American International University - Bangladesh (AIUB)

INTRODUCTION TO DATA SCIENCE [D]

Faculty Name: TOHEDUL ISLAM

Student Name: MD. NADIM HASAN

ID: 20-43004-1

Prediction Diabetics Using KNN from Diabetics Dataset on Kaggle

INTRODUCTION:

KNN **(K-Nearest Neighbor)** is one of the most popular Machine Learning Algorithm based on the supervised dataset. This technique is followed by selecting the number of K of the neighbors. After selecting the neighbors calculate the Euclidean distance. Among these k neighbors, counted the number of the data points in each category. Assign the new data points to that category for which the number of the neighbor is maximum. Then the model is ready. In the project there were trying to build a model for predicting the diabetics patient from a dataset on Kaggle using KNN-algorithm and R language.

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)

SOURCES:

(a) Original owners: National Institute of Diabetes and Digestive and kidney diseases
(b) Donor of database: Vincent Sigillito (vgs@aplcen.apl.jhu.edu)
Research Center, RMI Group Leader
Applied Physics Laboratory
The Johns Hopkins University
Johns Hopkins Road
Laurel, MD 20707
(301) 953-6231
(c) Date received: 9 May 1990

CODE AND THE STEPS OF THE PROJECTS:

- **IMPORT DATA:**

```r
f_dataset <- read.csv("C:/Users/user/Desktop/PROJECT_NADIM/diabetes.csv",header=TRUE,sep=",")
f_dataset
```

- **OUTPUT:**

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

- **OBSERVE THE DATA SET:**

```r
summary(f_dataset)
str(f_dataset)
```

- **OUTPUT:**

```
> summary(f_dataset)
  Pregnancies        Glucose       BloodPressure     SkinThickness      Insulin           BMI            Pedigree            Age            Outcome
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00   Min.   :0.000
 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   1st Qu.:24.00   1st Qu.:0.000
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5   Median :32.00   Median :0.3725   Median :29.00   Median :0.000
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24   Mean   :0.349
 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   3rd Qu.:41.00   3rd Qu.:1.000
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00   Max.   :1.000
```

```
> str(f_dataset)
'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 ...
 $ Pedigree     : 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 ...
```

- **Replace by mean value where is the attributes value is zero:**
  It may cause problem in KNN-algorithm to deal with the zero value and find the accurate distance.

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

- **Load the library of class for applying the KNN- algorithm:**

```
library(class)
```

- **Normalize the data:**

  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. This is because the distance calculation done in KNN uses feature values. When the one feature values are large than other, that feature will dominate the distance hence the outcome of the KNN. It shapes the data in 0 to 1. The main math of the normalization is

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

```r
noramalize_data <- function(x)
{
  nu= x-min(x)
  dn= max(x)-min(x)
  return(nu/dn)
}
make_data<-as.data.frame(lapply(f_dataset[1:8],noramalize_data))
```

- **Split Data into Test and Train**

There ware taken 70% data in train dataset and 30% data on the test dataset for making the confusion matrix so randomly. After that labels the data with the “Outcome” attribute.

```r
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_dataset[sample_data==1,9]
test_datalabels <- f_dataset[sample_data==2,9]
```

- **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. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. Here, 1st make prediction using the KNN function. After that, the prediction data was compare with the actual data and make a confusion matrix bellow.

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

con_matrix
```

- **Output:**

Here, True Negative =126, True Positive=36, False positive =14, False Negative= 50

```
                 prediction
test_datalabels    0   1
                0 126  14
                1  50  36
```

0=negative 1=positive		prediction	
		0	1
Accentual result	0	126(TN)	14(FP)
	1	50(FN)	36(TP)

true positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.

true negatives (TN): We predicted no, and they don't have the disease.

false positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")

false negatives (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

Total = 126+14+36+50 = 226

This is a list of rates that are often computed from a confusion matrix for a binary classifier:

Accuracy: Overall, how often is the classifier correct?

(TP+TN)/total = (36+126)/226 = 0.7168

Misclassification Rate: Overall, how often is it wrong?

(FP+FN)/total = (14+50)/226 = 0.283

- **Find 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,'%'))
```

- **Output:**

```
> print(paste('ACCURACY OF THIS MODEL IS = ',Accuracy(
[1] "ACCURACY OF THIS MODEL IS =  71.6814159292035 %"
```

- **Find accuracy based on different K values 1 to 50:**

```r
list_k <- c(1:50)
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)
```

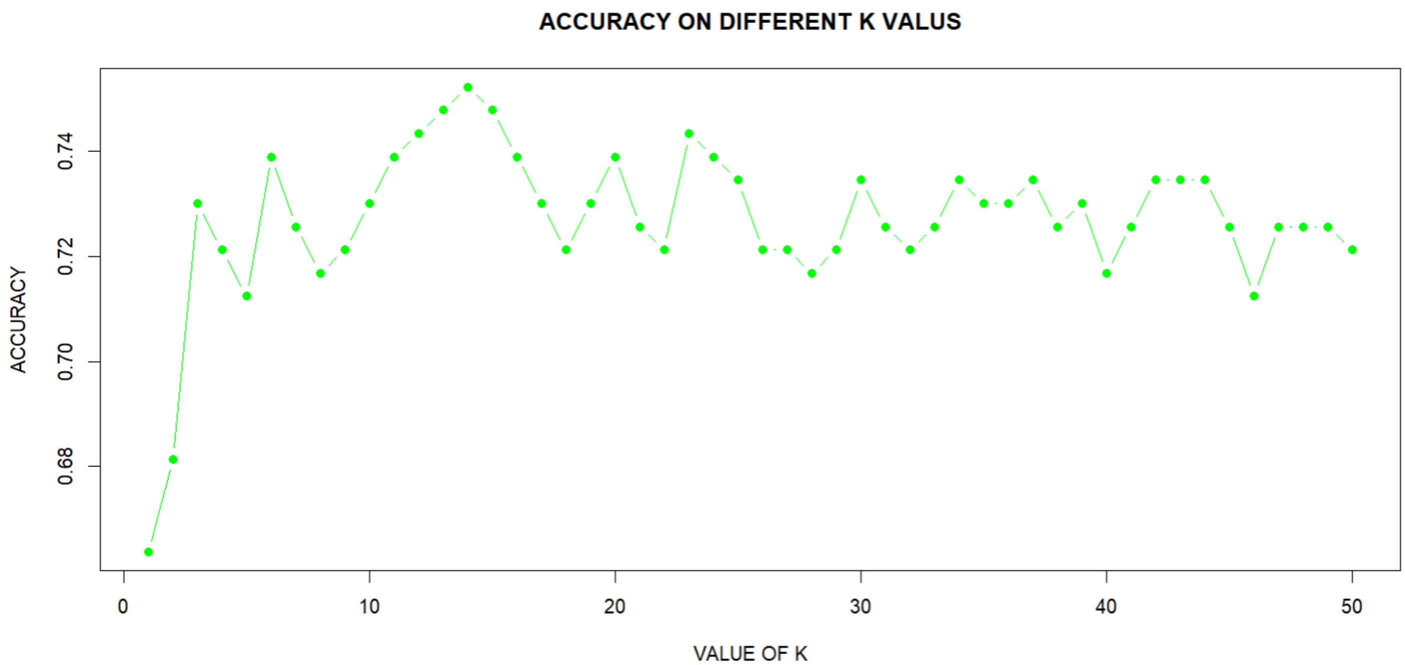
- **Output:**

```
> 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
[27,]     27    0.7212389
[28,]     28    0.7168142
[29,]     29    0.7212389
[30,]     30    0.7345133
[31,]     31    0.7256637
[32,]     32    0.7212389
[33,]     33    0.7256637
[34,]     34    0.7345133
[35,]     35    0.7300885
[36,]     36    0.7300885
[37,]     37    0.7345133
[38,]     38    0.7256637
[39,]     39    0.7300885
[40,]     40    0.7168142
[41,]     41    0.7256637
[42,]     42    0.7345133
```

- **Scatter plot of all accuracy with the K value:**

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

- **Output:**



Here we can see, there are found maximum accuracy was in between 10 to 20 values of K.

DISCUSSION:

There was total 768 data in a dataset. And there predicted the data by 70% train and 30% test dataset selected randomly. In this case if there were more data as well as more attributes given then the prediction will be more meaningful and accurate.