Data Science Capstone Project IDV Learners: Prediction of Diabetes

DD

2021-05-10

Executive Summary

For this project, I have chosen the dataset "Pima Indians Diabetes Database" in Kaggle. It is a subset of the orginal dataset from the National Institute of Diabetes and Digestive and Kidney Diseases. All patients in this dataset are females of atleast 21 years old and Pima Indian heritage.

The dataset contains eight predictor variables -

1) Pregnancies - Number of times pregnant
2) Gluose - 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) DiabetesPedrigreeFunction - Diabetes pedigree function. It is diabetes likelihood function based on family history
8) Age - Age (years)

The outcome/target variable is a binary variable indicating if a person has diabetes or not. The number of observations is 768.

The goal is to predict if a female person of Pima Indian heritage has diabetes or not based on certain predictors. This is binary classification machine learning.

Exploratory data anlysis was done to look at the distributions, correlations and data inconsistencies of the predictors and outcome. Four models were built using machine learning algorthims. The final model chosen is a model fit using k-nearest neighbours algorithm with pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function and age as predictors. The parameter for number of neighbours (k) was tuned using bootstrapping. This model was chosen as it had the highest accuracy and F1 score.The accuracy was 0.7792208 and F1 score was 0.8411215

Analysis

Pre-processing The required libraries are loaded and the dataset is read from the csv file.

```r
#Pre-processing
if(!require(tidyverse)) install.packages("tidyverse",
                                          repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret",
                                      repos = "http://cran.us.r-project.org")
if(!require(ggcorrplot)) install.packages("ggcorrplot",
                                           repos = "http://cran.us.r-project.org")
if(!require(GGally)) install.packages("GGally",
                                       repos = "http://cran.us.r-project.org")
if(!require(zoo)) install.packages("zoo",
                                    repos = "http://cran.us.r-project.org")
if(!require(gridExtra)) install.packages("gridExtra",
                                          repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr",
                                          repos = "http://cran.us.r-project.org")
library(tidyverse)
library(caret)
library(ggcorrplot)
library(GGally)
library(zoo)
library(gridExtra)
library(knitr)

data<- read.csv("diabetes.csv")
```

Data Cleansing The dataset does not have NA values. But, some predictor values that is not supposed to have 0 values have 0 values. This makes us conclude that NA values are coded as 0 values in the predictors Glucose, BloodPressure, SkinThickness, Insulin and BMI. Since the number of rows with these 0 values is quite substantial, we do not want to exclude these rows completely. Instead, this problem is solved by substituting these 0 values with mean value of the respecitve predictor. The predictor variable is converted to factors from integers.

```r
#Data Cleansing
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
```

```
#Converting 0 values to NA
data$Glucose[data$Glucose == 0 ] <- NA
data$BloodPressure[data$BloodPressure == 0 ] <- NA
data$SkinThickness[data$SkinThickness == 0 ] <- NA
data$Insulin[data$Insulin == 0 ] <- NA
data$BMI[data$BMI == 0 ] <- NA

#Converting NA values to mean
data<- na.aggregate(data, FUN=mean)

#Convert outcome to factor
data$Outcome <- as.factor(data$Outcome)
levels(data$Outcome) <- c(FALSE,TRUE)
```

```


#Exploratory Data Analysis
#Summary
summary(data)
```

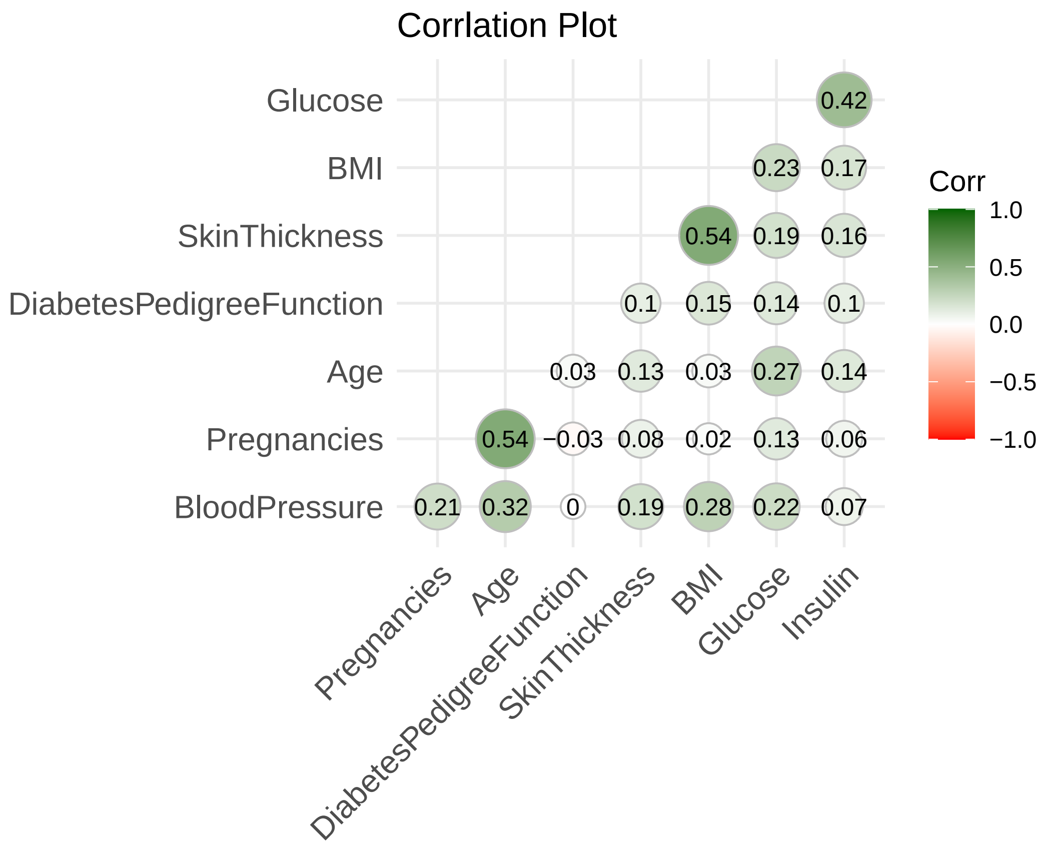
Exploratory Data Analysis

```
##   Pregnancies        Glucose       BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   : 44.00   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.75   1st Qu.: 64.00   1st Qu.:25.00
##  Median : 3.000   Median :117.00   Median : 72.20   Median :29.15
##  Mean   : 3.845   Mean   :121.69   Mean   : 72.41   Mean   :29.15
##  3rd Qu.: 6.000   3rd Qu.:140.25   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.00   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   : 14.0   Min.   :18.20   Min.   :0.0780           Min.   :21.00
##  1st Qu.:121.5   1st Qu.:27.50   1st Qu.:0.2437           1st Qu.:24.00
##  Median :155.5   Median :32.40   Median :0.3725           Median :29.00
##  Mean   :155.5   Mean   :32.46   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:155.5   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
##  FALSE:500
##  TRUE :268
##
##
##
##
```

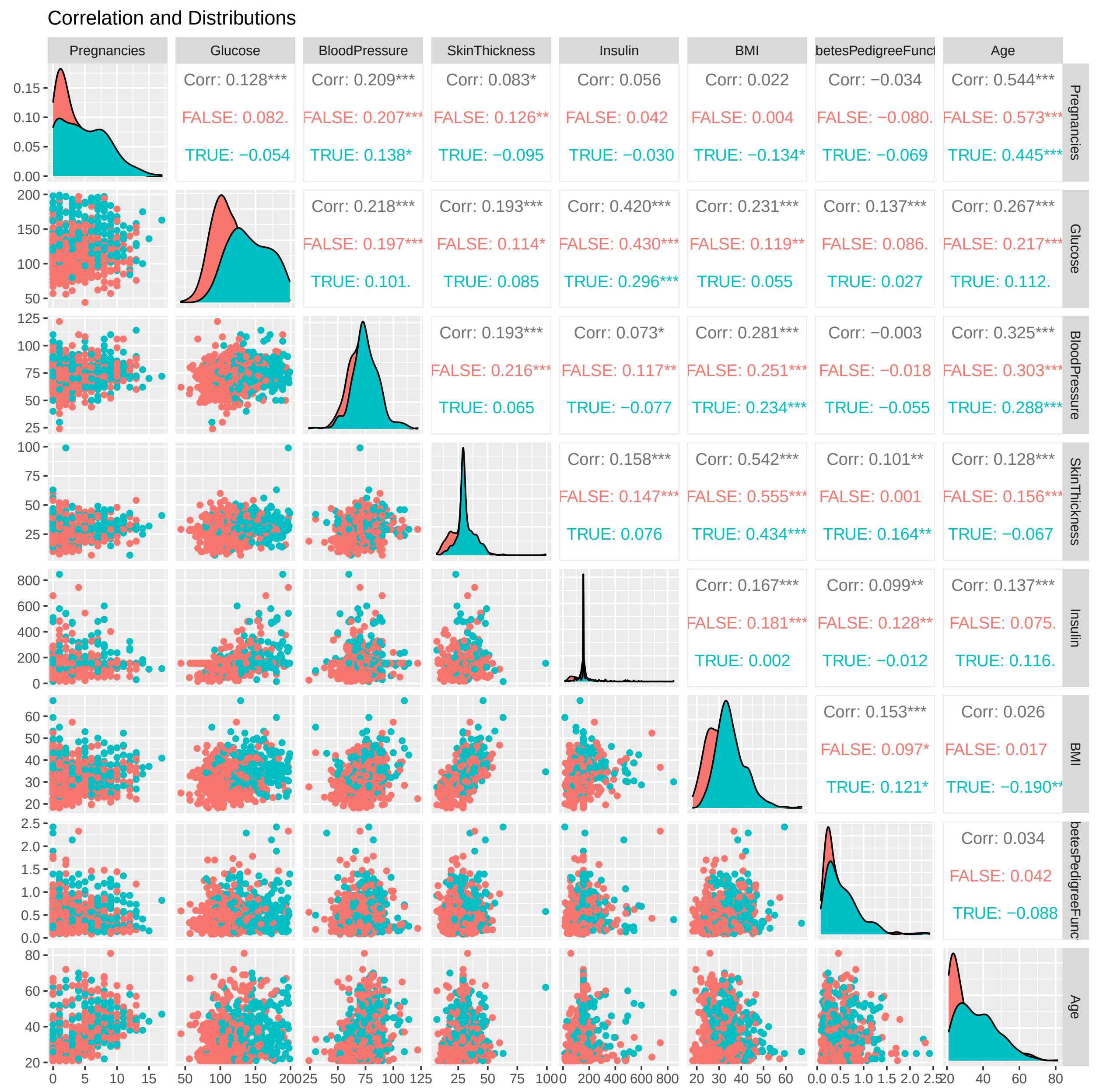
```
data %>%
  summarize(Nr_Patients = n())
```

```
##   Nr_Patients
## 1         768
```

```
#Correlation
ggcorrplot(cor(data[, 1:8]),
           method="circle",
           type = "lower",
           hc.order = TRUE,
           lab = TRUE,
           lab_size = 3,
           colors = c("red","white", "darkgreen"),
           title = "Corrlation Plot")
```

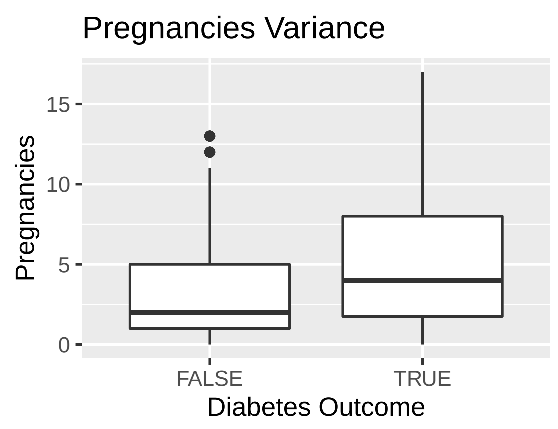


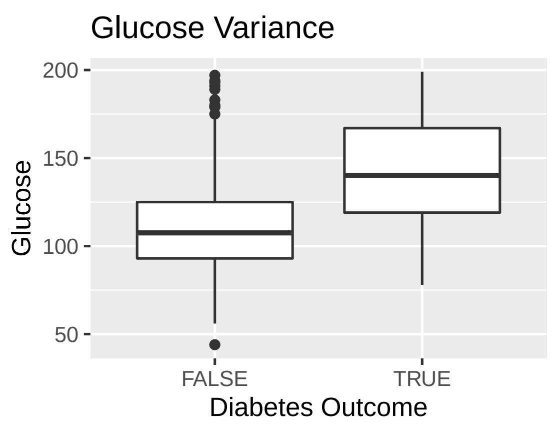
```
#Clusers
ggpairs(data, columns=1:8, aes(color=Outcome)) +
       ggtitle("Correlation and Distributions")
```

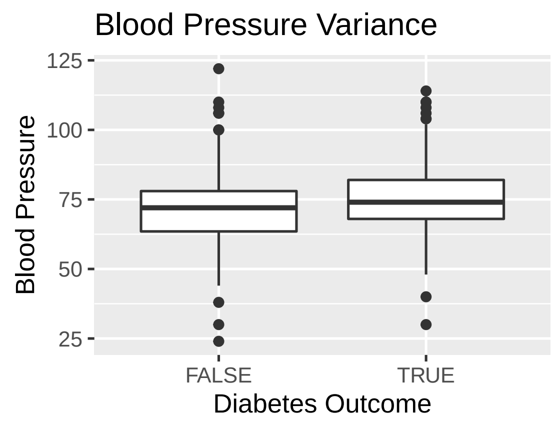


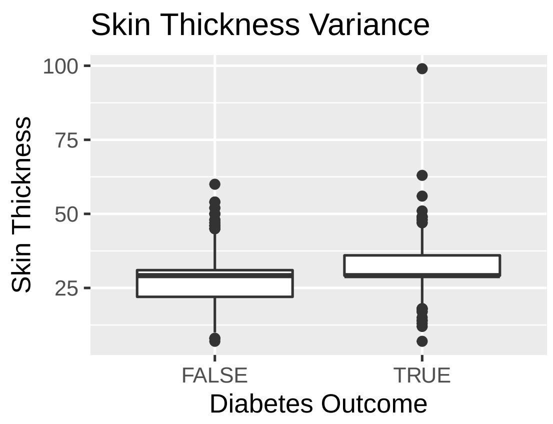
```r
#Variances
#Pregnancies Variance
p1<-ggplot(data, aes(x=Outcome, y=Pregnancies)) +
  geom_boxplot()+
  labs(title="Pregnancies Variance",
       x="Diabetes Outcome",
       y = "Pregnancies")

#Glucose Variance
p2<-ggplot(data, aes(x=Outcome, y=Glucose)) +
  geom_boxplot()+
  labs(title="Glucose Variance",
       x="Diabetes Outcome",
       y = "Glucose")
```

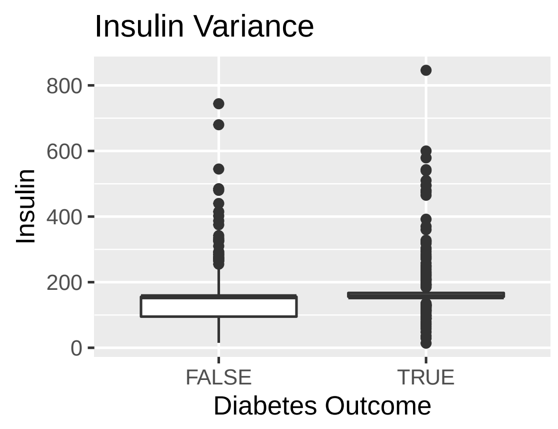
```r
#Blood Pressure Variance
p3<-ggplot(data, aes(x=Outcome, y=BloodPressure)) +
  geom_boxplot()+
  labs(title="Blood Pressure Variance",
       x="Diabetes Outcome",
       y = "Blood Pressure")

#Skin Thickness Variance
p4<-ggplot(data, aes(x=Outcome, y=SkinThickness)) +
  geom_boxplot()+
  labs(title="Skin Thickness Variance",
       x="Diabetes Outcome",
       y = "Skin Thickness")

#Insulin Variance
p5<-ggplot(data, aes(x=Outcome, y=Insulin)) +
  geom_boxplot()+
  labs(title="Insulin Variance",
       x="Diabetes Outcome",
       y = "Insulin")

#BMI Variance
p6<-ggplot(data, aes(x=Outcome, y=BMI)) +
  geom_boxplot()+
  labs(title="BMI Variance",
       x="Diabetes Outcome",
       y = "BMI")

#Diabetes Pedrigree Function Variance
p7<-ggplot(data, aes(x=Outcome, y=DiabetesPedigreeFunction)) +
  geom_boxplot()+
  labs(title="Diabetes Pedrigree Function Variance",
       x="Diabetes Outcome",
       y = "Diabetes Pedigree Function")

#Age Variance
p8<-ggplot(data, aes(x=Outcome, y=Age)) +
  geom_boxplot()+
  labs(title="Age Variance",
       x="Diabetes Outcome",
       y = "Age")

grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, ncol =2)
```

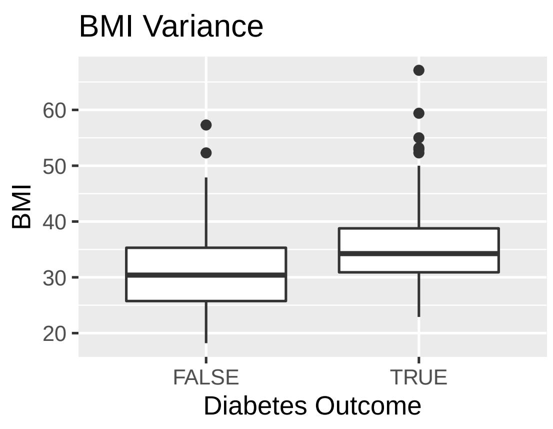

Pregnancies Variance
Pregnancies
0
5
10
15
FALSE
TRUE
Diabetes Outcome

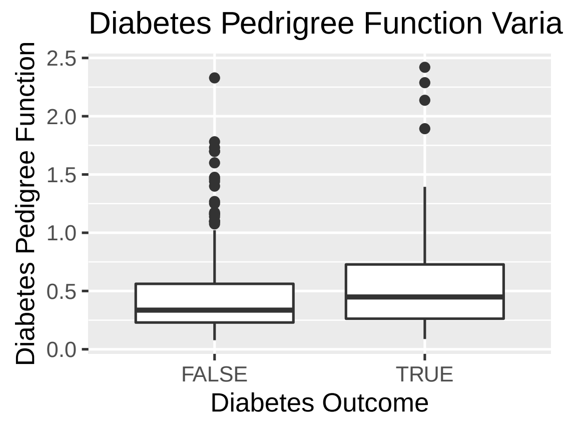

Glucose Variance
Glucose
50
100
150
200
FALSE
TRUE
Diabetes Outcome

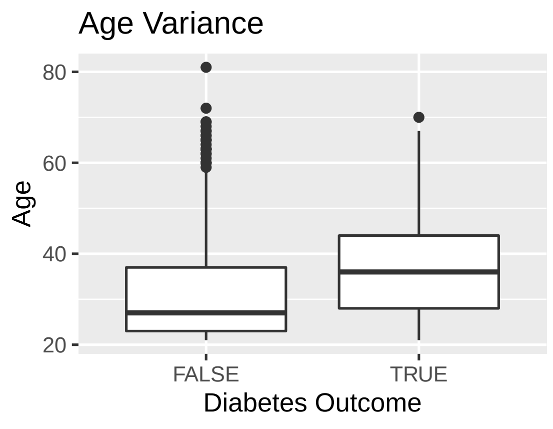

Blood Pressure Variance
Blood Pressure
25
50
75
100
125
FALSE
TRUE
Diabetes Outcome


Skin Thickness Variance
Skin Thickness
25
50
75
100
FALSE
TRUE
Diabetes Outcome


Insulin Variance
Insulin
0
200
400
600
800
FALSE
TRUE
Diabetes Outcome


BMI Variance
BMI
20
30
40
50
60
FALSE
TRUE
Diabetes Outcome


Diabetes Pedrigree Function Varia
Diabetes Pedigree Function
0.0
0.5
1.0
1.5
2.0
2.5
FALSE
TRUE
Diabetes Outcome


Age Variance
Age
20
40
60
80
FALSE
TRUE
Diabetes Outcome

Insights

1) We see the sample size as 768, which is relatively a small dataset for machine learning. This could affect the accuracy of the prediction.
2) Although we see some co-relation between the pairs Pregnancies-Age, SkinThickness-BMI and Glucose-Insulin, they are not particularly high to result in confounding.
3) Looking at the scater plots, we clearly see clusters of true and false outcomes between various pairs of predictors.
4) Looking at the box plots, we also see clear differnces in the quartiles between true and false outcomes espcially for the predictors pregnancies, glucose, skin thickness, insulin, BMI and age.
5) We can proceed to include all the predictors to train the models.

Model Training: Partitioning Training and Test Sets The dataset is partitioned into training (90%) and test (10%) datasets

```
# Splitting dataset into training and test datasets
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = data$Outcome, times = 1,
                                  p = 0.1, list = FALSE)
train_set <- data[-test_index,]
test_set <- data[test_index,]
```

Baseline Model The baseline accuracy and F1 score are calculated by predicting outcome as false for all cases in the test dataset. Any model we fit should have a higher accuracy and F1 score than the baseline model.

```
#Baseline
#predicting false outcome by default
y_hat<- rep(FALSE, nrow(test_set))%>% factor()
#retrieving accuracy from confusion matrix
Baseline_Accuracy<- confusionMatrix(y_hat, test_set$Outcome)$overall[["Accuracy"]]
#retrieving F1 score from confusion matrix
Baseline_F1<- confusionMatrix(y_hat, test_set$Outcome)$byClass[["F1"]]
#storing accuracy and F1 score
Models_Accuracy <- tibble(Model = "Baseline",
                          Accuracy = Baseline_Accuracy,
                          F1 = Baseline_F1)
kable(Models_Accuracy)
```

Model	Accuracy	F1
Baseline	0.6493506	0.7874016

Model 1: Logistic Regression The first model takes all the predictors and uses logistic regression to fit the model.

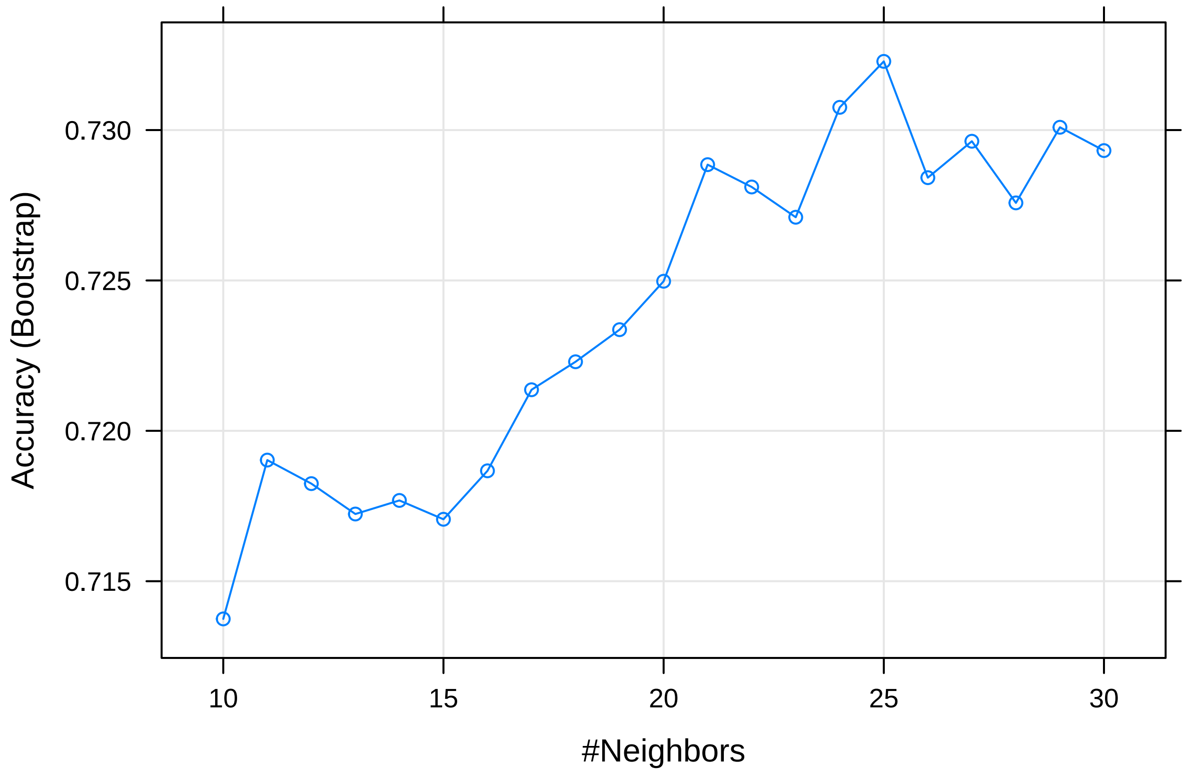
```
#Model 1: Logistic Regression
#training using logistic regression
```

```
Model1_glm <- glm(Outcome ~ ., data=train_set, family = "binomial")
#calculating the probabilities using the fitted model and test set
p_hat <- predict(Model1_glm, newdata = test_set, type="response")
#assigning outcomes based on the probabilities
y_hat <- ifelse(p_hat > 0.5, TRUE, FALSE)%>% factor()
Model1_Accuracy<- confusionMatrix(y_hat, test_set$Outcome)$overall[["Accuracy"]]
Model1_F1<- confusionMatrix(y_hat, test_set$Outcome)$byClass[["F1"]]
Models_Accuracy   <- rbind(Models_Accuracy,
                           c("Model 1: Logistic Regression",
                             Model1_Accuracy,
                             Model1_F1))
kable(Models_Accuracy)
```

Model	Accuracy	F1
Baseline	0.649350649350649	0.78740157480315
Model 1: Logistic Regression	0.753246753246753	0.822429906542056

Model 2: k-Nearest Neighbours The second model takes all the predictors and uses k-nearest neighbours to fit the model. Multiple models are trained for various values of k (number of neighbours) and the k-value from the model with highest accuracy is chosen.Test dataset is not used to tune the parmeter and instead bootstrapping is done in the training dataset.

```
#training multiple k-nearest neighbours models by tuning the k paramter (nr neighbours)
Model2_knn <- train(Outcome ~ .,
                      method = "knn",
                      data = train_set,
                      tuneGrid = expand.grid(k = 10:30))

plot(Model2_knn) #plotting accuracy vs various k values
```



```
#applying the model to the test dataset
y_hat <- predict(Model2_knn, newdata = test_set)
Model2_Accuracy<-confusionMatrix(y_hat, test_set$Outcome)$overall[["Accuracy"]]
Model2_F1<- confusionMatrix(y_hat, test_set$Outcome)$byClass[["F1"]]
Models_Accuracy <- rbind(Models_Accuracy,
                         c("Model 2: k-Nearest Neighbours",
                           Model2_Accuracy,
                           Model2_F1))
kable(Models_Accuracy)
```

Model	Accuracy	F1
Baseline	0.649350649350649	0.78740157480315
Model 1: Logistic Regression	0.753246753246753	0.822429906542056
Model 2: k-Nearest Neighbours	0.779220779220779	0.841121495327103

Model 3: Linear Discriminant Analysis (LDA) The third model takes all the predictors and uses linear discriminat analysis (LDA) to fit the model.

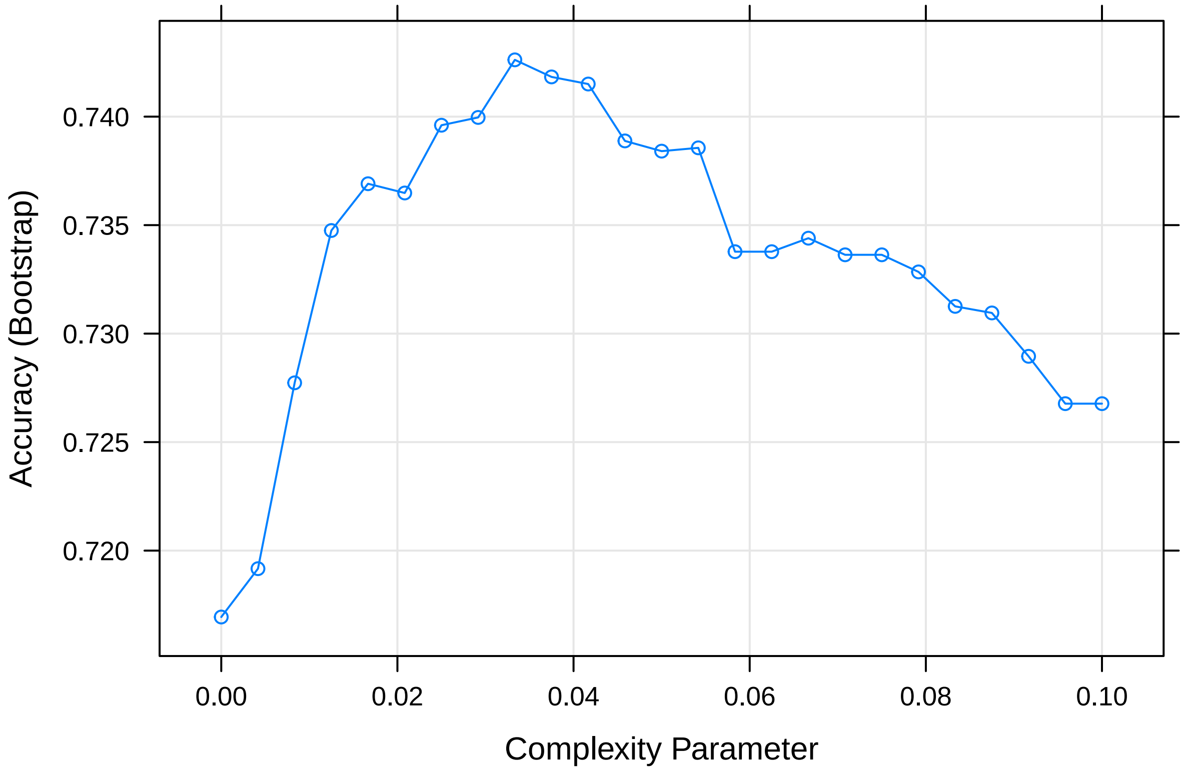
```
#training using LDA
Model3_lda <- train(Outcome ~ .,method = "lda", data = train_set)
#applying the model to the test dataset
y_hat <- predict(Model3_lda, newdata = test_set)
Model3_Accuracy<-confusionMatrix(y_hat, test_set$Outcome)$overall[["Accuracy"]]
```

```r
Model3_F1<- confusionMatrix(y_hat, test_set$Outcome)$byClass[["F1"]]
Models_Accuracy <- rbind(Models_Accuracy,
                         c("Model 3: Linear Discriminant Analysis (LDA)",
                           Model3_Accuracy,
                           Model3_F1))
kable(Models_Accuracy)
```

Model	Accuracy	F1
Baseline	0.649350649350649	0.78740157480315
Model 1: Logistic Regression	0.753246753246753	0.822429906542056
Model 2: k-Nearest Neighbours	0.779220779220779	0.841121495327103
Model 3: Linear Discriminant Analysis (LDA)	0.753246753246753	0.822429906542056

Model 4: Decision Tree The fourth model takes all the predictors and uses decision tree to fit the model. Multiple models are trained for various values of cp (complexity parameter) and the cp value from the model with highest accuracy is chosen.Test dataset is not used to tune the parmeter and instead bootstrapping is done in the training dataset.

```r
#Model 4: Decision Tree
#training multiple decision tree models by tuning the complexity paramter
Model4_rpart <- train(Outcome  ~ .,
                 method = "rpart",
                 data = train_set,
                 tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 25)))

#plotting accuracy vs various cp values
plot(Model4_rpart)
```



```r
#applying the model to the test dataset
y_hat <- predict(Model4_rpart, newdata = test_set)
Model4_Accuracy<-confusionMatrix(y_hat, test_set$Outcome)$overall[["Accuracy"]]
Model4_F1<- confusionMatrix(y_hat, test_set$Outcome)$byClass[["F1"]]
Models_Accuracy <- rbind(Models_Accuracy,
                         c("Model 4: Decision Tree",
                           Model4_Accuracy,
                           Model4_F1))
kable(Models_Accuracy)
```

Model	Accuracy	F1
Baseline	0.649350649350649	0.78740157480315
Model 1: Logistic Regression	0.753246753246753	0.822429906542056
Model 2: k-Nearest Neighbours	0.779220779220779	0.841121495327103
Model 3: Linear Discriminant Analysis (LDA)	0.753246753246753	0.822429906542056
Model 4: Decision Tree	0.74025974025974	0.821428571428571

Results

Based on the accuracy and F1 score of all the models, we see that the model fit using k-nearest neighbours results in the best accuracy and F1 score.

We see that the predictor glucose has the highest influence on predicting the outcome. After glucose, the predictors with highest influence are age, BMI, insulin, skin thickness, pregnancies and diabetes pedigree function respectively. Blood pressure does not seem to have any influence.

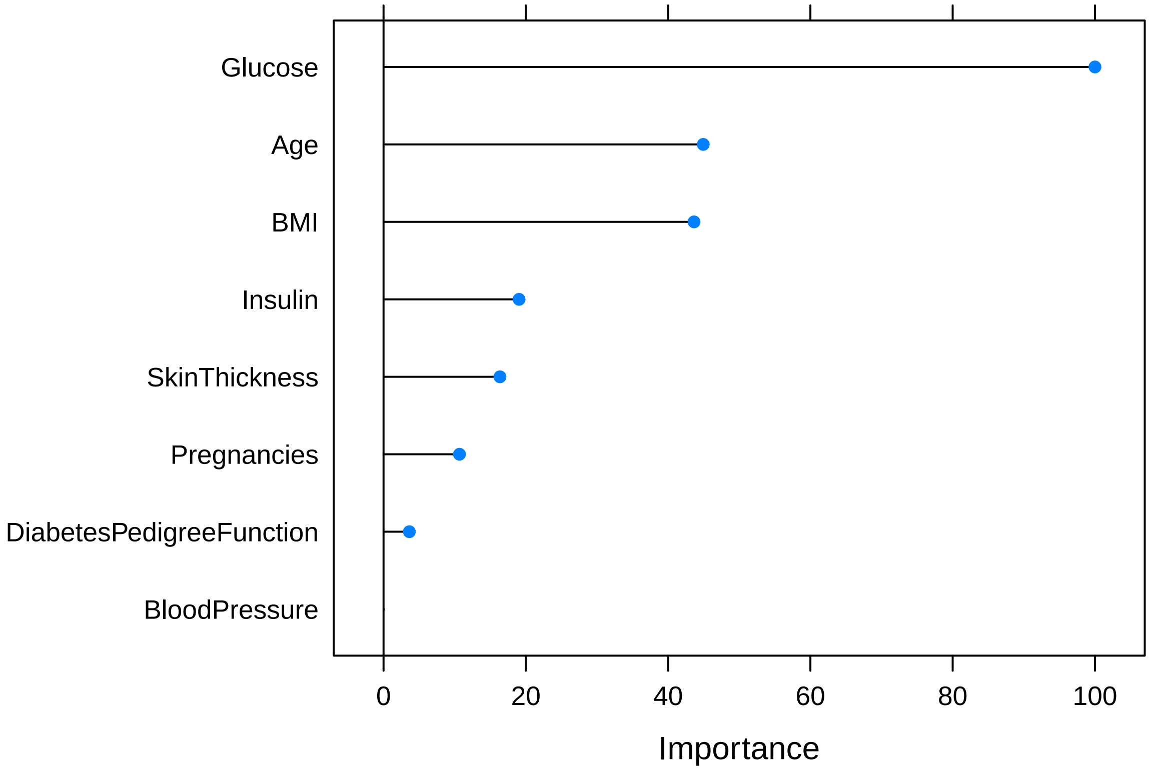
```r
#Final Model
Model2_Accuracy
```

```
## [1] 0.7792208
```

```r
Model2_F1
```

```
## [1] 0.8411215
```

```r
#Variable Importance
plot(varImp(Model2_knn))
```



Conclusion

The final model chosen is a model fit using k-nearest neighbours algorithm with pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function and age as predictors. The parameter for number of neighbours (k) was tuned using bootstrapping. This model was chosen as it had the highest accuracy and F1 score.The accuracy was 0.7792208 and F1 score was 0.8411215

The impact of the model is that it can be used to predict the diabetes onset of Pima Indian females.

The limitation of this model is that it only predicts diabetes onset for a specific group of people (females above 21 with Pima Indian heritage). It can be expanded to a wider group of people by using the original dataset and having more predictors like gender. The number of sample size would also be larger in the original dataset, which could result in a higher accuracy and F1 score.