Diabetics Prediction Using Classification Models

Eddah

2023-12-18

The purpose for this analysis was to predict if a person is diabetic or not based on given variables.
The following models were used: Logistic Regression,K Nearest Neighbors,Support Vector Machine and Decision trees.
Logistic Regression had the highest accuracy.

```
#Getting  and the working directory
getwd()
```

```
## [1] "C:/Users/User/Desktop/R projects/new_project"
```

```
#setting working directory
setwd("C:/Users/User/Desktop/R projects/new_project")
```

```
#loading the dataset
data<-read.csv('diabetes.csv',header=TRUE)
```

```
#head of data
head(data)
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       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
```

```
# number of columns in (data)
length(data)
```

```
## [1] 9
```

```
#number of rows in data
nrow(data)
```

```
## [1] 768
```

```
#structure of data
str(data)
```

```
## '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 ...
```

```
#summary statistics
summary(data)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
#checking if there are null values
sum(is.na(data))
```

```
## [1] 0
```

```
#null values for each column
colSums(is.na(data))
```

```
##              Pregnancies                  Glucose            BloodPressure
##                        0                        0                        0
##            SkinThickness                  Insulin                      BMI
##                        0                        0                        0
## DiabetesPedigreeFunction                      Age                  Outcome
##                        0                        0                        0
```

```
#checking if there are duplicates in data
data[duplicated(data)]
```

```
## data frame with 0 columns and 768 rows
```

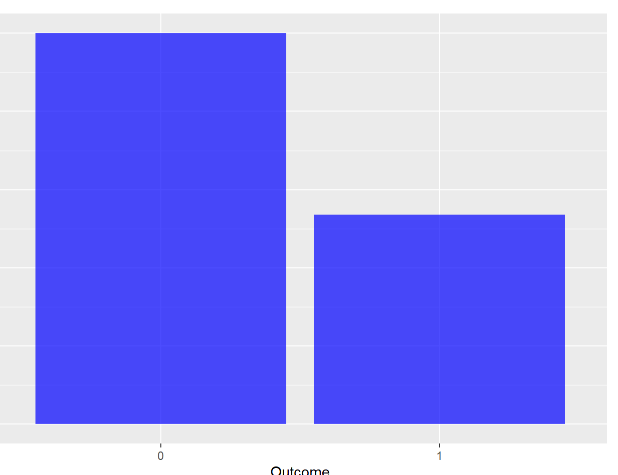
Explonatory Data Analyisis

```
#installing ggplot2
if (!require("ggplot2")) {
  install.packages("ggplot2")
  library("ggplot2")
}
```

```
## Loading required package: ggplot2
```

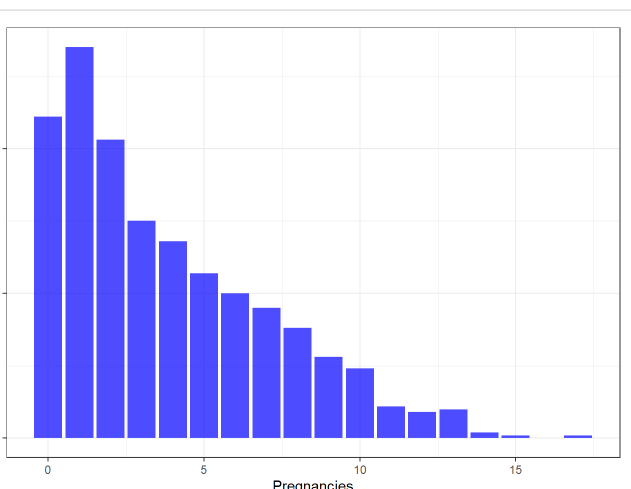
```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
#barplot on Outcome
ggplot(data, aes(x=reorder(Outcome,Outcome, function(x)-length(x)))) + geom_bar(fill='blue',alpha=0.7)+labs(x='Outcome')
```

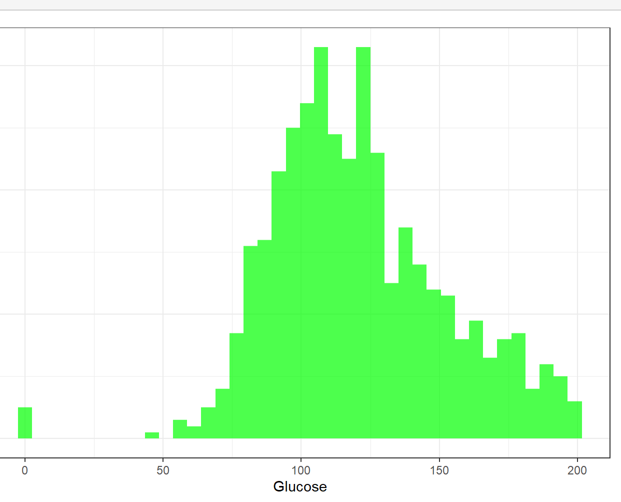



```
#histogram on pregnancies
ggplot(data,aes(x=Pregnancies))+geom_bar(fill='blue',bins=40,alpha=0.7)+theme_bw()
```

```
## Warning in geom_bar(fill = "blue", bins = 40, alpha = 0.7): Ignoring unknown
## parameters: `bins`
```

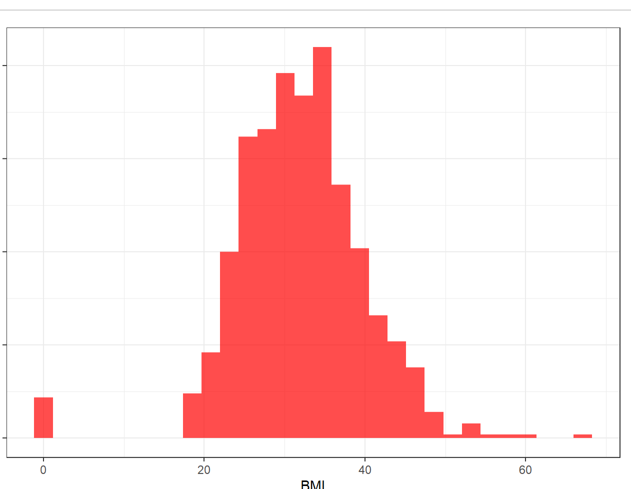



```
#histogram on Glucose
ggplot(data,aes(x=Glucose))+geom_histogram(fill='green',bins=40,alpha=0.7)+theme_bw()
```

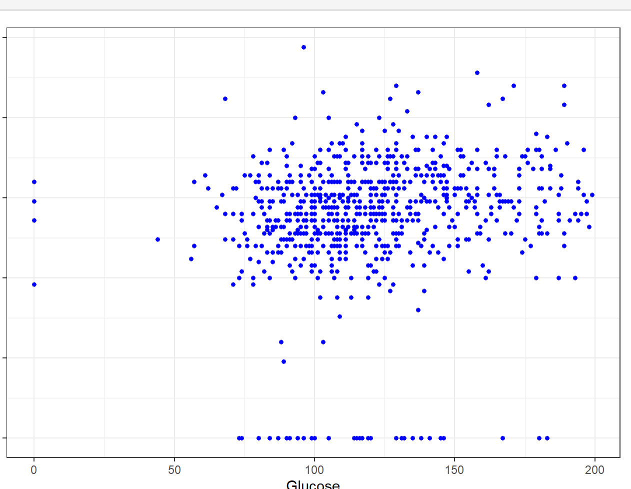



```
#histogram on BMI
ggplot(data,aes(x=BMI))+geom_histogram(fill='red',alpha=0.7)+theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

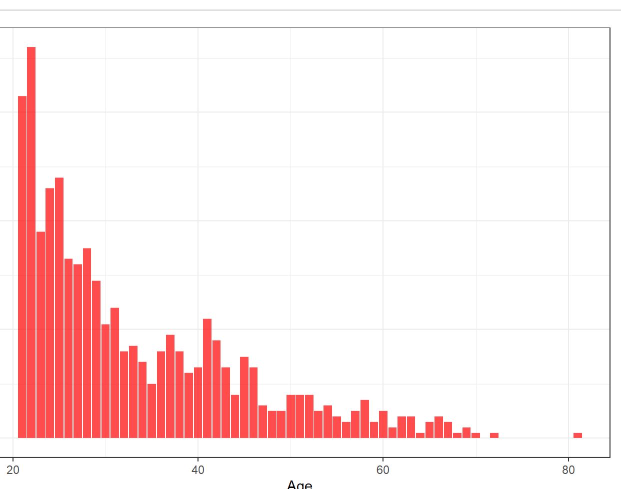



```
# Creating a scatterplot on Glusose against Blood Pressure
ggplot(data,aes(x=Glucose,y=BloodPressure))+ geom_point(color = "blue",cex = 1.3)+theme_bw()
```




```
#histogram on
ggplot(data,aes(x=Age))+geom_bar(fill='red',bins=80,alpha=0.7)+theme_bw()
```

```
## Warning in geom_bar(fill = "red", bins = 80, alpha = 0.7): Ignoring unknown
## parameters: `bins`
```




```
# CORRELATION
cor(data)
```

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

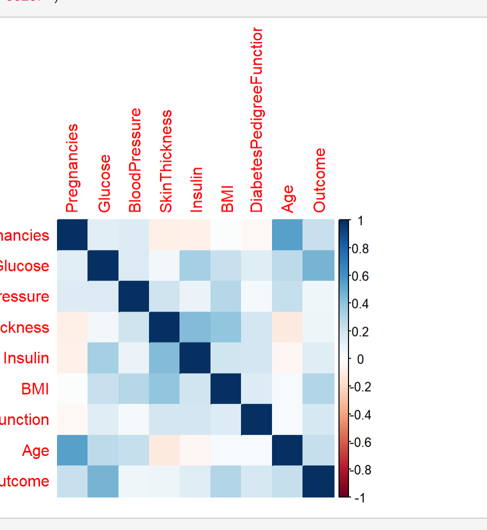
```
#installing corrplot
if (!require("corrplot")) {
  install.packages("corrplot")
  library("corrplot")
}
```

```
## Loading required package: corrplot
```

```
## Warning: package 'corrplot' was built under R version 4.3.2
```

```
## corrplot 0.92 loaded
```

```
corrplot(cor(data),method='color')
```




```
#STANDARDIZING THE DATA

stand.features<- as.data.frame(scale(data[,1:8]))
head(stand.features)
```

```
##   Pregnancies    Glucose BloodPressure SkinThickness    Insulin        BMI
## 1   0.6395305  0.8477713     0.1495433     0.9066791 -0.6924393  0.2038799
## 2  -0.8443348 -1.1226647    -0.1604412     0.5305558 -0.6924393 -0.6839762
## 3   1.2330766  1.9424580    -0.2637694    -1.2873733 -0.6924393 -1.1025370
## 4  -0.8443348 -0.9975577    -0.1604412     0.1544326  0.1232213 -0.4937213
## 5  -1.1411079  0.5037269    -1.5037073     0.9066791  0.7653372  1.4088275
## 6   0.3427574 -0.1530851     0.2528715    -1.2873733 -0.6924393 -0.8108128
##   DiabetesPedigreeFunction         Age
## 1                0.4681869  1.42506672
## 2               -0.3648230 -0.19054773
## 3                0.6040037 -0.10551539
## 4               -0.9201630 -1.04087112
## 5                5.4813370 -0.02048305
## 6               -0.8175458 -0.27558007
```

```
df<-cbind(stand.features,data[9])
```

```
#checking head of df
head(df)
```

```
##   Pregnancies    Glucose BloodPressure SkinThickness    Insulin        BMI
## 1   0.6395305  0.8477713     0.1495433     0.9066791 -0.6924393  0.2038799
## 2  -0.8443348 -1.1226647    -0.1604412     0.5305558 -0.6924393 -0.6839762
## 3   1.2330766  1.9424580    -0.2637694    -1.2873733 -0.6924393 -1.1025370
## 4  -0.8443348 -0.9975577    -0.1604412     0.1544326  0.1232213 -0.4937213
## 5  -1.1411079  0.5037269    -1.5037073     0.9066791  0.7653372  1.4088275
## 6   0.3427574 -0.1530851     0.2528715    -1.2873733 -0.6924393 -0.8108128
##   DiabetesPedigreeFunction         Age Outcome
## 1                0.4681869  1.42506672       1
## 2               -0.3648230 -0.19054773       0
## 3                0.6040037 -0.10551539       1
## 4               -0.9201630 -1.04087112       0
## 5                5.4813370 -0.02048305       1
## 6               -0.8175458 -0.27558007       0
```

```
#TRAIN TEST SPLIT
if (!require("caTools")) {
  install.packages("caTools")
  library("caTools")
}
```

```
## Loading required package: caTools
```

```
## Warning: package 'caTools' was built under R version 4.3.2
```

```
set.seed(100)
sample<-sample.split(df$Outcome,SplitRatio = 0.7)
train=subset(df,sample==TRUE)
test=subset(df,sample==F)
```

```
# MODEL ONE
#LOGISTIC REGRESSION MODEL

reg.model<-glm(Outcome~.,family=binomial(link='logit'),data=train)
summary(reg.model)
```

```
##
## Call:
## glm(formula = Outcome ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -0.85895    0.11336  -7.577 3.52e-14 ***
## Pregnancies               0.35049    0.12486   2.807  0.00500 **
## Glucose                   1.11315    0.13910   8.003 1.22e-15 ***
## BloodPressure            -0.37309    0.12732  -2.930  0.00339 **
## SkinThickness             0.02993    0.13423   0.223  0.82358
## Insulin                  -0.20105    0.12218  -1.645  0.09987 .
## BMI                       0.62897    0.14055   4.475 7.64e-06 ***
## DiabetesPedigreeFunction  0.19420    0.10985   1.768  0.07708 .
## Age                       0.10960    0.12989   0.844  0.39875
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 696.28  on 537  degrees of freedom
## Residual deviance: 523.29  on 529  degrees of freedom
## AIC: 541.29
##
## Number of Fisher Scoring iterations: 5
```

```
#Table on probabilities
fitted.probabilities<-predict(reg.model,test,type='response')
table(test$Outcome,fitted.probabilities>0.5)
```

```
##
##     FALSE TRUE
##   0   139   11
##   1    35   45
```

```
##METRICS
```

```
#accuracy
accuracy<-(139+45)/(139+45+11+35)
print(accuracy)
```

```
## [1] 0.8
```

```
#misclassification.error
misclassification.error<-(11+35)/(139+45+11+35)
print(misclassification.error)
```

```
## [1] 0.2
```

```
#recall
recall<-(139)/(139+11)
print(recall)
```

```
## [1] 0.9266667
```

```
#precision
precision<-(139)/(139+35)
print(precision)
```

```
## [1] 0.7988506
```

```
#MODEL TWO
#K NEAREST NEIGHBORS MODEL
```

```
if (!require("class")) {
  install.packages("class")
  library("class")
}
```

```
## Loading required package: class
```

```
## Warning: package 'class' was built under R version 4.3.2
```

```
Prediction=knn(train[1:8],test[1:8],train$Outcome,k=2)
```

```
miss.error<-mean(test$Outcome!=Prediction)
miss.error
```

```
## [1] 0.2782609
```

```
accuracy<-1-miss.error
print(accuracy)
```

```
## [1] 0.7217391
```

```
k_values <- 1:40
error_rate <- numeric(20)

for (i in 1:40) {
  knn_model <- knn(train[1:8], test[1:8], train$Outcome, k = i)
  error_rate[i] <- mean(test$Outcome != knn_model)
}

error_df <- data.frame(error_rate, k_values)

ggplot(error_df, aes(x = k_values, y = error_rate)) +
  geom_point() +
  geom_line(lty = 'dotted', color = 'red') +
  labs(title = "Error Rate vs. K Value in KNN", x = "K Value", y = "Error Rate")
```


```
#checking accuracy for k=19
Predict=knn(train[1:8],test[1:8],train$Outcome,k=20)

error<-mean(test$Outcome!=Predict)
error
```

```
## [1] 0.2391304
```

```
accuracy<-1-error
print(accuracy)
```

```
## [1] 0.7608696
```

```
# MODEL THREE
#Support Vector Machine (SVM) MODEL
if (!require("e1071")) {
install.packages("e1071")

  library("e1071")
}
```

```
## Loading required package: e1071
```

```
## Warning: package 'e1071' was built under R version 4.3.2
```

```
svm.model<-svm(Outcome~.,data=train)
fitted.probabilities<-predict(svm.model,test,type='response')
table(test$Outcome,fitted.probabilities>0.5)
```

```
##
##     FALSE TRUE
##   0   136   14
##   1    51   29
```

```
 #accuracy
accuracy<-(136+29)/(136+29+14+51)
print(accuracy)
```

```
## [1] 0.7173913
```

```
#misclassification.error
misclassification.error<-(14+51)/(136+51+14+29)
print(misclassification.error)
```

```
## [1] 0.2826087
```

```
#recall
recall<-(136)/(136+14)
print(recall)
```

```
## [1] 0.9066667
```

```
#precision
precision<-(136)/(136+51)
print(precision)
```

```
## [1] 0.7272727
```

```
#MODEL FOUR
#DECISION TREES MODEL
if (!require("rpart")) {
  install.packages("rpart")
  library("rpart")
}
```

```
## Loading required package: rpart
```

```
tree<-rpart(Outcome~.,method='class',data=train)
fitted.probabilities<-predict(tree,test,type='vector')
table(test$Outcome,fitted.probabilities)
```

```
##    fitted.probabilities
##       1   2
##   0 144   6
##   1  47  33
```

```
#accuracy
accuracy<-(144+33)/(144+33+6+47)
print(accuracy)
```

```
## [1] 0.7695652
```

```
misclassification.error<-(6+47)/(144+33+6+47)
print(misclassification.error)
```

```
## [1] 0.2304348
```

```
#recall
recall<-(144)/(144+6)
print(recall)
```

```
## [1] 0.96
```

```
#precision
precision<-(144)/(144+47)
print(precision)
```

```
## [1] 0.7539267
```

CONCLUSION
The following are accuracies values from the four models:
1.Logistic regression=80%
2.KNN=76%
3.Decision Trees=77%
4.SVM=72%

Hence Logistic regression showed to be the best model in predictic diabetic disease amongst the four models.