American International University-Bangladesh (AIUB)

Course: INTRODUCTION TO DATA SCIENCE [D]

Faculty Name: TOHEDUL ISLAM

Student Name: Shanto Kumar Basak

ID: 20-42945-1

Submitted Date: 30th April 2023

Using KNN from Diabetics Dataset

INTRODUCTION:

On the basis of supervised datasets, KNN (K-Nearest Neighbor) is one of the most well-liked machine learning algorithms. By choosing the Kth neighbor, this procedure is then used. Calculate the Euclidean distance after picking the neighbors. The number of data points in each category among these k neighbors were counted. The category for which the number of neighbors is highest should receive the new data points. Next, the model is prepared. With the help of the KNN algorithm and the R programming language, the project's goal was to create a model for predicting diabetic patients from data on Kaggle.

METHODOLOGY:

The main data was found from Kaggle (link- https://www.kaggle.com/datasets/mathchi/diabetes-data-set). Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration 2 hours in an oral glucose tolerance test
- Blood Pressure: Diastolic blood pressure (mm Hg)
- Skin Thickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/ (height in m) ^2)
- Diabetes Pedigree Function: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

1) Print the Data Set

f_dset <- read.csv("F:/f_dataset.csv",header=TRUE,sep=",")
f_dset

```
> f_dset <- read.csv("F:/f_dataset.csv",header=TRUE,sep=",")
> f_dset
   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI DiabetesPedigreeFunction Age
1            6     148            72            35       0 33.6                    0.627  50
2            1      85            66            29       0 26.6                    0.351  31
3            8     183            64             0       0 23.3                    0.672  32
4            1      89            66            23      94 28.1                    0.167  21
5            0     137            40            35     168 43.1                    2.288  33
6            5     116            74             0       0 25.6                    0.201  30
7            3      78            50            32      88 31.0                    0.248  26
8           10     115             0             0       0 35.3                    0.134  29
9            2     197            70            45     543 30.5                    0.158  53
10           8     125            96             0       0  0.0                    0.232  54
11           4     110            92             0       0 37.6                    0.191  30
12          10     168            74             0       0 38.0                    0.537  34
13          10     139            80             0       0 27.1                    1.441  57
14           1     189            60            23     846 30.1                    0.398  59
15           5     166            72            19     175 25.8                    0.587  51
16           7     100             0             0       0 30.0                    0.484  32
17           0     118            84            47     230 45.8                    0.551  31
18           7     107            74             0       0 29.6                    0.254  31
19           1     103            30            38      83 43.3                    0.183  33
```

2) An overview of each attribute in the data set

summary(f_dset)

```
> summary(f_dset)
  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
```

str(f_dset)

```
> str(f_dset)
'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                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

3) Change to mean value in which case the attributes value is 0

f_dset$Pregnancies[f_dset$Pregnancies ==0] = mean(f_dset$Pregnancies,)
f_dset$Glucose [f_dset$Glucose ==0] = mean(f_dset$Glucose,)
f_dset$BloodPressure[f_dset$BloodPressure ==0] = mean(f_dset$BloodPressure,)
f_dset$SkinThickness[f_dset$SkinThickness ==0] = mean(f_dset$SkinThickness,)
f_dset$Insulin[f_dset$Insulin ==0] = mean(f_dset$Insulin,)
f_dset$BMI [f_dset$BMI ==0] = mean(f_dset$BMI,)
f_dset$Age[f_dset$Pregnancies ==0] = mean(f_dset$Age,)

```
 $ Outcome                  : int  1 0 1 0 1 0 1 0 1 1 ...
> f_dset$Pregnancies[f_dset$Pregnancies ==0] = mean(f_dset$Pregnancies,)
> f_dset$Glucose [f_dset$Glucose ==0] = mean(f_dset$Glucose,)
> f_dset$BloodPressure[f_dset$BloodPressure ==0] = mean(f_dset$BloodPressure,)
> f_dset$SkinThickness[f_dset$SkinThickness ==0] = mean(f_dset$SkinThickness,)
> f_dset$Insulin[f_dset$Insulin ==0] = mean(f_dset$Insulin,)
> f_dset$BMI [f_dset$BMI ==0] = mean(f_dset$BMI,)
> f_dset$Age[f_dset$Pregnancies ==0] = mean(f_dset$Age,)
> |
```

Load the library of class

```
library(class)
```

4) Data Normalization

Normalization is a very important part of KNN. It is hard to make the current accuracy of is the data is not in well-shaped.It shapes the data in 0 to 1. The main math of the normalization is

= (value-min(value))/(max(value)-min(value))

```
> noramalize_data <- function(x)
+ {
+   nu= x-min(x)
+   dn= max(x)-min(x)
+   return(nu/dn)
```

5) Data Splitting

Data splitting basically involves splitting the data set into training and testing data set.There ware taken 70% data in train dataset and 30% data on the test dataset for making the confusion matrix so randomly.

```
> sample_data <- sample(2,nrow(make_data),replace = TRUE ,prob = c(0.70,0.30))
>
> train_data<- make_data[sample_data==1, 1:8]
> test_data <- make_data[sample_data==2, 1:8]
>
> train_datalabels <- f_dset[sample_data==1,9]
> test_datalabels <- f_dset[sample_data==2,9]
>
```

6) Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model or "classifier" on a set of test data for which the true values are known.

```
prediction= knn(train = train_data,test = test_data,train_datalabels,k=5)
con_matrix= table(test_datalabels,prediction)

con_matrix
```

True Negative =128, True Positive=45, False positive =26, False Negative= 33

```
> con_matrix
                prediction
test_datalabels   0   1
              0 128  26
              1  33  45
>
```

7) Here is the Some Accuracy

```
Accuracy= function(con_matrix)

{

 sum=0

 for(i in 1:nrow(con_matrix))

  sum=sum+con_matrix[i,i]

 return(sum/sum(con_matrix))

}

print(paste('ACCURACY OF THIS MODEL IS = ',Accuracy(con_matrix)*100,'%'))
```

```
> Accuracy= function(con_matrix)
> {
>   sum=0
>   for(i in 1:nrow(con_matrix))
>     sum=sum+con_matrix[i,i]
>   return(sum/sum(con_matrix))
> }
>
> print(paste('ACCURACY OF THIS MODEL IS = ',Accuracy(con_matrix)*100,'%'))
[1] "ACCURACY OF THIS MODEL IS =  74.5689655172414 %"
>
```

8) Calculate accuracy with various K values between 1 and 100

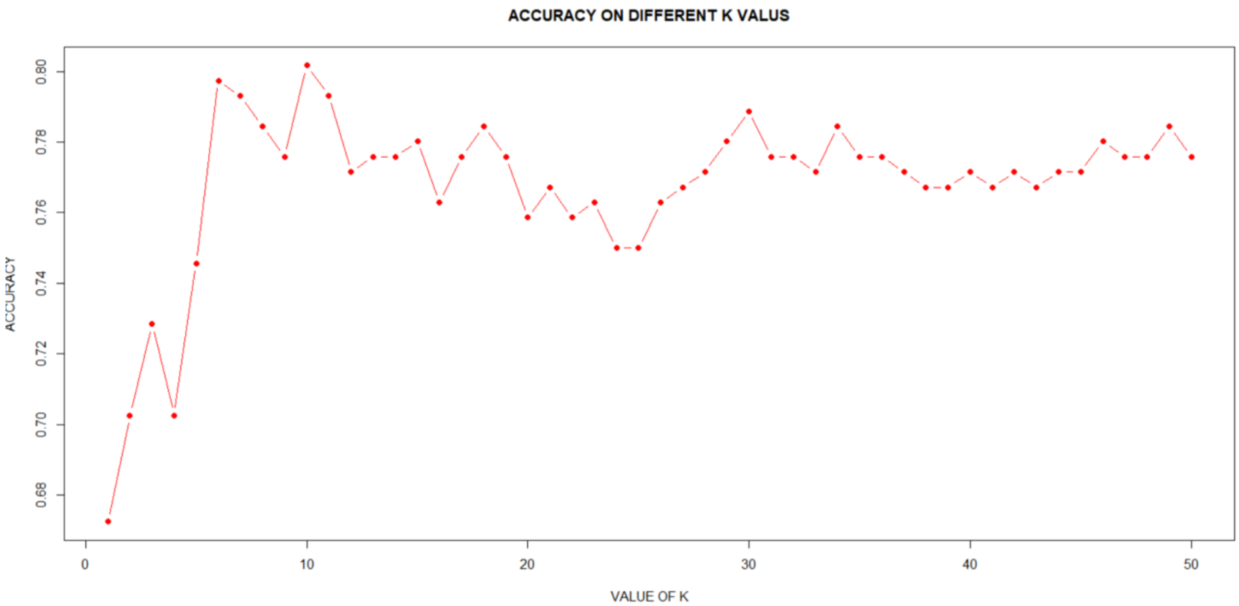
```
list_k <- c(1:100)

arr_k_result <-c()


for(i in 1: length(list_k))

{

  prediction= knn(train = train_data,test = test_data,train_datalabels,k=i)

        con_matrix= table(test_datalabels,prediction)

        arr_k_result[i]<-Accuracy(con_matrix)
        }

        knn_rslt <- cbind(list_k ,arr_k_result)

        colnames(knn_rslt)<-c("Value of k","      Accuracy")
        knn_rslt <- as.data.frame(knn_rslt)
```

```
> knn_rslt
      list_k arr_k_result
 [1,]      1    0.6637168
 [2,]      2    0.6814159
 [3,]      3    0.7300885
 [4,]      4    0.7212389
 [5,]      5    0.7123894
 [6,]      6    0.7389381
 [7,]      7    0.7256637
 [8,]      8    0.7168142
 [9,]      9    0.7212389
[10,]     10    0.7300885
[11,]     11    0.7389381
[12,]     12    0.7433628
[13,]     13    0.7477876
[14,]     14    0.7522124
[15,]     15    0.7477876
[16,]     16    0.7389381
[17,]     17    0.7300885
[18,]     18    0.7212389
[19,]     19    0.7300885
[20,]     20    0.7389381
[21,]     21    0.7256637
[22,]     22    0.7212389
[23,]     23    0.7433628
[24,]     24    0.7389381
[25,]     25    0.7345133
[26,]     26    0.7212389
```

9) All-accuracy scatterplot with the K value

knn_rslt

plot(knn_rslt$`Value of k`,knn_rslt$`      Accuracy`,type="b",pch=16, col="red", lwd=1, xlab="VALUE OF K", ylab="ACCURACY", main="ACCURACY ON DIFFERENT K VALUS")



DISCUSSION:

A dataset contained a total of 730 data. And there predicted the data by 70% train and 30% test dataset selected randomly. The forecast would be more accurate and useful in this situation if there were more data and qualities provided.