KNN OF DIABATES DATA SET

Here, we try to create a KNN Model on the Diabates of pregnent mother's Data set in R (using Google Colaab)

First we run the under mentioned programme to run R programms in Google Colaab

```
%load_ext rpy2.ipython
```

```
The rpy2.ipython extension is already loaded. To reload it, use:
  %reload_ext rpy2.ipython
```

1.INSTALL THE REQUIRED PACKAGES FOR THIS PROJECT

Here we need the following packages:

1. gt
2. class
3. caret
4. GGally

```
%%R
install.packages(c("gt","class","caret","GGally"))
```

```
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]:

R[write to console]: downloaded 164 KB



R[write to console]: trying URL 'https://cran.rstudio.com/src/contrib/sass_0.4.0.tar.

R[write to console]: Content type 'application/x-gzip'
R[write to console]:  length 3022459 bytes (2.9 MB)

R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
```

```
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
R[write to console]: =
```

2.IMPORTING LIBRARIES AND DATA SET

Then we Import the given data set dataset.csv into this project.

```
%%R
library(gt)
library(class)
library(caret)
library(GGally)
diabates_ds<-read.csv('abcd.csv',stringsAsFactors = FALSE)
```

3.OVERLOOK OF OUR DATA SET

Here we can observe our data set, which have 768 observations with 9 variables, with target variable "Outcome". Where "0" outcome means "non-diabatic", and "1" outcome means "diabatic".

```
%%R
head(diabates_ds)
```

```
  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
1           6     148            72      35.00000       0 33.6
2           1      85            66      29.00000       0 26.6
3           8     183            64      20.53646       0 23.3
4           1      89            66      23.00000      94 28.1
5           0     137            40      35.00000     168 43.1
6           5     116            74      20.53646       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
```

```
%%R
str(diabates_ds)
```

```
'data.frame':   768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : num  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : num  72 66 64 66 40 ...
```

```
 $ SkinThickness           : num  35 29 20.5 23 35 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 ...
 $ 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 "0","1": 2 1 2 1 2 1 2 1 2 2 ...
```

4.CHECKING IS THERE ANY MISSING VALUE?

```
%%R
sum(is.na(diabates_ds))
```

```
[1] 0
```

5.PRE-PROCESSING THE DATA TO INCREASE THE QUALITY OF THE MODEL

Though there are no missing value in the data set, but in the columns of Glucose, Blood Pressure , Skin thickness,BMI have zero values present. But these values can not be zero. so we replace these values, with the mean of their respective columns.

```
%%R
diabates_ds$Glucose[diabates_ds$Glucose==0]<-mean(diabates_ds$Glucose)
diabates_ds$BloodPressure[diabates_ds$BloodPressure==0]<-mean(diabates_ds$BloodPressure)
diabates_ds$SkinThickness[diabates_ds$SkinThickness==0]<-mean(diabates_ds$SkinThickness)
diabates_ds$BMI[diabates_ds$BMI==0]<-mean(diabates_ds$BMI)
head(diabates_ds)
```

```
  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
1           6     148            72      35.00000       0 33.6
2           1      85            66      29.00000       0 26.6
3           8     183            64      20.53646       0 23.3
4           1      89            66      23.00000      94 28.1
5           0     137            40      35.00000     168 43.1
6           5     116            74      20.53646       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
```

6.CHANGING THE LAST COLUMN

As the attribute of interest based on which we need to classify the data is “Diabatic or not” where (Diabatic,Non-Diabatic):(1,0), we change the type of the outcome Data as Factors:

```
%%R
diabates_ds$Outcome<-as.factor(diabates_ds$Outcome)
str(diabates_ds)
```

```
'data.frame':   768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : num  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : num  72 66 64 66 40 ...
 $ SkinThickness           : num  35 29 20.5 23 35 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 ...
 $ 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 "0","1": 2 1 2 1 2 1 2 1 2 2 ...
```

7.ANALYZING THE DATA SET

Now we Start the Analysis Stage.

Well first we normalize the data set using Z-scores so that there are no biases in the data due to difference in location and scale:

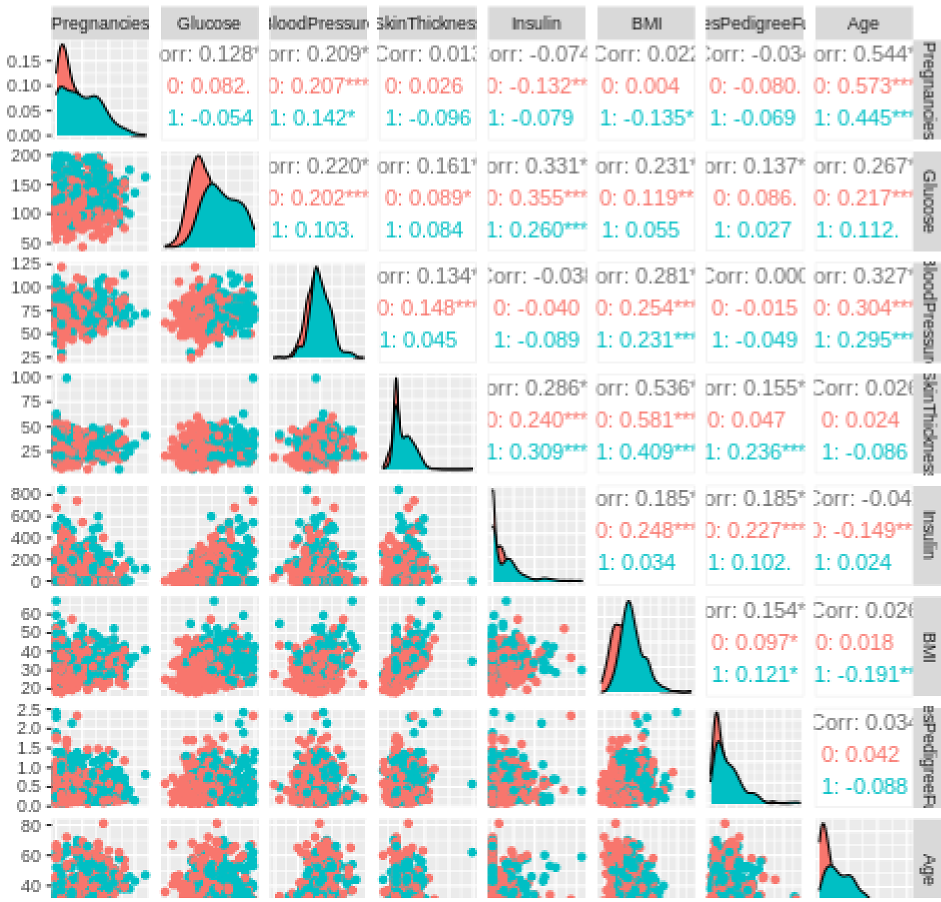
```
%%R
normlz <-function(x) {return(x-as.numeric(mean(x)))/as.numeric(sd(x)) }
diabates_norm<-as.data.frame(lapply(diabates_ds[,1:8], normlz))
summary(diabates_norm)
```

```
  Pregnancies         Glucose          BloodPressure       SkinThickness
 Min.   :-3.8451   Min.   :-77.682   Min.   :-48.2548   Min.   :-19.606
 1st Qu.:-2.8451   1st Qu.:-21.932   1st Qu.: -8.2548   1st Qu.: -6.070
 Median :-0.8451   Median : -4.682   Median : -0.2548   Median : -3.606
 Mean   : 0.0000   Mean   :  0.000   Mean   :  0.0000   Mean   :  0.000
 3rd Qu.: 2.1549   3rd Qu.: 18.568   3rd Qu.:  7.7452   3rd Qu.:  5.394
 Max.   :13.1549   Max.   : 77.318   Max.   : 49.7452   Max.   : 72.394
    Insulin            BMI           DiabetesPedigreeFunction      Age
 Min.   :-79.80   Min.   :-14.2508   Min.   :-0.39388         Min.   :-12.241
 1st Qu.:-79.80   1st Qu.: -4.9508   1st Qu.:-0.22813         1st Qu.: -9.241
 Median :-49.30   Median : -0.4508   Median :-0.09938         Median : -4.241
 Mean   :  0.00   Mean   :  0.0000   Mean   : 0.00000         Mean   :  0.000
 3rd Qu.: 47.45   3rd Qu.:  4.1492   3rd Qu.: 0.15437         3rd Qu.:  7.759
 Max.   :766.20   Max.   : 34.6492   Max.   : 1.94812         Max.   : 47.759
```

8.DIAGRAMS

Here these plots can easily describe the associationship (using correlation) between the variables.

```
%%R
ggpairs(diabates_ds,columns=1:8,mapping =aes(color=Outcome))
```



9.SPLITTING THE DATA SET INTO TRAINING AND TEST DATA SET

Here we split the total observations into 70%-30% as Training and Testing data set.

Total observations is 768, so the 1st 537 (70% of 768) observations are selected for training purpose, and the rest are for testing purpose.

```
%%R
diabates_train<-diabates_norm[1:537,1:8]
diabates_test<-diabates_norm[538:768,1:8]
diabates_train_labels<-as.array(diabates_ds[1:537,9])
diabates_test_labels<-as.array(diabates_ds[538:768,9])
```

10.Finding the Optimal Number Of Neighbours

Here we create a loop to find the Percentage of Accuracy for each k from 1 to 30:

```
%%R
set.seed(456)
i=1
k.optm=1
for (i in 1:30)
{
knn.pred <- knn(train=diabates_train, test=diabates_test, cl=diabates_train_labels, k=i)
k.optm[i] <- 100 * sum(diabates_test_labels == knn.pred)/NROW(diabates_test_labels)
```

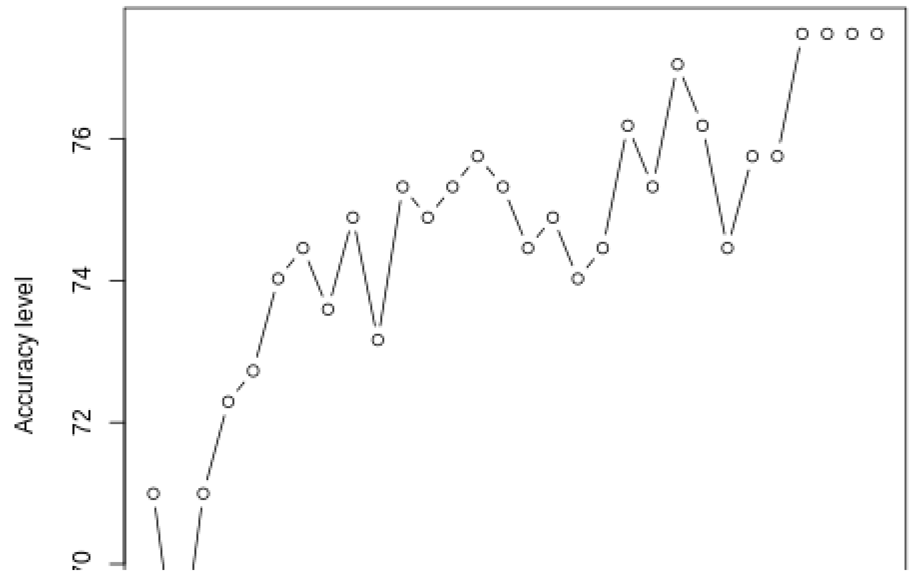
```
k=i
cat(k,'=',k.optm[i],'
')
}
```

```
1 = 70.99567
2 = 68.39827
3 = 70.99567
4 = 72.29437
5 = 72.72727
6 = 74.02597
7 = 74.45887
8 = 73.59307
9 = 74.89177
10 = 73.16017
11 = 75.32468
12 = 74.89177
13 = 75.32468
14 = 75.75758
15 = 75.32468
16 = 74.45887
17 = 74.89177
18 = 74.02597
19 = 74.45887
20 = 76.19048
21 = 75.32468
22 = 77.05628
23 = 76.19048
24 = 74.45887
25 = 75.75758
26 = 75.75758
27 = 77.48918
28 = 77.48918
29 = 77.48918
30 = 77.48918
```

Accuracy vs. k graph

Then we draw the accuracy plot and determine the value of k for which we have the highest Accuracy:

```
%%R
acc_plot<-plot(k.optm,·type="b",·xlab="K-·Value",ylab="Accuracy·level")
```




```
Thus, from the graph we see that the model has
K=27 neighbours.

                               **THANK YOU**
```

Thus, from the graph we see that the model has the best accuracy of 77.49% for K=27 neighbours.

THANK YOU

0s completed at 10:41 PM