American International University-Bangladesh(AIUB)

Final Project Report

Fall Semester [2022-2023]

Topic: *Diabetes Detection Using KNN and finding correlation between different factor contributing to diabetes*

Course: *Introduction To Data Science*

Section: *D*

Submitted By

Md.Miftahul Alam
18-38839-3

Submitted To

TOHEDUL ISLAM

Assistant Professor, Computer Science

AIUB
Email: islamtohedul@aiub.edu

Date: 11/12/2022

Contents

Section 1: Project Overview

Medical sectors need detailed study on symptoms, factors and features which can be direct or indirect reasons for the diseases. Here I worked on Data obtained on diabetes patients from Kaggle which gave us insights about the major reasons which cause diabetes or features found in diabetes patients. Here we applied KNN algorithm. K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. My reason to use KNN is because it produces extremely accurate predictions, the KNN algorithm can compete with the most accurate models. As a result, the KNN algorithm can be used for applications that need high accuracy but don't need a model that can be read by humans.

Section 2: Dataset Overview

We collected our dataset from Kaggle. Our dataset name is “Pima Indian Diabetes dataset” which was originated from National Institute of Diabetes and Digestive and Kidney Diseases. Based on specific diagnostic measurements present in the dataset, the dataset's goal is to predict whether a patient has diabetes or not. These instances were chosen from a larger database under a number of restrictions. Particularly, all patients in this facility are Pima Indian women who are at least 21 years old. The Pima are a tribe of Native Americans who inhabited what is now central and southern Arizona, as well as the states of Sonora and Chihuahua in northwest Mexico. All patients were female.

Url: ***https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database***

It has total 768 instances and 9 features/attributes. Total size of the dataset is 23.87 Kilobytes.

Feature Details:

Predictor Variable:

1. “Pregnancies” [Number of times pregnant]
2. “Glucose” [Plasma glucose concentration a 2-hours in an oral glucose tolerance test]
3. “BloodPressure” [Diastolic blood pressure (mm Hg)]
4. “SkinThickness” [Triceps skin fold thickness (mm)]
5. “Insulin” [2-Hour serum insulin (mu U/ml)]
6. “BMI” [Body mass index (weight in kg/(height in m)^2)]
7. “DiabetesPedigreeFunction” [Diabetes pedigree function]
8. “Age” [Age (years)]

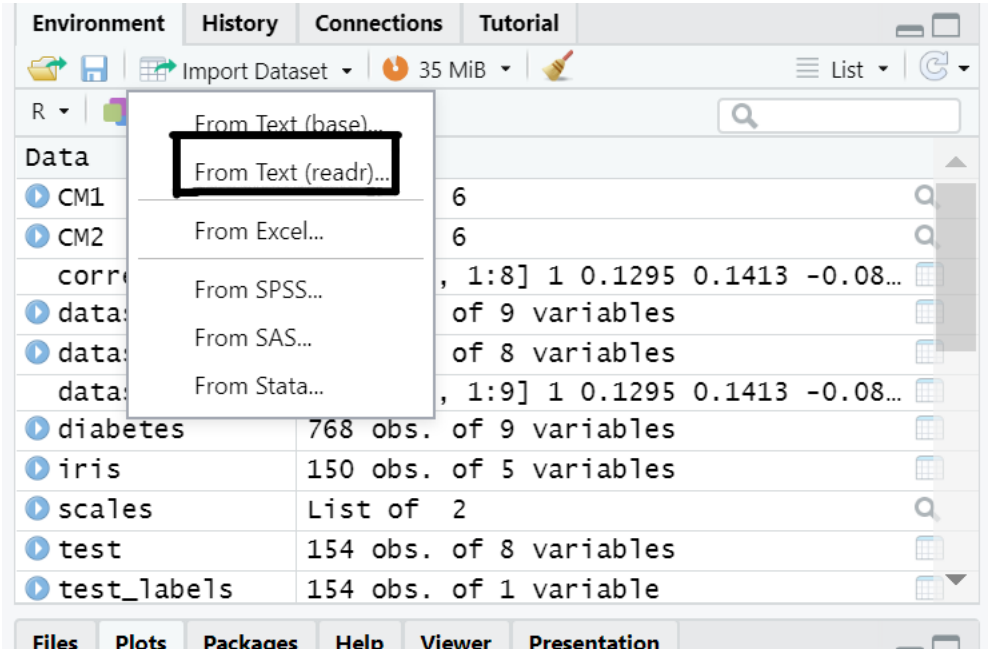
Target Variable:

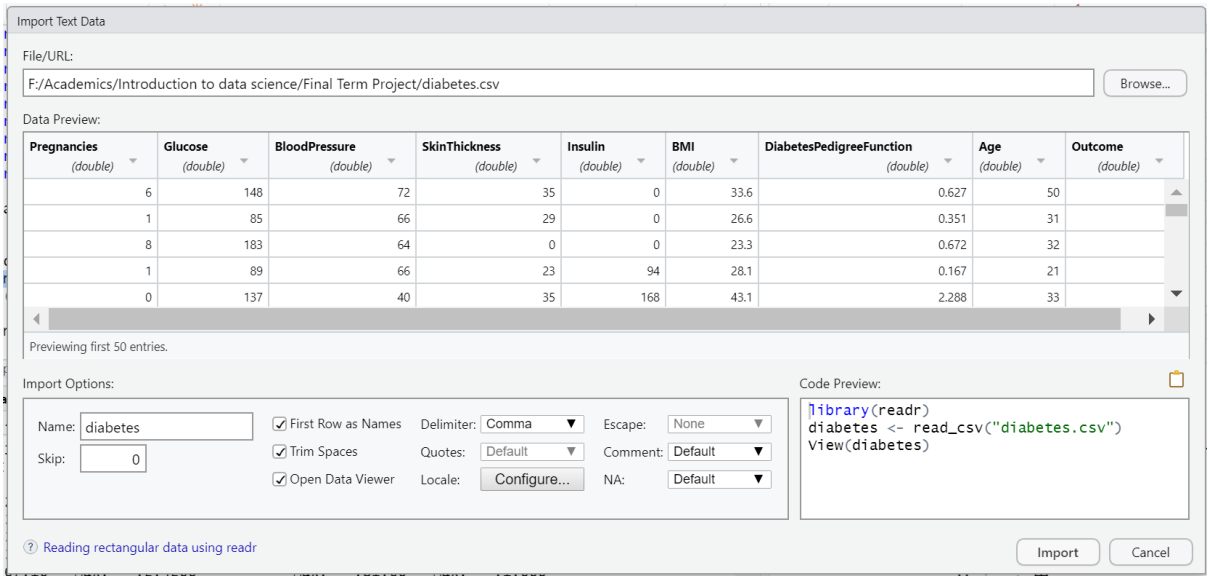
9. "Outcome", *(0/1)*. 0 means the patient does not have diabetes and 1 means the patient has diabetes.

It is a binary classification since there are two values – 0 or 1.

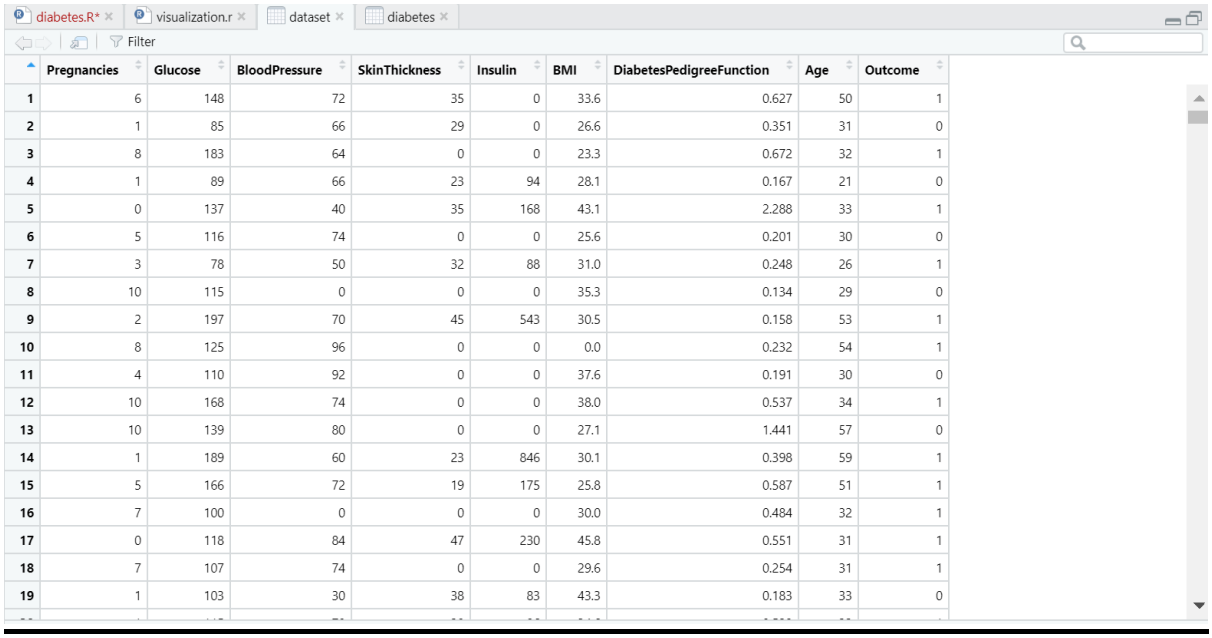
Section 3: Model Development

Step1: Loading dataset and viewing it.






```
10
11  dataset = diabetes
12
13
14  head(dataset)
15  summary(dataset)
16  str(dataset)
17
```



diabetes.R* | visualization.r | dataset | diabetes

Filter

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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

```
> head(dataset)
# A tibble: 6 × 9
  Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI DiabetesP…¹   Age Outcome
        <db l>   <db l>         <db l>         <db l>   <db l> <db l>       <db l> <db l>   <db l>
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       2.29     33       1
6           5     116            74             0        0  25.6       0.201    30       0
# … with abbreviated variable name 'DiabetesPedigreeFunction
> summary(dataset)
  Pregnancies        Glucose      BloodPressure    SkinThickness      Insulin
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00   1st Qu.:  0.0
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.2
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.0
      BMI        DiabetesPedigreeFunction      Age            Outcome
 Min.   : 0.00   Min.   :0.0780           Min.   :21.00   Min.   :0.000
 1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00   1st Qu.:0.000
 Median :32.00   Median :0.3725           Median :29.00   Median :0.000
 Mean   :31.99   Mean   :0.4719           Mean   :33.24   Mean   :0.349
 3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00   3rd Qu.:1.000
 Max.   :67.10   Max.   :2.4200           Max.   :81.00   Max.   :1.000
```

```
> str(dataset)
spc_tbl_ [768 × 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Pregnancies             : num [1:768] 6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : num [1:768] 148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : num [1:768] 72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : num [1:768] 35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : num [1:768] 0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num [1:768] 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num [1:768] 0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : num [1:768] 50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : num [1:768] 1 0 1 0 1 0 1 0 1 1 ...
 - attr(*, "spec")=
  .. cols(
  ..   Pregnancies = col_double(),
  ..   Glucose = col_double(),
  ..   BloodPressure = col_double(),
  ..   SkinThickness = col_double(),
  ..   Insulin = col_double(),
  ..   BMI = col_double(),
  ..   DiabetesPedigreeFunction = col_double(),
  ..   Age = col_double(),
  ..   Outcome = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

Step 2: Normalization. (Min-Max Method)

One of the most popular techniques for normalizing data is min-max normalization. Every feature's minimum and maximum values are each converted to a 0 and a 1, respectively, while all other values are converted to a decimal between 0 and 1.

```
normalize = function(x){

  return((x-min(x))/(max(x)-min(x)))

}

dataset_norm <- as.data.frame(lapply(dataset[,1:8], normalize))

head(dataset_norm)
```

```
> normalize = function(x){
+
+   return((x-min(x))/(max(x)-min(x)))
+
+ }
>
> dataset_norm <- as.data.frame(lapply(dataset[,1:8], normalize))
>
> head(dataset_norm)
  Pregnancies   Glucose BloodPressure SkinThickness   Insulin       BMI
1  0.35294118 0.7437186     0.5901639     0.3535354 0.0000000 0.5007452
2  0.05882353 0.4271357     0.5409836     0.2929293 0.0000000 0.3964232
3  0.47058824 0.9195980     0.5245902     0.0000000 0.0000000 0.3472429
4  0.05882353 0.4472362     0.5409836     0.2323232 0.1111111 0.4187779
5  0.00000000 0.6884422     0.3278689     0.3535354 0.1985816 0.6423249
6  0.29411765 0.5829146     0.6065574     0.0000000 0.0000000 0.3815201
  DiabetesPedigreeFunction       Age
1               0.23441503 0.4833333
2               0.11656704 0.1666667
3               0.25362938 0.1833333
4               0.03800171 0.0000000
5               0.94363792 0.2000000
6               0.05251921 0.1500000
> |
```

Step3: Data Splitting.

The data was split into 8:2 ratio between Training set and Test set. Training Set having random 80% data and test set having rest 20% data.

```
set.seed(123)
dataset_spl<-sample(1:nrow(dataset_norm),size=nrow(dataset_norm)*0.8,replace = FALSE)
train<-dataset_norm[dataset_spl,]
test<-dataset_norm[-dataset_spl,]
```

Training Set -> 614 instances

Test set ->154 instances

Step4:Dataframes for target variable

Creating Separate dataframes for target variable Outcome from test and training set named as Train_Labels and Test_Labels.

```
37
38  train_labels <- dataset[dataset_spl,9]
39  test_labels <-  dataset[-dataset_spl,9]
40  train_labels
41  test_labels
42  length(train_labels)
43  length(test_labels)
```

```
> View(train)
> train_labels <- dataset[dataset_spl,9]
> test_labels <-  dataset[-dataset_spl,9]
> train_labels
# A tibble: 614 × 1
   Outcome
     <dbl>
 1       1
 2       0
 3       0
 4       0
 5       0
 6       0
 7       1
 8       0
 9       1
10       1
# … with 604 more rows
# ℹ Use `print(n = ...)` to see more rows
> test_labels
# A tibble: 154 × 1
   Outcome
     <dbl>
 1       1
 2       1
 3       1
 4       1
 5       0
 6       1
 7       0
 8       1
 9       0
10       0
# … with 144 more rows
# ℹ Use `print(n = ...)` to see more rows
> |
```

Step5:Building the model

Here we considered value of K to be both 27 and 28.

Model1:k=28

Model2:k=27

```
cl = train_labels[,, drop = TRUE]
nrow(dataset)
round(sqrt(nrow(dataset)))
model1<- knn(train=train,test=test,cl,k=(round(sqrt(nrow(dataset)))))
model1
model2 <- knn(train,test,cl,k=27)
model2
NROW(test_labels)
```

```
> cl = train_labels[,, drop = TRUE]
> nrow(dataset)
[1] 768
> round(sqrt(nrow(dataset)))
[1] 28
> model1<- knn(train=train,test=test,cl,k=(round(sqrt(nrow(dataset)))))
> model1
  [1] 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 1 0 0 1 0 1 0
 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
[111] 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
Levels: 0 1
> model2 <- knn(train,test,cl,k=27)
> model2
  [1] 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 1 0 0 1 0 1 0
 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
[111] 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
Levels: 0 1
```

Step6: Model Evaluation.

Checking accuracy

Model 1 accuracy = 75.32%

Model 2 accuracy = 75.97%

```
ACC.model1 <- 100 * sum(test_labels[,,drop=TRUE] == model1)/NROW(test_labels)
ACC.model1
ACC.model2 <- 100 * sum(test_labels[,,drop=TRUE] == model2)/NROW(test_labels)
ACC.model2
```

```
> ACC.model1
[1] 75.32468
> ACC.model2 <- 100 * sum(test_labels[,,drop=TRUE] == model2)/NROW(test_labels)
> ACC.model2
[1] 75.97403
```

confusion Matrix using table() function

```
table1 <- table(model1 ,test_labels[,, drop=TRUE])
table1
table2 <- table(model2,test_labels[,, drop=TRUE])
table2
```

```
> table1 <- table(model1 ,test_labels[,, drop=TRUE])
> table1

model1  0  1
     0 91 27
     1 11 25
> table2 <- table(model2,test_labels[,, drop=TRUE])
> table2

model2  0  1
     0 91 26
     1 11 26
```

Detailed analysis using confusionMatrix() function

```
CM1 <- confusionMatrix(table(model1 ,test_labels[,, drop=TRUE]))
CM2 <- confusionMatrix(table(model1,test_labels[,, drop=TRUE]))
CM1
CM2
CM1$byClass
CM2$byClass
```

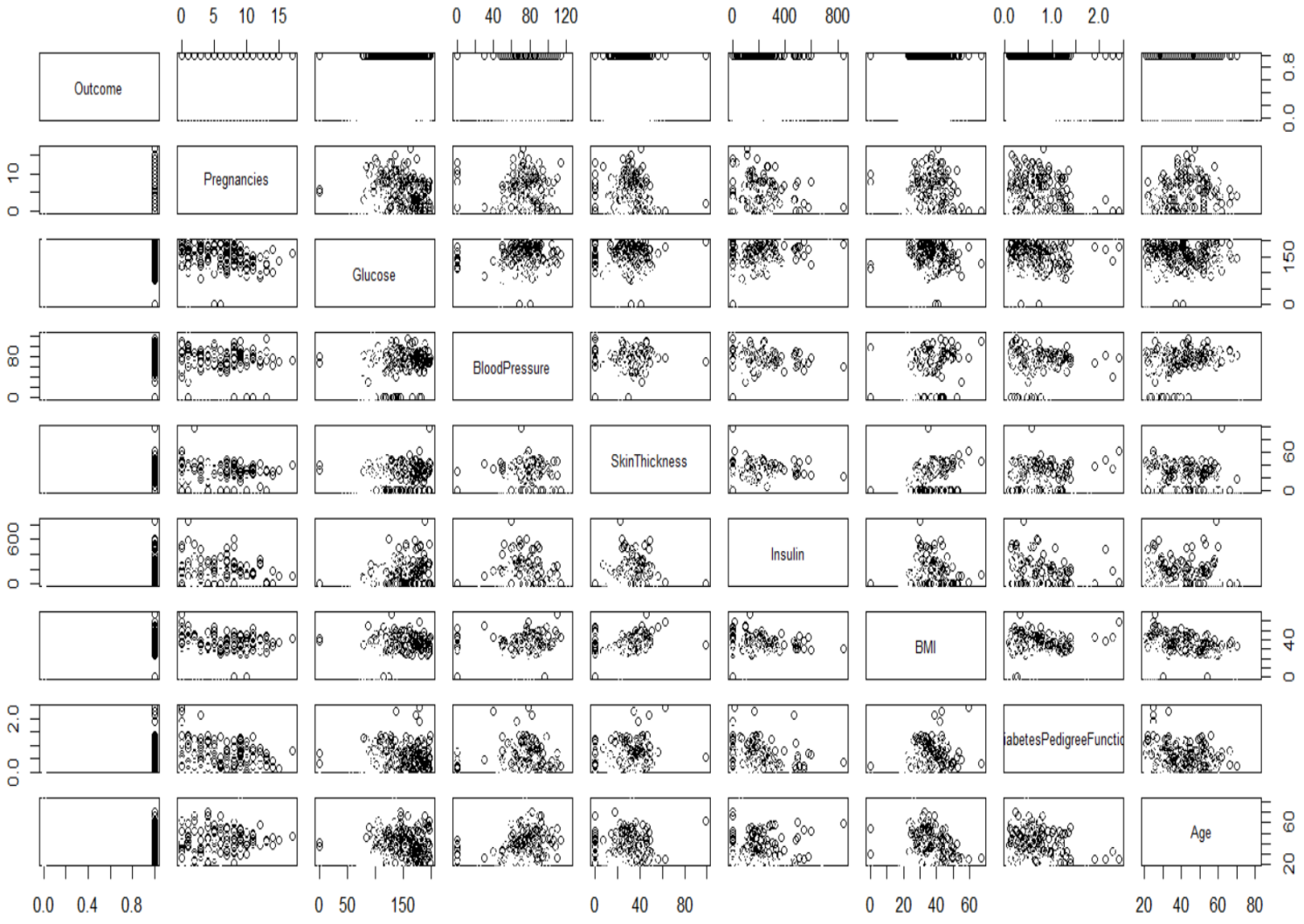
```
> CM1
Confusion Matrix and Statistics


model1  0  1
     0 91 27
     1 11 25

               Accuracy : 0.7532
                 95% CI : (0.6774, 0.8191)
    No Information Rate : 0.6623
    P-Value [Acc > NIR] : 0.009418

                  Kappa : 0.4033

 Mcnemar's Test P-Value : 0.014961

            Sensitivity : 0.8922
            Specificity : 0.4808
         Pos Pred Value : 0.7712
         Neg Pred Value : 0.6944
             Prevalence : 0.6623
         Detection Rate : 0.5909
   Detection Prevalence : 0.7662
      Balanced Accuracy : 0.6865

       'Positive' Class : 0
```

```
> CM2
Confusion Matrix and Statistics


model2  0  1
     0 91 26
     1 11 26

               Accuracy : 0.7597
                 95% CI : (0.6844, 0.8248)
    No Information Rate : 0.6623
    P-Value [Acc > NIR] : 0.005717

                  Kappa : 0.422

 Mcnemar's Test P-Value : 0.021359

            Sensitivity : 0.8922
            Specificity : 0.5000
         Pos Pred Value : 0.7778
         Neg Pred Value : 0.7027
             Prevalence : 0.6623
         Detection Rate : 0.5909
   Detection Prevalence : 0.7597
      Balanced Accuracy : 0.6961

       'Positive' Class : 0
```

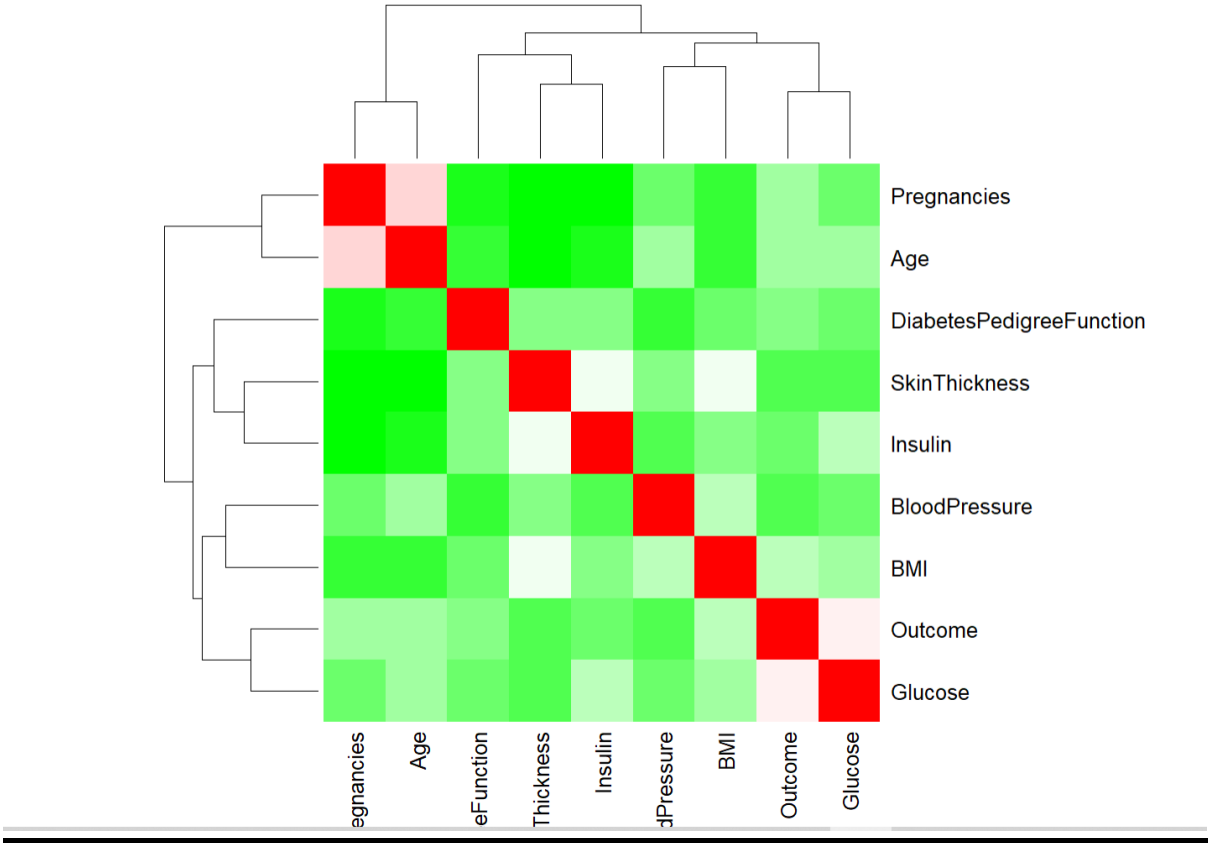
```
> CM1$byClass
        Sensitivity         Specificity      Pos Pred Value      Neg Pred Value           Precision
     Recall                  F1
         0.8921569           0.4807692           0.7711864           0.6944444           0.7711864
  0.8921569           0.8272727
        Prevalence      Detection Rate Detection Prevalence    Balanced Accuracy
         0.6623377           0.5909091           0.7662338           0.6864630
> CM2$byClass
        Sensitivity         Specificity      Pos Pred Value      Neg Pred Value           Precision
     Recall                  F1
         0.8921569           0.5000000           0.7777778           0.7027027           0.7777778
  0.8921569           0.8310502
        Prevalence      Detection Rate Detection Prevalence    Balanced Accuracy
         0.6623377           0.5909091           0.7597403           0.6960784
```

Step 7: Data Visualization

```
pairs(dataset_norm)
pairs(Pregnancies~., dataset, col=dataset$Outcome)
palette = colorRampPalette(c("green", "white", "red")) (20)
heatmap(x = dataset.cor, col = palette, symm = TRUE)
```



Scatter Plot by class.


Pregnancies
Age
DiabetesPedigreeFunction
SkinThickness
Insulin
BloodPressure
BMI
Outcome
Glucose
egnancies
Age
eFunction
Thickness
Insulin
dPressure
BMI
Outcome
Glucose

Heatmap

Section 4: Findings and Discussion

Classification Metrics	**Result**
Predictive Accuracy	Model1- ***75.32%*** Model2- ***75.97%***
Precision	Model1 -77.11% Model2- 77.77%
Recall	Model1- 89.21% Model2- 89.21%
F1-score	Model1- 82.72% Model2- 83.105%
Positive Predicted Value	Model1 - 77.11% Model2 - 77.77%
Negative Predicted Value	Model1- 69.44% Model2- 70.27%
Value of K	Model1: k=28 Model2: k=27

Confusion Matrix

Model1(k=28)

```
model1  0  1
     0 91 27
     1 11 25
```

Model2(k=27)

```
model2  0  1
     0 91 26
     1 11 26
```

```
> dataset.cor = cor(dataset)
> dataset.cor
                          Pregnancies    Glucose BloodPressure
Pregnancies                1.00000000 0.12945867    0.14128198
Glucose                    0.12945867 1.00000000    0.15258959
BloodPressure              0.14128198 0.15258959    1.00000000
SkinThickness             -0.08167177 0.05732789    0.20737054
Insulin                   -0.07353461 0.33135711    0.08893338
BMI                        0.01768309 0.22107107    0.28180529
DiabetesPedigreeFunction -0.03352267 0.13733730    0.04126495
Age                        0.54434123 0.26351432    0.23952795
Outcome                    0.22189815 0.46658140    0.06506836
                          SkinThickness     Insulin        BMI
Pregnancies                 -0.08167177 -0.07353461 0.01768309
Glucose                      0.05732789  0.33135711 0.22107107
BloodPressure                0.20737054  0.08893338 0.28180529
SkinThickness                1.00000000  0.43678257 0.39257320
Insulin                      0.43678257  1.00000000 0.19785906
BMI                          0.39257320  0.19785906 1.00000000
DiabetesPedigreeFunction     0.18392757  0.18507093 0.14064695
Age                         -0.11397026 -0.04216295 0.03624187
Outcome                      0.07475223  0.13054795 0.29269466
                          DiabetesPedigreeFunction         Age
Pregnancies                             -0.03352267  0.54434123
Glucose                                  0.13733730  0.26351432
BloodPressure                            0.04126495  0.23952795
SkinThickness                            0.18392757 -0.11397026
Insulin                                  0.18507093 -0.04216295
BMI                                      0.14064695  0.03624187
DiabetesPedigreeFunction                 1.00000000  0.03356131
Age                                      0.03356131  1.00000000
Outcome                                  0.17384407  0.23835598
                             Outcome
Pregnancies               0.22189815
Glucose                   0.46658140
BloodPressure             0.06506836
SkinThickness             0.07475223
Insulin                   0.13054795
BMI                       0.29269466
DiabetesPedigreeFunction 0.17384407
Age                       0.23835598
Outcome                   1.00000000
>
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree function	Age	Outcome
Pregnancies	1.00	0.1294	0.14	-0.081	-0.07	0.017	-0.033	0.54	0.22
Glucose	0.129	1.00	0.15	0.05	0.33	0.22	0.13	0.26	0.46
BP	0.14	0.15	1.00	0.207	0.088	0.28	0.041	0.23	0.065
Skin thickness	-0.081	0.05	0.207	1.00	0.43	0.39	0.183	-0.11	0.074
Insulin	-0.073	0.331	0.088	0.43	1.00	0.19	0.185	-0.04	0.013
BMI	0.017	0.221	0.281	0.39	0.19	1.00	0.14	0.036	0.29
Diabetes Pedigree Function	-0.033	0.137	0.041	0.18	0.18	0.14	1.00	0.033	0.17
Age	0.263	0.263	0.239	-0.11	-0.04	0.03	0.03	1.00	0.23
Outcome	0.221	0.466	0.065	0.074	0.13	0.29	0.17	0.23	1.00

As we can see from above tables, we can observe that using KNN algorithm for the dataset with K=28 vs K=27 we get slightly higher accuracy with k=27 which is named as model2. The accuracy is not that much high and it is not bad at the same time. It is average performance around approximately 75%. Since in target variable majority classification were 0 so the model predicted it correctly but were unable to predict same way incase of Negative classification 1 because KNN is a lazy algorithm which sometimes fails to classify minority classes in classification. In main dataset, outcome which is class variable, 268 patients were classified as having diabetes(1) and 500 patients were classified as not having diabetes(0)

I also worked on finding correlation using cor () function which also gave us much insights. For the final outcome the factors played most role are : Glucose, BMI, Age and pregnancies. Correlation does not mean causation but it gives insight they have some connections.

Conclusion:

There was an imbalance of target variable classification. Number of positive classes were higher than number of negative classes. So, the model1 and model2 could not achieve highest accuracy.

On the other hand correlation gave us a lot of hints. People with growing age should take special care of their glucose level and weight. People specially female with pregnancies sometimes gain weight and their glucose level increases which has risk to get diabetes. Three factors should be kept in mind by young people that they should keep glucose and weight in control from very early age and follow healthy lifestyle. BMI is connected weight so is diabetes.

References:

1. *Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261--265). IEEE Computer Society Press.*

2. *Diabetes Dataset-Classification using K-Nearest Neighbors Algorithm.*
   Authors: Masum Shah Junayed ,University of Connecticut
   Afsana Ahsan Jeny, Wayne State University

3. https://rpubs.com/Shraddha20/Diabetes_Prediction_using_R
4. https://www.geeksforgeeks.org/k-nn-classifier-in-r-programming/