the dataset that we have to work on.

**str(diabetes)** function gave the structure of the dataset telling that it is a data frame with 768 rows and 9 columns. It also gave information about the class of each variable of the dataset i.e. whether it is a factor or a numeric data.

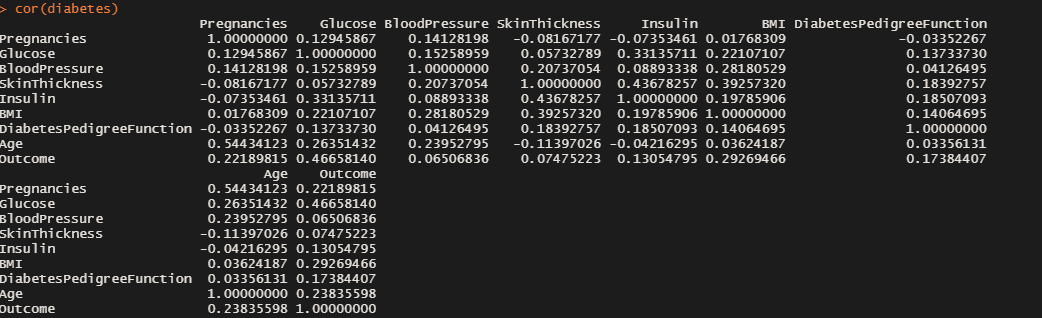


**dim(diabetes)** function again gave us the information about the number of rows and columns🡪 **[768 x 9]**

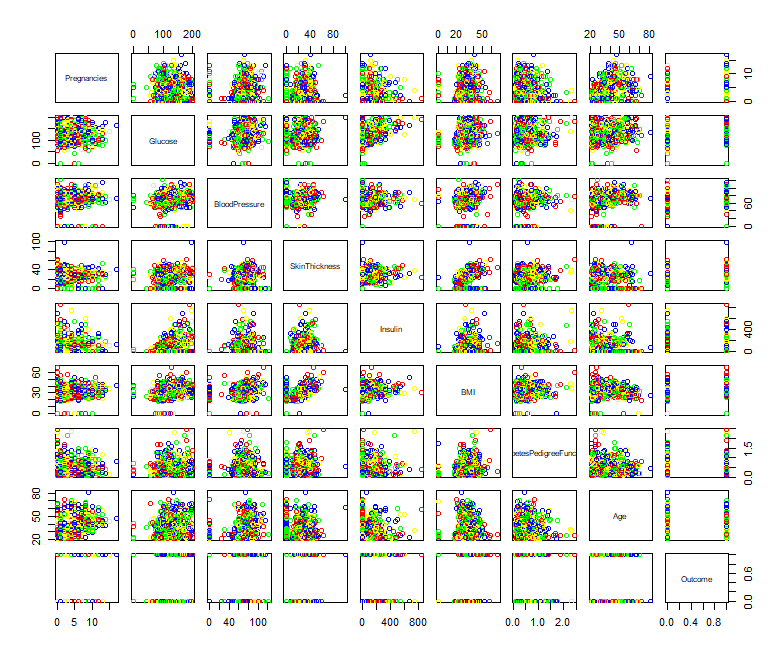
**names(diabetes)** function displayed the names of all the variables present in the dataset.

>> summary (diabetes) >> cor(diabetes)

**summary (diabetes)** Summary of a dataset provided all the statistical information about each variable of the dataset like its mean, median, minimum value, maximum value, 1st quartile and 3rd quartile.



**pairs(diabetes)** with the help of this function a matrix of scatterplots is produced.

****

>> attach(diabetes)

with the help of attach() function the College dataset was attached to the R search path. This means that the database is searched by R when evaluating a variable, so objects in the database can be accessed by simply giving their names.

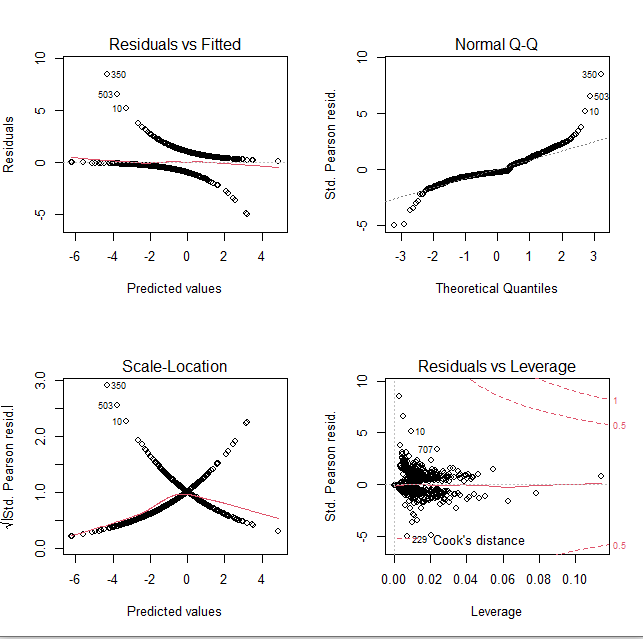
**Approach-1:**

**MODEL CREATION BY GLM FUNCTION-**

>> Logistic.Regression <- glm(Outcome ~ .,data = diabetes, family = "binomial")

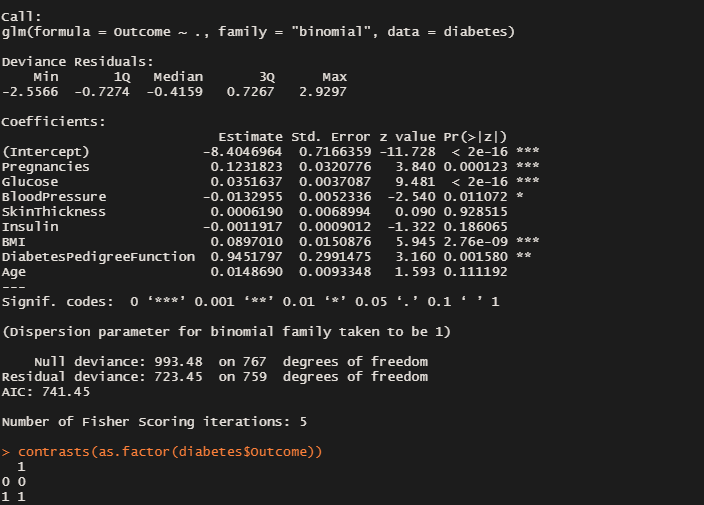
Performed logistic regression on the dtaset with the help of glm(generalized linear model) function

>> windows();plot(Logistic.Regression)



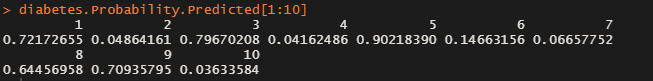
>> summary(Logistic.Regression)

>> contrasts(as.factor(Outcome))

>> diabetes.Probability.Predicted <- predict(Logistic.Regression, type = "response");diabetes.Probability.Predicted

>> diabetes.Probability.Predicted[1:10]

With the help of above lines of codes we predicted the values of variable-Outcome for our model

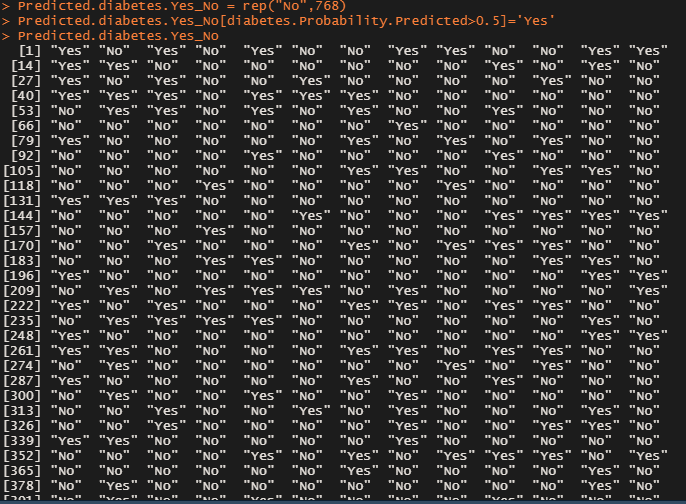


>> Predicted.diabetes.Yes\_No = rep("No",768)

>> Predicted.diabetes.Yes\_No[diabetes.Probability.Predicted>0.5]='Yes'

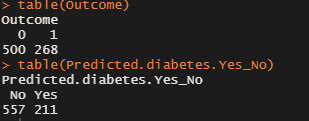
>> Predicted.diabetes.Yes\_No

With the help of above lines of codes we calculated the probability of a person being diabetic.



>> table(Outcome)

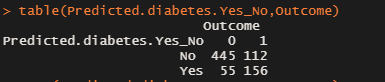
>> table(Predicted.diabetes.Yes\_No)



>> table(Predicted.diabetes.Yes\_No,Outcome)

>> mean(Predicted.diabetes.Yes\_No==Outcome)

With the help of above lines of code a confusion matrix was created by which we were able to calculate the accuracy of our model which came out to be 78.28 %. Therefore the training error rate is 21.72%

****

**Approach-2**

**SPLITTING OF DATASET ON THE BASIS OF AGE > 40-**

In this approach we split our dataset into train and test, The train dataset included person of age below 41 and test contained people above 40 years of age.

>> diabetes.train = (Age < 41)

>> diabetes.train

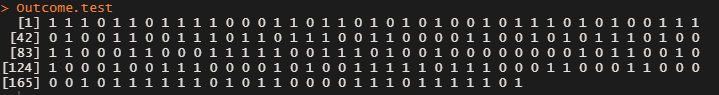
>> diabetes.test = diabetes[!diabetes.train,]

>> dim(diabetes.test)



>> Outcome.test = Outcome[!diabetes.train]

>> Outcome.test



glm.fit <- glm(Outcome~., data = diabetes,family = "binomial",subset = diabetes.train)

performed logistic regression on the train dataset and created a classification algorithm based model

>>glm.pred = predict(glm.fit,diabetes.test,type = "response")

>>glm.prob <- rep("NO",194)

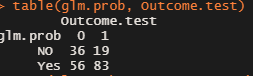
>>glm.prob[glm.pred > 0.5]="Yes"

With the help of above lines of codes we calculated the probability of a person being diabetic.

>>table(glm.pred, Outcome.test)

>>mean(glm.pred == Outcome.test)

With the help of above lines of code a confusion matrix was created by which we were able to calculate the accuracy of our model which came out to be 61.34 %. Therefore the testing error rate is 38.65%



**Approach-3**

**LINEAR DISCRIMINANT ANALYSIS-**

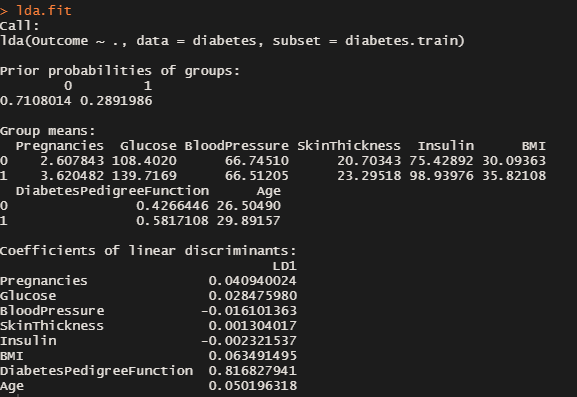
**Discriminant function analysis** is used to determine which variables discriminate between two or more naturally occurring groups.

>>library(MASS)

function ‘lda’ was present in MASS library, therefore we need to install it first. LDA is dimensionality reduction technique used to project the features in higher dimension space into a lower dimension space.

>>lda.fit = lda(Outcome~ . , data = diabetes , subset = diabetes.train)

created a classification model with the help of lda function



>>lda.pred = predict(lda.fit ,diabetes.test)

>>names(lda.pred)

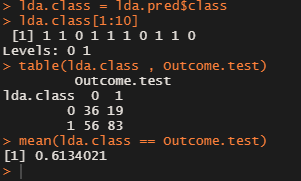
With the help of above lines of codes we predicted the values of variable-Outcome for our model

>>lda.class = lda.pred$class

>>lda.class[1:10]

>>table(lda.class , Outcome.test)

>>mean(lda.class == Outcome.test)



With the help of above lines of code a confusion matrix was created by which we were able to calculate the accuracy of our model which came out to be 61.34 %. Therefore the error rate is 38.65%

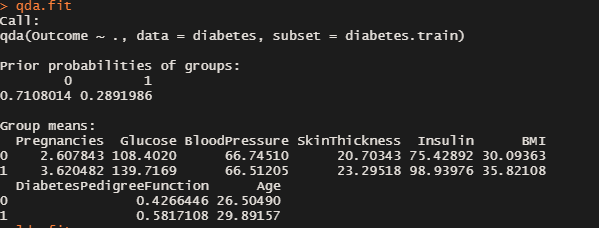
**Approach-4**

**Quadratic DISCRIMINANT ANALYSIS-**

LDA (Linear Discriminant Analysis) is used when a linear boundary is required between classifiers and QDA (Quadratic Discriminant Analysis) is used to find a non-linear boundary between classifiers.

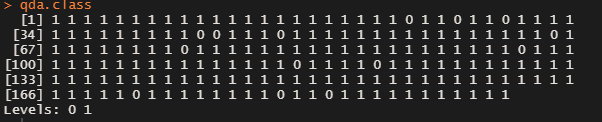
>>qda.fit = qda(Outcome~ . , data = diabetes , subset = diabetes.train)

>>qda.fit



>>qda.class = predict(qda.fit , diabetes.test)$class

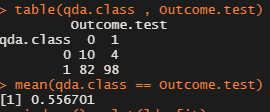
>>qda.class



>>table(qda.class , Outcome.test)

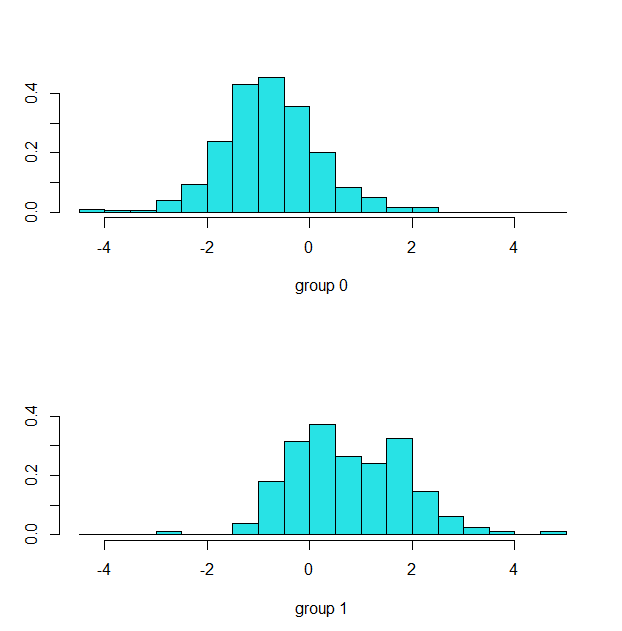
>>mean(qda.class == Outcome.test)

With the help of above lines of code a confusion matrix was created by which we were able to calculate the accuracy of our model which came out to be 55.67 %. Therefore the error rate is 44.33%



>>windows(); plot(lda.fit)

>>par(mfrow = c(1,1))



**Approach-5**

**K- NEAREST NEIGHBOUR-**

>>library(class)

knn function is present in class library, therefore we first installed it. The K in knn denotes the number of nearest neighbours to a new unknown variable that has to be predicted or classified.

>>names(diabetes)

>>diabetes.train = (Age<41)

>>dim(diabetes.train)

>>str(diabetes.train)

>>diabetes.test = diabetes[!diabetes.train,]

>>dim(diabetes.test)

>>Outcome.test = Outcome[!diabetes.train]

>>Outcome.train= Outcome[diabetes.train]

With the above lines of code we split he dataset into train and test

>>train.X = cbind(Insulin,Glucose, BloodPressure, Age, Pregnancies)[diabetes.train,]

>>train.X

>>test.X = cbind(Insulin,Glucose, BloodPressure, Age, Pregnancies)[!diabetes.train,]

>>test.X

afterwards we created a separated dataframe of both test and train dataset with certain variables from the parent dataset

>>set.seed(123)

>>knn.pred = knn(train.X, test.X, Outcome.train, k=4)

on this newly formed dataset we created our model using knn function

summary(knn.pred)

table(knn.pred, Outcome.test)

mean(knn.pred == Outcome.test)

With the help of above lines of code a confusion matrix was created by which we were able to calculate the accuracy of our model which came out to be 63.4 %. Therefore the error rate is 36.6%

