Pima Indian Diabetes Prediction

Alex Rivera Cruz

19/11/2020

Overview

This project is related to the Prediction Diabetes Project for the Harvardx Capstone course of Data Science. The present report start with a general idea of the project and by representing its objective.

Then the given dataset will be prepared and setup. An exploratory data analysis is carried out in order to develop a machine learning algorithm that could predict if someone has Diabetes until a final model. Results will be explained. Finally the report ends with some concluding remarks.

Introduction

Diabetes affects an estimated 30.3 million people in the United States and is the seventh leading cause of death. Diabetes can affect many parts of the body and is associated with serious complications, such as heart disease and stroke, blindness, kidney failure, and lower limb amputation.

To limit the rates of this disease, prevention on a primary and secondary level is preferred to life-long treatment. In order to do this, healthcare providers must have robust methods of predicting which patients would likely have the disease. To anticipate risk of diabetes, development of a robust model is necessary.

This study is focused on analysis of the Diabetes dataset downloaded to a personal computer from Kaggle at: https://www.kaggle.com/uciml/pima-indians-diabetes-database The dataset relates frequencies and statistics of physiological measurements on women over the age of 21 belonging to the Pima Native American tribe residing of Arizona.

Aim of the project

The goal of this project is to develop a machine learning algorithm that can predict incidence of Diabetes in a population of Pima Indians. Four different regression approaches will be attempted to yield an optimised accuracy with sensitivity and specificity measures.

Dataset

We require the next packages to complete the project.

The dataset will be downloaded from Kaggle and stored on a personal computer, where it will be pushed into R.

```r
#Packages required
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(tibble)) install.packages("tibble", repos = "http://cran.us.r-project.org")
if(!require(plyr)) install.packages("plyr", repos = "http://cran.us.r-project.org")
if(!require(ggrepel)) install.packages("ggrepel", repos = "http://cran.us.r-project.org")
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")
if(!require(readr)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("xgboost", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(tibble)
library(readr)
library(plyr)
library(ggrepel)
library(gridExtra)
library(readr)
library(corrplot)
library(caret)
library(xgboost)
library(e1071)
library(randomForest)
#Download the dataset from an excel sheet on computer
file.exists("E:\\Courses\\Data Science\\Lessons\\9. Capstone Project All Learners\\9 Capstone Project Al
```

```
## [1] TRUE
```

```r
pima<- read_csv("E:\\Courses\\Data Science\\Lessons\\9. Capstone Project All Learners\\9 Capstone Proje
```

Methods and Analysis

Data Analysis

After upload, we will study the structure of the `pima` dataset

```
#Check the first 6 rows of the provided dataset
head(pima)%>%
  print.data.frame()
```

```
##   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
```

`pima` appears to be in tidy format, meaning that each variable forms a column, and each row represents a observation, and the observational unit forms a table. Next, the parameters within the dataset will be defined.

```
#See the overall structure of the dataset
str(pima)
```

```
## tibble [768 x 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()
##   .. )
```

It appears that `pima` has 768 observations of 9 variables, all of which are in numeric format.

-`Pregnancies` describes the gravidity of each patient in the dataset.

-`Glucose` describes the mg of glucose per every dL of blood in the patient.

-`BloodPressure` describes the diastolic blood pressure of each patient.

-`SkinThickness` measures the epidermal, dermal, and subcutaneous layers of brachial skin for each patient.

-`Insulin` describes the mIU of insulin protein per litre of blood after 2 hours of fasting.

-`BMI` describes the standardised body mass index of each patient.

-`DiabetesPedigreeFunction` describes the result of an unlisted function that calculates the genetic influence of Diabates in each patient.

-`Age` describes the rounded age in years of each patient.

-`Outcome` denotes `0` for non-diabetic and `1` for diabetic for each patient.

Next, missing values are quite common in real-life datasets, so it is imperative to inspect the set for this.

```
#Find the proportion of the dataset that is NA
sum(is.na(pima))
```

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

```
sum(is.na(pima))/(ncol(pima)*nrow(pima))
```

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

We confirm that there are no missing values in the `pima` dataset.

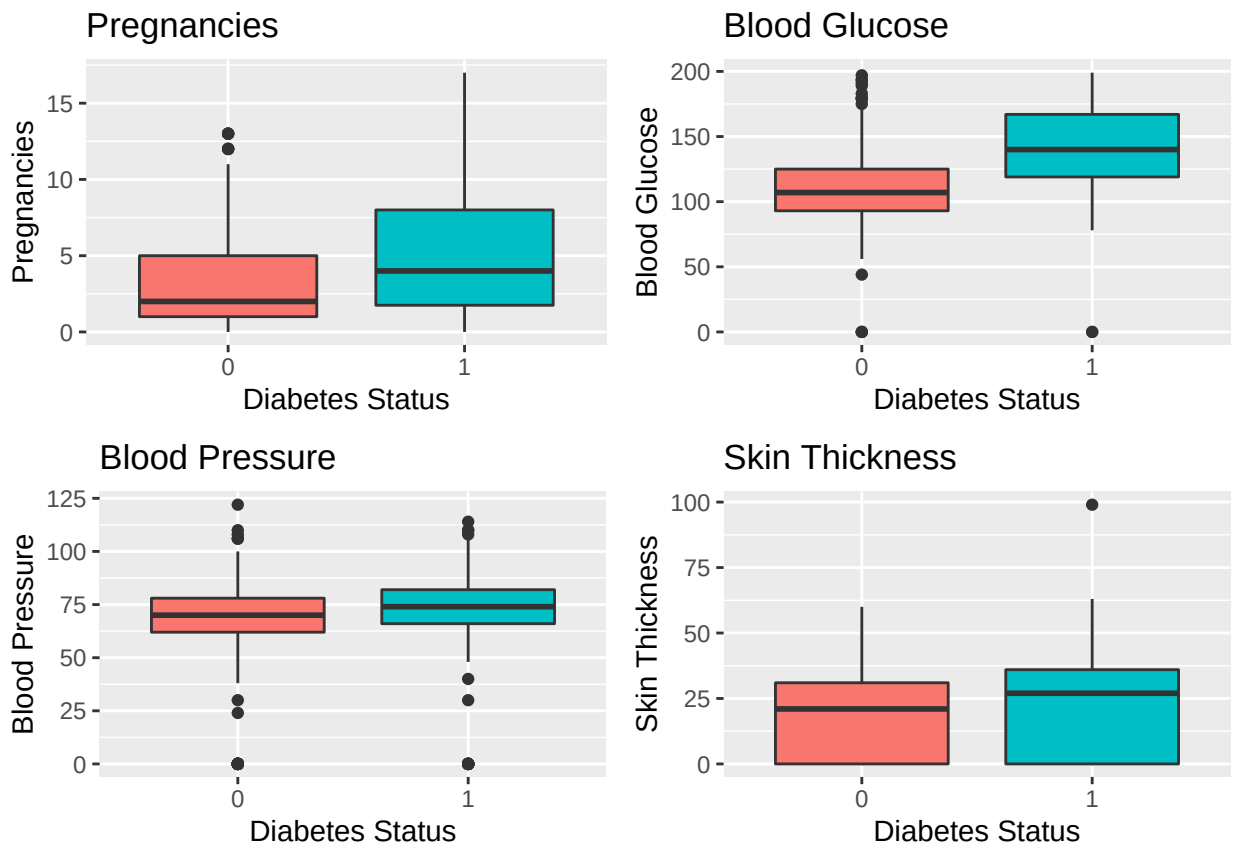
Now that the data has been cleaned up, it is ready for some visualisation analysis.

Visualisation

Visualisation is a key step before modelling. Generating plots enables the data scientist to have an overall understand of trends in the data, faciliating the analysis.

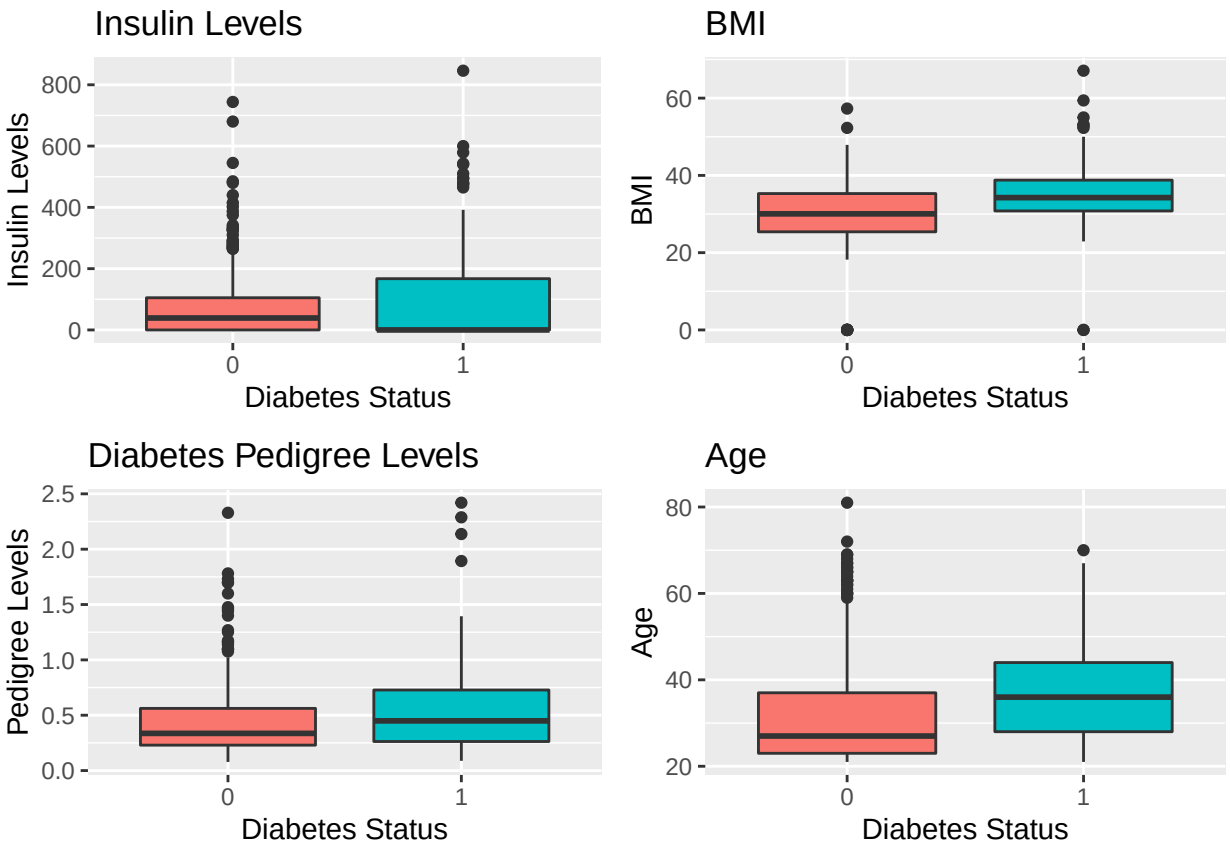
There are 8 parameters that could be possibly correlated with Diabetes in the dataset. In this section, each prospective parameter will be stratified by Diabates status to observe differences in the distribution.

```
#Convert outcome to binary factor class#
pima$Outcome <- as.factor(pima$Outcome)
#Box and whisker plot for pregnancies stratified by Diabates status
p1<- pima %>% group_by(Outcome) %>% ggplot(aes(y=Pregnancies,x=Outcome)) + geom_boxplot(aes(fill=Outcome
  labs(title="Pregnancies", x= "Diabetes Status", y = "Pregnancies") + theme(legend.position = "none")
#Box and whisker plot for blood glucose stratified by Diabates status
p2<- pima %>% group_by(Outcome) %>% ggplot(aes(y=Glucose,x=Outcome)) + geom_boxplot(aes(fill=Outcome)) +
  labs(title="Blood Glucose", x= "Diabetes Status", y = "Blood Glucose")  + theme(legend.position = "no
#Box and whisker plot for BP stratified by Diabates status
p3<- pima %>% group_by(Outcome) %>% ggplot(aes(y=BloodPressure,x=Outcome)) + geom_boxplot(aes(fill=Outc
  labs(title="Blood Pressure", x= "Diabetes Status", y = "Blood Pressure")  + theme(legend.position = "
#Box and whisker plot for skin thickness stratified by Diabates status
p4<- pima %>% group_by(Outcome) %>% ggplot(aes(y=SkinThickness,x=Outcome)) + geom_boxplot(aes(fill=Outc
  labs(title="Skin Thickness", x= "Diabetes Status", y = "Skin Thickness")  + theme(legend.position = "
#arrange the plots in a 2x2
grid.arrange(p1,p2,p3,p4,ncol=2)
```



From looking at the distribution of these parameters stratified by Diabates status, it appears that women who have had more pregnancies have a higher mean incidence of Diabates compared to women with fewer pregnancies. Additionally, the mean plasma glucose levels in those with Diabates is higher than in healthy patients. These two parameters may prove to be useful predictors in the models. Blood pressure and skin thickness don't appear to have a distinct difference across Diabates status.

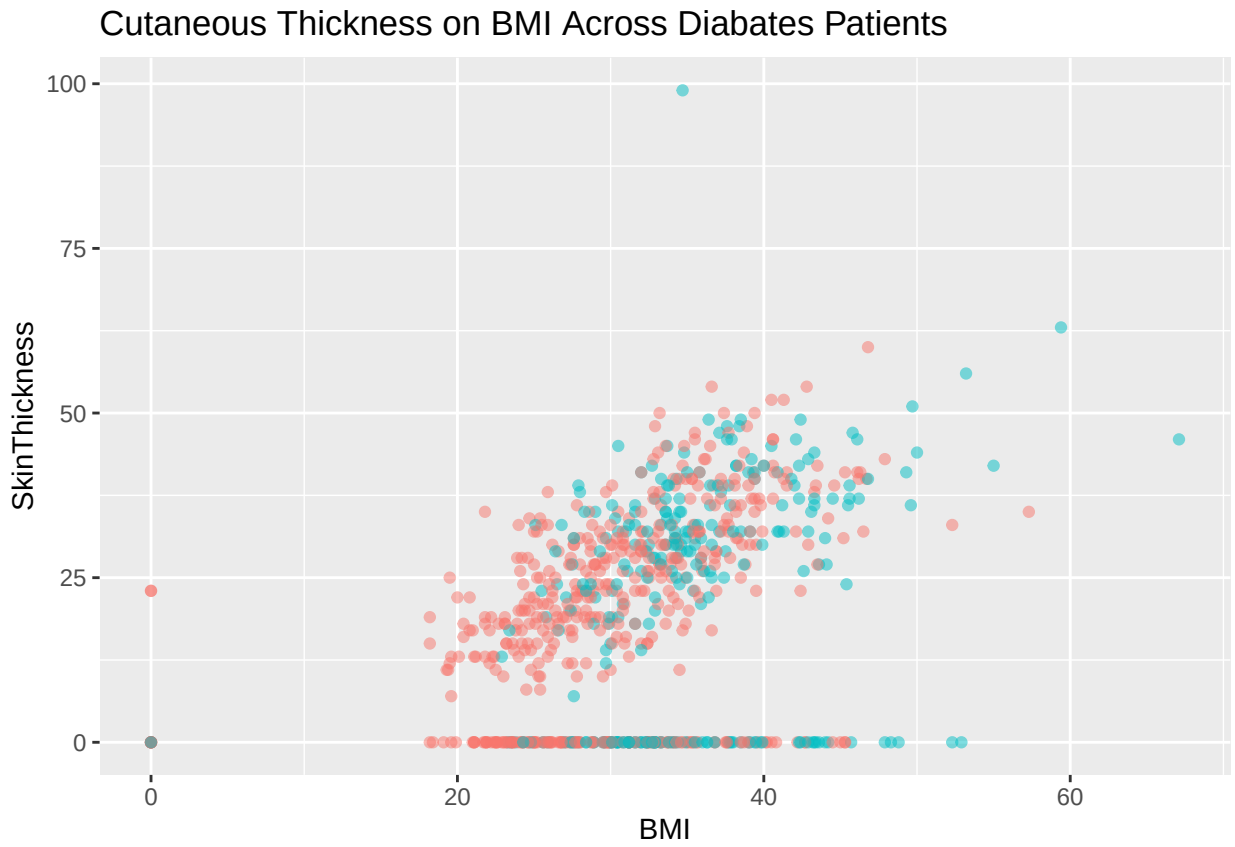
```
#Box and whisker plot for insulin levels stratified by Diabates status
p5<- pima %>% group_by(Outcome) %>% ggplot(aes(y=Insulin,x=Outcome)) + geom_boxplot(aes(fill=Outcome)) +
  labs(title="Insulin Levels", x= "Diabetes Status", y = "Insulin Levels")  + theme(legend.position = "n
#Box and whisker plot for BMI stratified by Diabates status
p6<- pima %>% group_by(Outcome) %>% ggplot(aes(y=BMI,x=Outcome)) + geom_boxplot(aes(fill=Outcome)) +
  labs(title="BMI", x= "Diabetes Status", y = "BMI")  + theme(legend.position = "none")
#Box and whisker plot for pedigree levels stratified by Diabates status
p7<- pima %>% group_by(Outcome) %>% ggplot(aes(y=DiabetesPedigreeFunction,x=Outcome)) + geom_boxplot(aes
  labs(title="Diabetes Pedigree Levels", x= "Diabetes Status", y = "Pedigree Levels")  + theme(legend.p
#Box and whisker plot for ages stratified by Diabates status
p8<- pima %>% group_by(Outcome) %>% ggplot(aes(y=Age,x=Outcome)) + geom_boxplot(aes(fill=Outcome)) +
  labs(title="Age", x= "Diabetes Status", y = "Age")  + theme(legend.position = "none")
#arrange the plots in a 2x2#
grid.arrange(p5,p6,p7,p8,ncol=2)
```



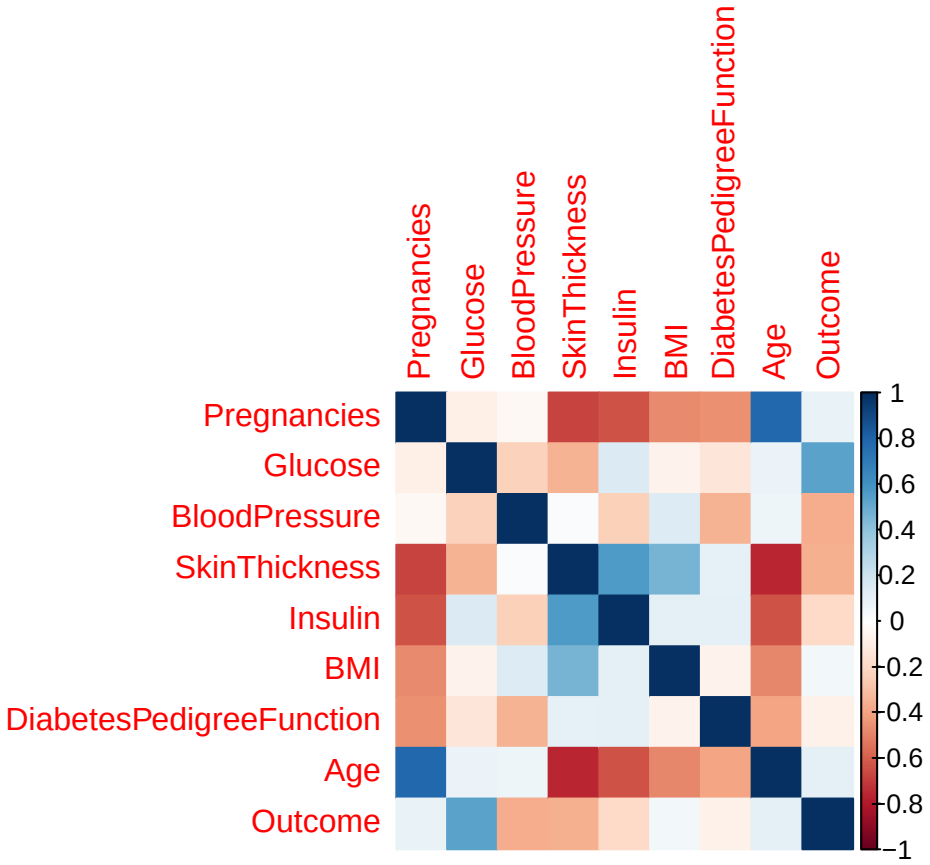
In this last charts, BMI and and age more clearly have a relationship that can be seen across the diabetes status, but insulin levels and diabetes pedigree don't appear to have a distinct difference. However, all the parameters analysed will have to initially be put into the model to see which variables are statistically significant in their relationship with Diabates status.

Now there may be high potential for confounding variables in this dataset. For example, the parameter `SkinThickness` is a measure of fat accumulation while BMI is also a common measure for load of the body. If these two variables have a high absolute correlation (>0.5), then they should be excluded from the analysis. To confirm that hypothesis we have to see this plot:

```
#scatterplot for Skin Thickness on BMI across Diabates status
pima %>% group_by(Outcome) %>% ggplot() + geom_point(aes(x=BMI, y=SkinThickness, colour=Outcome), alpha=
  theme(legend.position = "none") + ggtitle("Cutaneous Thickness on BMI Across Diabates Patients")
```




```
#correlation plot between all variables in 'pima'
par( mfrow = c(1,1) )
pima$Outcome<- as.numeric(pima$Outcome)
cor1 <- cor(pima, method = c("pearson"))
cor2 <- round(cor(cor1),2)
corrplot::corrplot(cor2, method = "color")
```



It appears that although the variables do have a correlation, none of them have an absolute value above 0.5, making all of them suitable for the models.

Modelling Approach

Now, we have to split the data into training and test sets. The training set will be used to develop the models, and the test set will be to evaluate the accuracy, sensitive and specificity of each model. The test set will be represented by the 20% of the dataset, as the dataset has less than 800 observations. This split will create a strong balance between parameter estimates and performance statistics.

```
#Attempt to convert the variables to class of factor
pima[sapply(pima, is.character)] <- lapply(pima[sapply(pima, is.character)], as.factor)
#Split both datasets into an 80:20 split using a seed
set.seed(1, sample.kind="Rounding")
sample_size<- floor(0.2*nrow(pima))
test_index <- sample(seq_len(nrow(pima)), size=sample_size)
pima_train<- pima[-test_index,]
pima_test<- pima[test_index,]
```

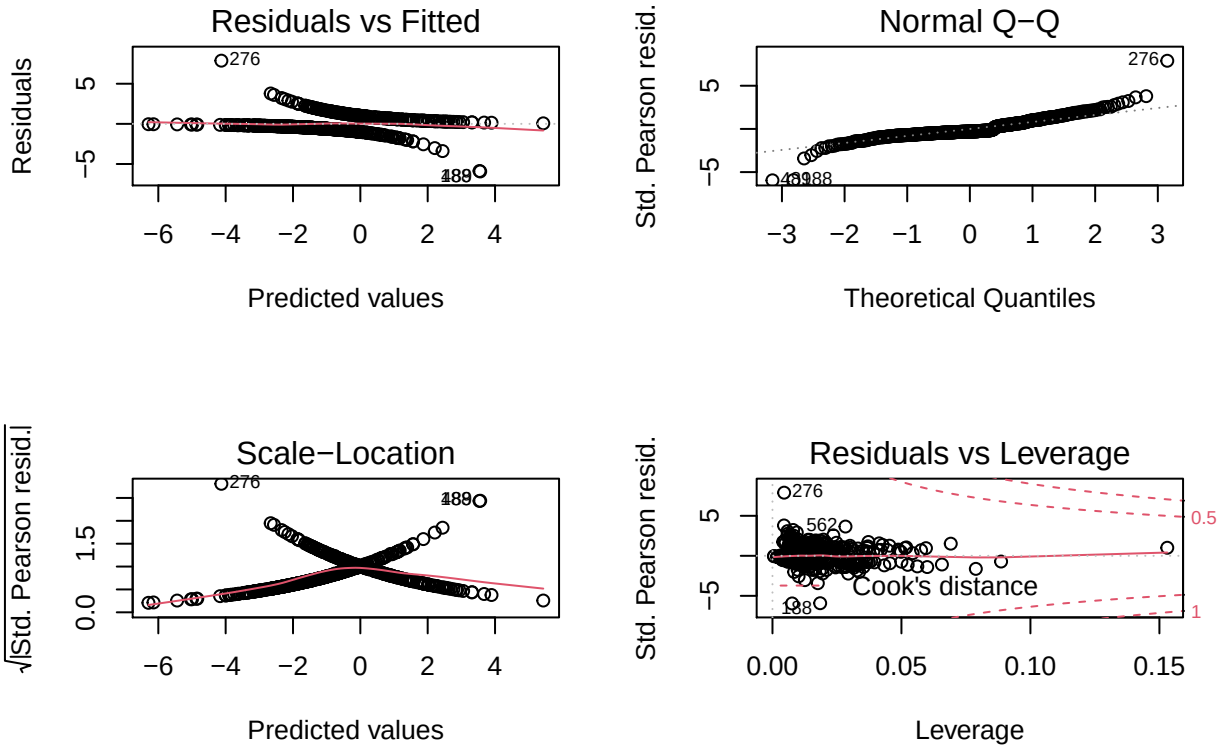
I. Logistic Regression

Logistic regressions are models similar to linear regressions, but they are specialised for discrete rather than continuous outcomes. This makes them a perfect starting point for this study of classification algorithms, namely the binomial outcomes "Diabetic" or "non-Diabetic."

```r
#Set the plots to a 2x2
par(mfrow = c(2,2))
#Calculate the deviance residuals, coefficients, and significances
pima_train$Outcome<- as.factor(pima_train$Outcome)
mod_glm <- glm(Outcome ~ ., data = pima_train, family = binomial(link = "logit"))
summary(mod_glm)
```

```
##
## Call:
## glm(formula = Outcome ~ ., family = binomial(link = "logit"),
##     data = pima_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6787  -0.7260  -0.4227   0.7112   2.8785
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -8.6134123  0.8176411 -10.534  < 2e-16 ***
## Pregnancies               0.1501737  0.0354522   4.236 2.28e-05 ***
## Glucose                   0.0347289  0.0042125   8.244  < 2e-16 ***
## BloodPressure            -0.0107631  0.0059293  -1.815 0.069489 .
## SkinThickness             0.0003224  0.0076690   0.042 0.966468
## Insulin                  -0.0013166  0.0010076  -1.307 0.191297
## BMI                       0.0945281  0.0172164   5.491 4.01e-08 ***
## DiabetesPedigreeFunction  1.1605340  0.3428217   3.385 0.000711 ***
## Age                       0.0083561  0.0103591   0.807 0.419871
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 797.28  on 614  degrees of freedom
## Residual deviance: 577.59  on 606  degrees of freedom
## AIC: 595.59
##
## Number of Fisher Scoring iterations: 5
```

```r
plot(mod_glm)
```



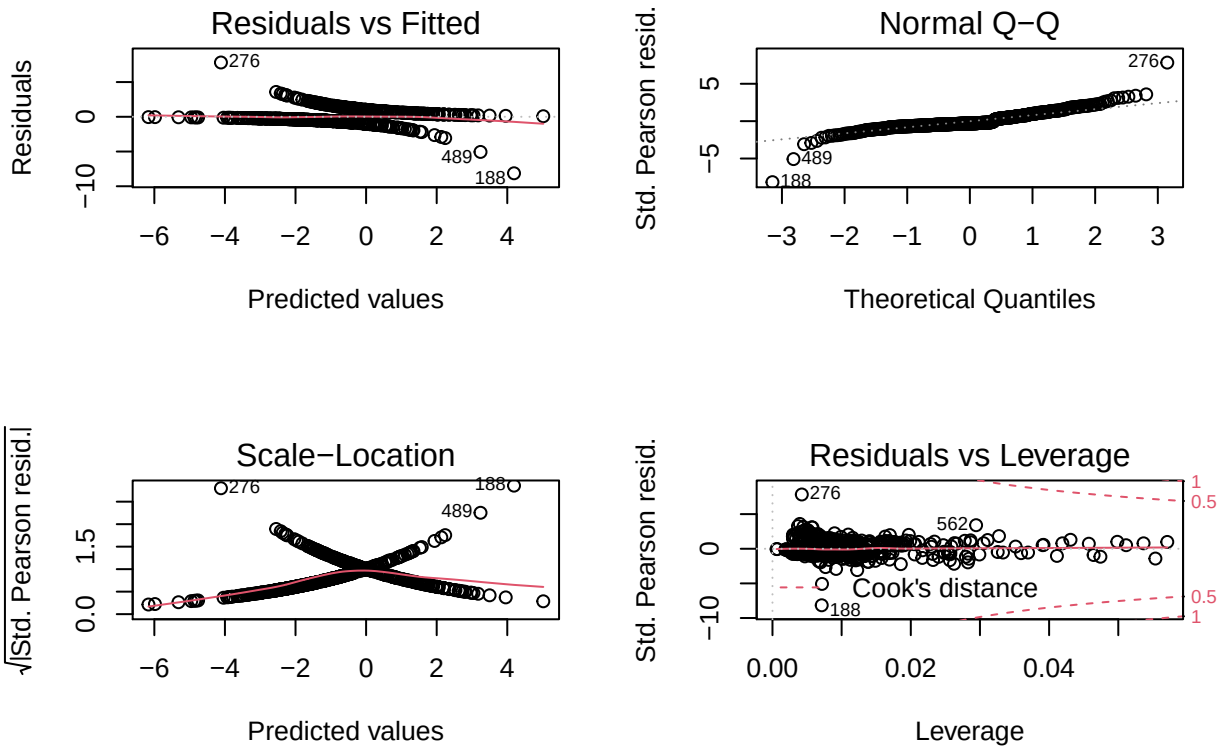
Another reason why the general logistic regression is much appreciated is because it shows which variables are significant in their relationship to Diabates outcome. From the table and plots above, it appears that regression intercept, `Pregnancies`, `Glucose`, `BMI`, and `DiabetesPedigreeFunction` are statistically significant to an alpha of almost 0. The variable `BloodPressure` is significant at the alpha=0.05. All of these parameters will be selected to continue to the optimised regression.

```
#Optimised logistic regression with significant parameters
par(mfrow = c(2,2))
mod_glm2 <- glm(Outcome~ Pregnancies+Glucose+BloodPressure+BMI + DiabetesPedigreeFunction,
                data=pima_train, family=binomial(link= "logit"))
summary(mod_glm2)
```

```
##
## Call:
## glm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
##     BMI + DiabetesPedigreeFunction, family = binomial(link = "logit"),
##     data = pima_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9005  -0.7323  -0.4186   0.7021   2.8735
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -8.24126    0.77246 -10.669  < 2e-16 ***
## Pregnancies                0.16883    0.03088   5.467 4.59e-08 ***
```

```
## Glucose                   0.03348    0.00383   8.742  < 2e-16 ***
## BloodPressure            -0.01004    0.00575  -1.747  0.08068 .
## BMI                       0.09050    0.01611   5.617 1.95e-08 ***
## DiabetesPedigreeFunction  1.09233    0.33688   3.242  0.00119 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 797.28  on 614  degrees of freedom
## Residual deviance: 580.52  on 609  degrees of freedom
## AIC: 592.52
##
## Number of Fisher Scoring iterations: 5
```

```
plot(mod_glm2)
```



In order to predict a binary outcome, bounds must be set. The regression continues in order to create a confusion matrix.

```
pima_test$Outcome<- as.factor(pima_test$Outcome)
pred_glm <- predict(mod_glm2,pima_test, type = "response")
pred_glm <- ifelse(pred_glm <= 0.5, 1, 2)
pred_glm<- as.factor(pred_glm)
cm_glm<- confusionMatrix(pred_glm,pima_test$Outcome)
cm_glm
```

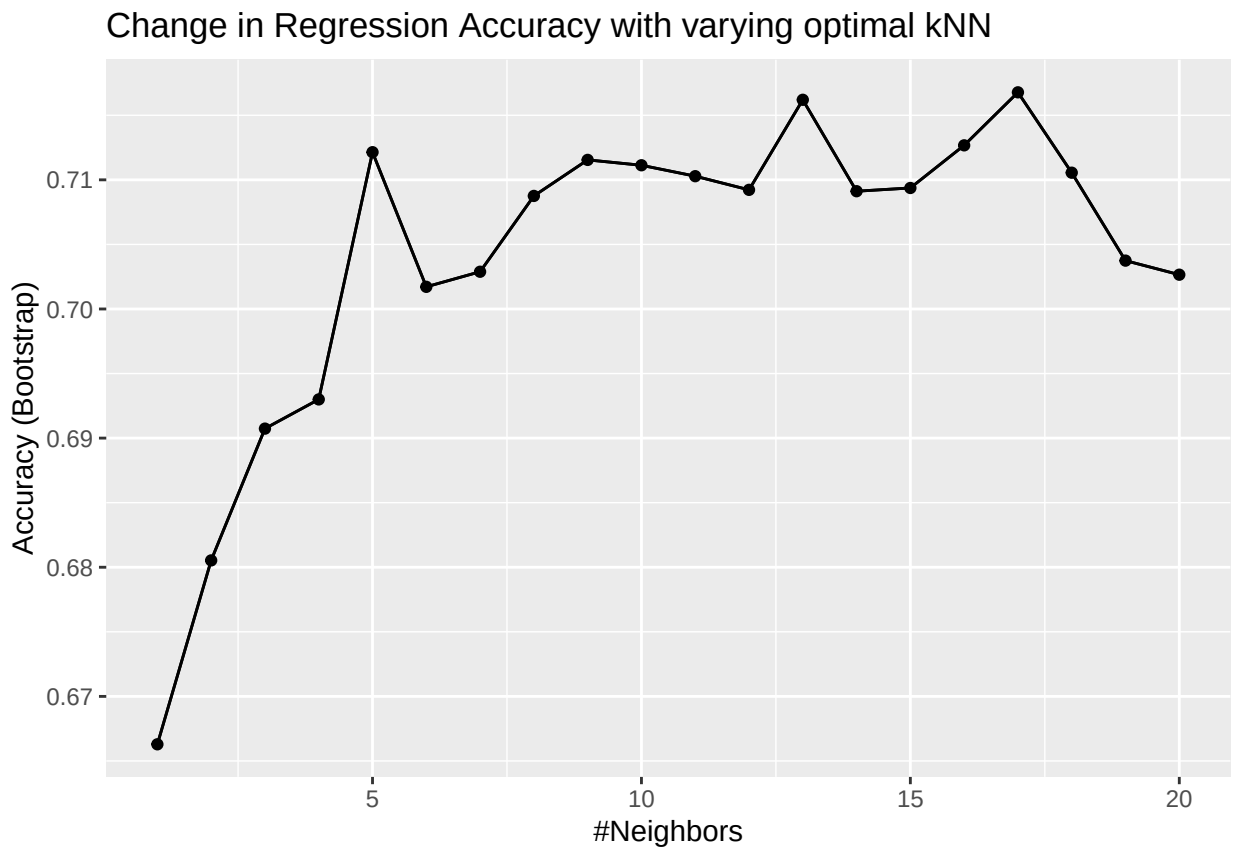
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 88 16
##          2 13 36
##
##                Accuracy : 0.8105
##                  95% CI : (0.7393, 0.8692)
##     No Information Rate : 0.6601
##     P-Value [Acc > NIR] : 2.948e-05
##
##                   Kappa : 0.5716
##
##  Mcnemar's Test P-Value : 0.7103
##
##             Sensitivity : 0.8713
##             Specificity : 0.6923
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.7347
##              Prevalence : 0.6601
##          Detection Rate : 0.5752
##    Detection Prevalence : 0.6797
##       Balanced Accuracy : 0.7818
##
##        'Positive' Class : 1
##
```

The confusion matrix generated with the general logistic regression has a balanced accuracy of 0.7818, which is actually quite impressive for an initial trial. The sensitivity of the model is high at 0.8713. However, the specificity has lower value than desired, it is 0.6923.

II. k-Nearest Neighbors

The next model to be attempted will be the k-Nearest Neighbours algorithm. This model operates on the principle that test datapoints are similar to their "neighbouring" datapoints. These `k` values are used to develop the prediction. Compared to the general logistic regression, the kNN approach is a non-parametric model that is more supportive of non-linear models.

```
#Develop a model to test the accuracy on the number of neighbours
mod_knn <- train(Outcome ~ ., data= pima_test, method = "knn", tuneGrid = data.frame(k = seq(1,20,1)))
mod_knn %>% ggplot()+geom_line(aes(x=k, y=Accuracy)) +
  labs(title= "Change in Regression Accuracy with varying optimal kNN")
```



According to the graph, the optimal number of neighbours for the model to search for is k=17.

```
mod_knn$bestTune
```

```
##     k
## 17 17
```

This value will be used in the prediction for the test dataset.

```
pred_knn <- predict(mod_knn, pima_test, type="raw")
cm_knn<- confusionMatrix(pred_knn,pima_test$Outcome)
cm_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 93 26
##          2  8 26
##
##                Accuracy : 0.7778
##                  95% CI : (0.7036, 0.8409)
##     No Information Rate : 0.6601
##     P-Value [Acc > NIR] : 0.001032
##
```

```
##                  Kappa : 0.4594
##
##  Mcnemar's Test P-Value : 0.003551
##
##             Sensitivity : 0.9208
##             Specificity : 0.5000
##          Pos Pred Value : 0.7815
##          Neg Pred Value : 0.7647
##              Prevalence : 0.6601
##          Detection Rate : 0.6078
##    Detection Prevalence : 0.7778
##       Balanced Accuracy : 0.7104
##
##        'Positive' Class : 1
##
```

The kNN regression show us that we have a lower balanced accuracy compared to the logistic regression model, at 0.7104. Its sensitivity improved at 0.9208. However, the specificity is quite low at 0.5.

III. Random Forest

This type of regression is a little more advanced, based on the concept of creating several "decision trees". This involves a branching equation to derive the output, making random forest models theoretically quite accurate, but very time-intensive.

```
#Develop a random forest regression model to produce a confusion matrix#
mod_rf <- train(Outcome ~ ., method = "rf", data = pima_train)
pred_rf <- predict(mod_rf, pima_test)
pima_test$Outcome<- as.factor(pima_test$Outcome)
cm_rf<- confusionMatrix(pred_rf,pima_test$Outcome)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 82 18
##          2 19 34
##
##                Accuracy : 0.7582
##                  95% CI : (0.6824, 0.8237)
##     No Information Rate : 0.6601
##     P-Value [Acc > NIR] : 0.005655
##
##                   Kappa : 0.4636
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.8119
##             Specificity : 0.6538
##          Pos Pred Value : 0.8200
##          Neg Pred Value : 0.6415
##              Prevalence : 0.6601
```

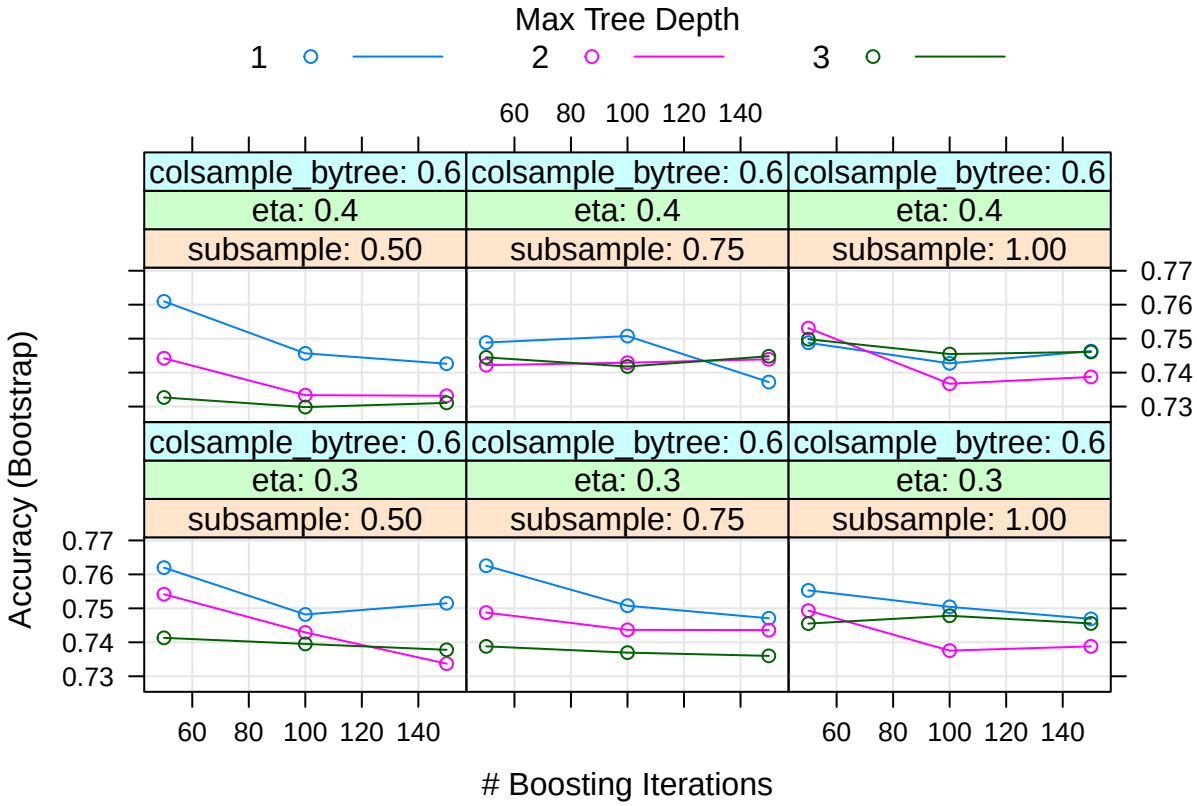
```
##          Detection Rate : 0.5359
##    Detection Prevalence : 0.6536
##       Balanced Accuracy : 0.7329
##
##        'Positive' Class : 1
##
```

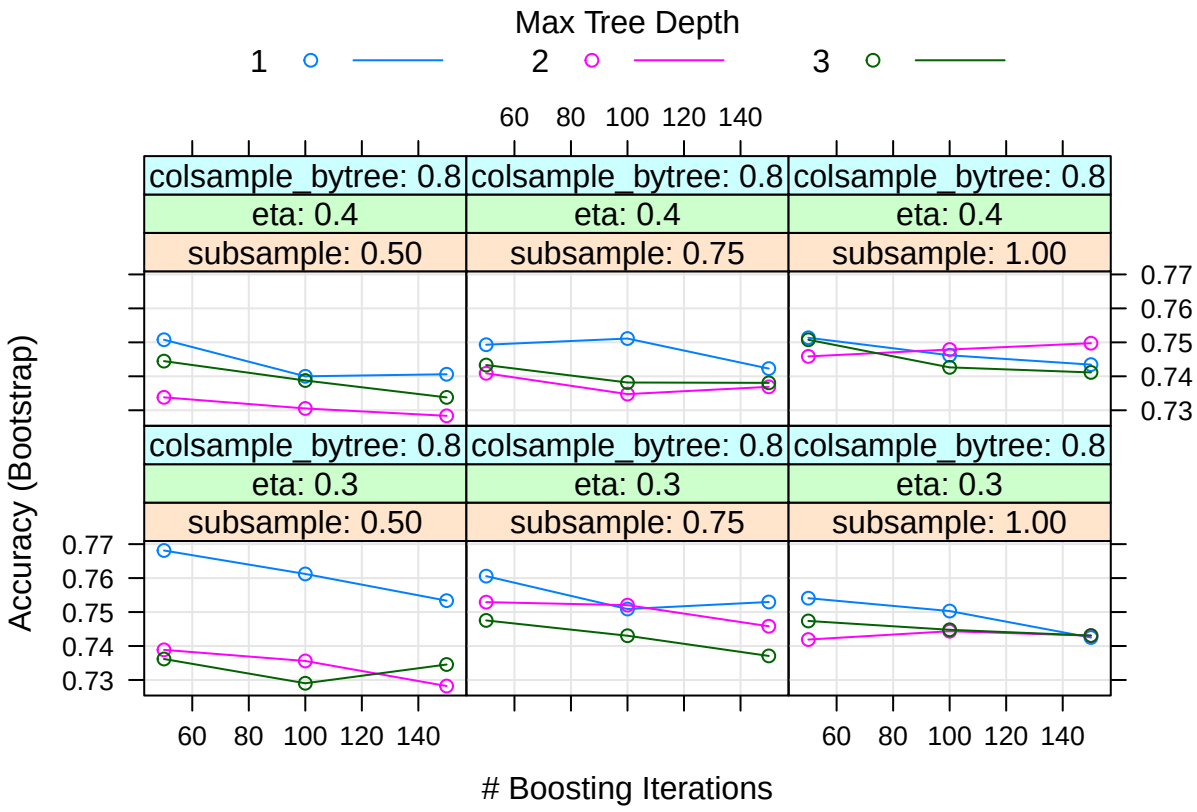
The random forest model shows us that the balanced accuracy is 0.7328, the sensitivity is 0.8119, and the specificity is low at 0.6538. Its values, along with the kNN values, are inferior to the initial logistic regression.

IV. XGBoost

Finally, the novel eXtreme Gradient Boosting software algorithm will be analysed. This machine-learning approach has gained several awards and traction on the web, and is the algorithm of choice for data science competitions on Kaggle. Its parallel tree boosting method is incredibly flexible and more time-efficient than the Random Forest algorithm.

```
# Develop an XGB regression model to produce a confusion matrix#
par(mfrow = c(2,1))
mod_xgb <- train(Outcome ~ ., method = "xgbTree", data = pima_test)
plot(mod_xgb)
```






```
pred_xgb <- predict(mod_xgb, pima_test)
cm_xgb<- confusionMatrix(pred_xgb,pima_test$Outcome)
cm_xgb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 95 14
##          2  6 38
##
##                Accuracy : 0.8693
##                  95% CI : (0.8054, 0.9183)
##     No Information Rate : 0.6601
##     P-Value [Acc > NIR] : 3.355e-09
##
##                   Kappa : 0.6974
##
##  Mcnemar's Test P-Value : 0.1175
##
##             Sensitivity : 0.9406
##             Specificity : 0.7308
##          Pos Pred Value : 0.8716
##          Neg Pred Value : 0.8636
##              Prevalence : 0.6601
```

```
##          Detection Rate : 0.6209
##    Detection Prevalence : 0.7124
##       Balanced Accuracy : 0.8357
##
##        'Positive' Class : 1
##
```

It appears that the XGBoost model was the superior approach compared to the three previously tested model approaches. Its balanced accuracy is impressive at 0.8356, and its sensitivity is very high at 0.9405. Specificity has been shown to be lower in this modelling study, but this approach had the highest value at 0.7308.

Results and Discussion

```
# Create a knitr table of the regression approaches and their values#
cm<- data.frame(c("Logistic Model", "K Nearest Neighbors", "Random Forest", "XGBoost"), c(cm_glm$byClass
                cm_knn$byClass[1], cm_rf$byClass[1], cm_xgb$byClass[1]), c(cm_glm$byClass[2], cm_knn$by
                cm_rf$byClass[2], cm_xgb$byClass[2]), c(cm_glm$byClass[11], cm_knn$byClass[11], cm_rf$b
                cm_xgb$byClass[11]))
cm<- as_tibble(cm)
colnames(cm) <- c("Model", "Sensitivity", "Specificity", "Balanced Accuracy")
cm %>% knitr::kable()
```

Model	Sensitivity	Specificity	Balanced Accuracy
Logistic Model	0.8712871	0.6923077	0.7817974
K Nearest Neighbors	0.9207921	0.5000000	0.7103960
Random Forest	0.8118812	0.6538462	0.7328637
XGBoost	0.9405941	0.7307692	0.8356816

The analysis of the four regression approaches shows that the XGBoost was the most superior, with the highest sensitivity, highest specificity, and balanced accuracy. The logistic regression model has the next highest balanced accuracy, followed by the k Nearest Neighbors and the Random Forest Approaches.

Conclusion

Machine learning algorithm was built to predict Diabetes status with Pima Indians Diabetes dataset.

The XGBoost model is characterized by the higher accuracy value and is hence the optimal model to use for the present project.

For further improvement in the analysis of this study, it would be recommended to increase the diversity of algorithms tested. The caret package that was needed in this study holds a plethora of different approaches that may be suitable for this investigation.