Harvardx Capstone Diabetes Prediction Project

Apurba Das

5/25/2020

Contents

1 Introduction

This project is the a part of Harvardx's Data Science course and serves as the Choose Your Own Capstone project. This section describes the dataset and variables, and summarizes the goal of the project and key steps that were performed to achieve it.

1.1 Overview

The aim of this project is to create a classification or prediction system based on several independent diagnostic and medical factors that predicts if a patient has diabetes or not using the Pima Indians Diabetes dataset. As defined by the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK), diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high. Blood glucose is your main source of energy and comes from the food you eat. Insulin, a hormone made by the pancreas, helps glucose from food get into your cells to be used for energy. Of all the types of diabetes, the most common types are type 1, type 2, and gestational diabetes. This project contains the different methods used to create several models to predict diabetes.

1.2 Procedure

Firstly, the Diabetes data is downloaded and datasets are created. Then, we develop algorithms using the training data set. For a final test of these algorithms, prediction of whether a patient is suffering from diabetes or not is done in the testing (acts as validation) set as if they were unknown. Confusion Matrix and primarily, sensitivity and accuracy will be used to evaluate how close our predictions are to the true values in the validation set.

As we train multiple machine learning algorithms using the inputs in one subset to predict if a patient is suffering from diabetes in the validation set, we also analyze and understand the dataset along the way.

The final goal is to come up with multiple machine learning algorithms and understand the advanced techniques used in them while building the models.

1.3 Dataset and variables

The dataset used is the Pima Indians Diabetes dataset which is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. With this dataset which a subset of a much larger database, several constraints were placed on the selection of these instances. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Overall, this data set consists of 768 observations of 9 variables: 8 variables which will be used as model predictors (number of times pregnant, plasma glucose concentration, diastolic blood pressure (mm Hg), triceps skin fold thickness (in mm), 2-hr serum insulin measure, body mass index, a diabetes pedigree function, and age) and 1 outcome variable (whether or not the patient has diabetes). This diabetes dataset is automatically downloaded in our code from our GitHub repo.

```
##################################
# Load data and required packages
##################################

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ------------------

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts -- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

## Warning: package 'data.table' was built under R version 3.6.3

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart.plot

## Warning: package 'rpart.plot' was built under R version 3.6.3

## Loading required package: rpart
```

```
if(!require(gbm)) install.packages("gbm", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.6.3
```

```
## Loaded gbm 2.1.5
```

```
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(devtools)) install.packages("devtools", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: devtools
```

```
## Warning: package 'devtools' was built under R version 3.6.3
```

```
## Loading required package: usethis
```

```
## Warning: package 'usethis' was built under R version 3.6.3
```

```
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(ggcorrplot)) install.packages("ggcorrplot", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: ggcorrplot
```

```
## Warning: package 'ggcorrplot' was built under R version 3.6.3
```

```
library(devtools)

# Load the Diabetes data set from my github account
diabetes_data <- read.table("https://raw.githubusercontent.com/apurba-das/Diabletes-Prediction/master/p
                            header = FALSE,
                            sep = ",")

colnames(diabetes_data) <- c("Pregnancies",
                             "Glucose",
                             "BloodPressure",
                             "SkinThickness",
                             "Insulin",
                             "BMI",
                             "DiabetesPedigreeFunction",
                             "Age",
                             "Outcome")

diabetes_data$Outcome <- ifelse(diabetes_data$Outcome == "1", "Diabetes",
                            ifelse(diabetes_data$Outcome == "0", "NoDiabetes", NA))
```

2 Methods/ Analysis

This section documents the methods/ analysis techniques used and presents the findings, along with supporting statistics and figures.

2.1 Data Exploration

We start by exploring the data to increase our understanding of the dataset.

First, let's see the number of missing data/ NAs in the dataset.

```
diabetes_data[diabetes_data == "?"] <- NA

# how many NAs are in the data
length(which(is.na(diabetes_data)))
```

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

There are no NAs in the dataset, so there is no need to remove or impute any rows.

Now, let's try to understand the data a little more.

```
# See the initial few rows of the data
head(diabetes_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   Diabetes
## 2                    0.351  31 NoDiabetes
## 3                    0.672  32   Diabetes
## 4                    0.167  21 NoDiabetes
## 5                    2.288  33   Diabetes
## 6                    0.201  30 NoDiabetes
```

```
# Understand the diabetes data better
str(diabetes_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                 : chr  "Diabetes" "NoDiabetes" "Diabetes" "NoDiabetes" ...
```

Each row represents a potential diabetes patient record.

We can see the number of patients affected with diabetes and those without it:

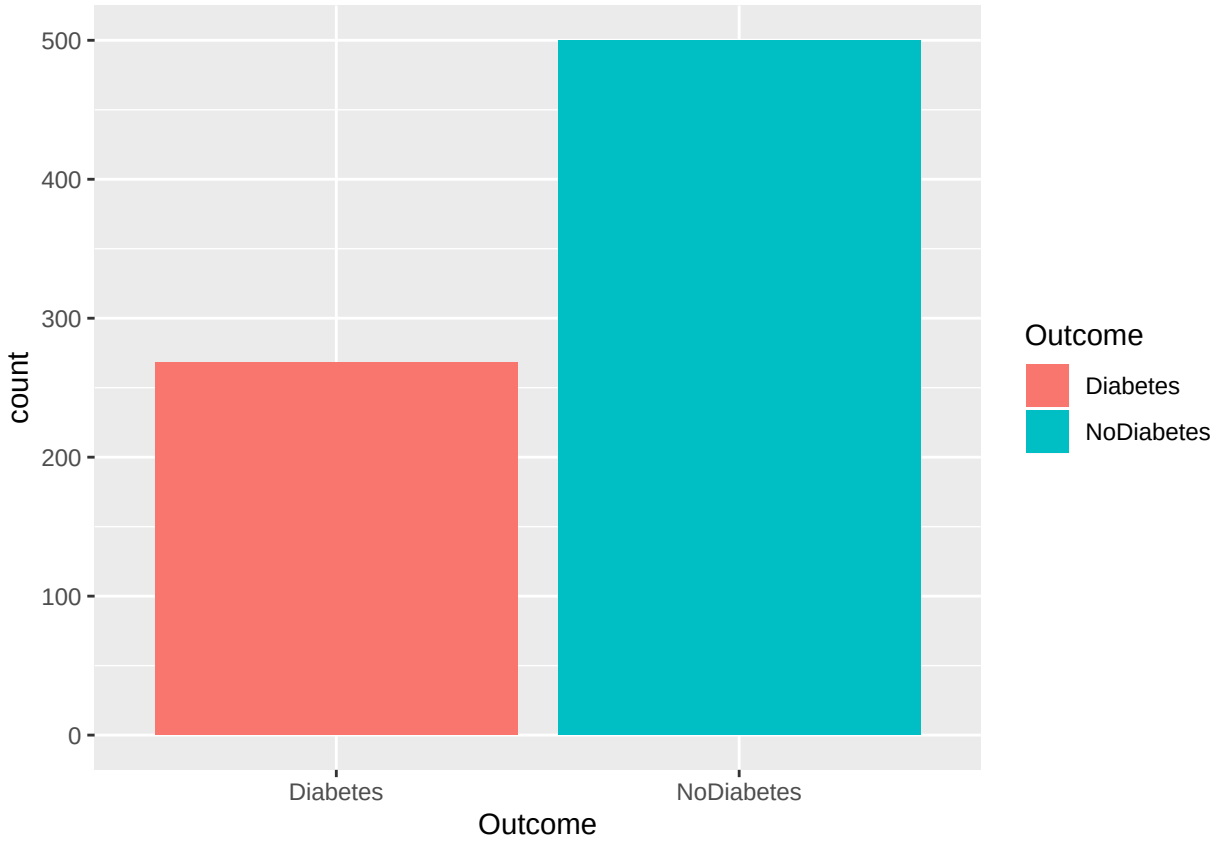
```
diabetes_data$Outcome <- as.factor(diabetes_data$Outcome)

summary(diabetes_data$Outcome)
```

```
##   Diabetes NoDiabetes
##        268        500
```

```
library(ggplot2)

ggplot(diabetes_data, aes(x = Outcome, fill = Outcome)) +
  geom_bar()
```



We see that our data is unbalanced with majority of the cases as false (500 out of 768). Most machine learning classification algorithms are sensitive to such imbalance. If we consider an even higher or extreme level of imbalance where say 95% of the patients do not have diabetes then even if our model simply predicts false for each case, it'll still have a very high accuracy which is extremely dangereous. Hence, there is a need to create an optimal model that would result in a high accuracy level combinated with high sensitivity i.e., a low rate of false-negatives.

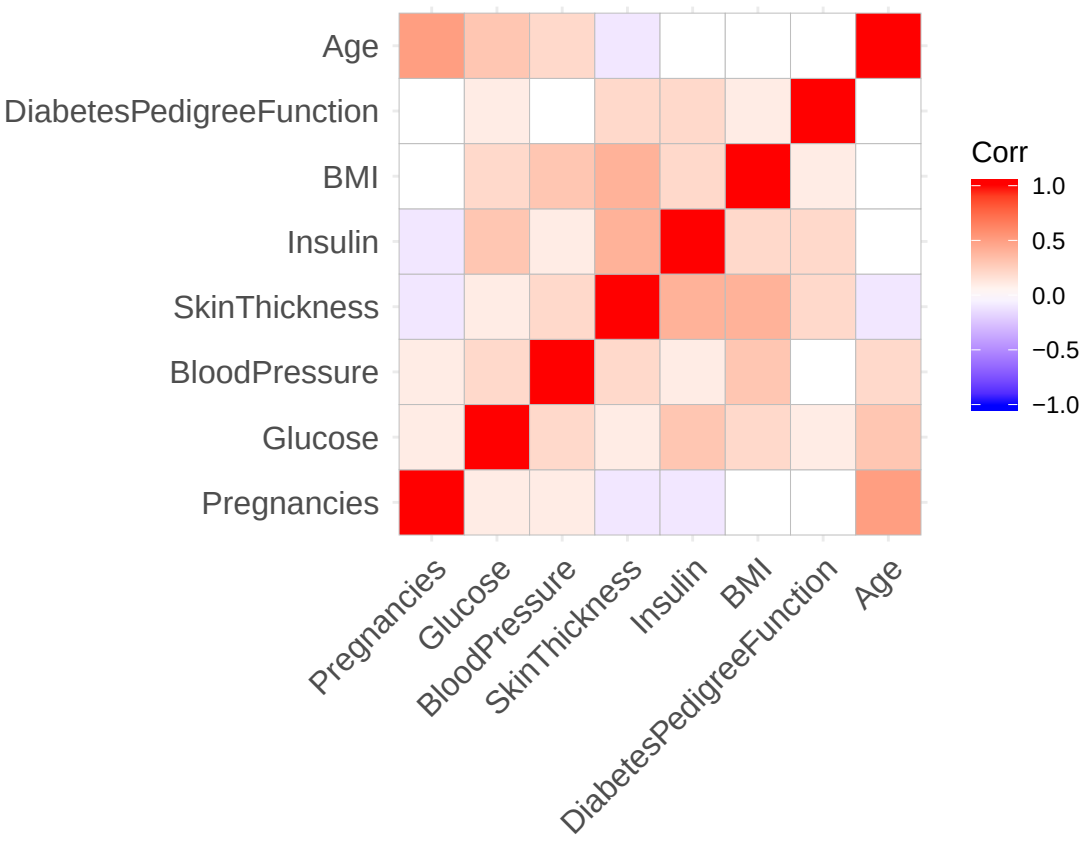
2.1.1 Correlation Matrix

We start our understanding and exploration of the features using the correlation matrix.

```r
# Compute correlation matrix
data_corr <- round(cor(diabetes_data[1:8]),1)
data_corr
```

```
##                           Pregnancies Glucose BloodPressure SkinThickness
## Pregnancies                       1.0     0.1           0.1          -0.1
## Glucose                           0.1     1.0           0.2           0.1
## BloodPressure                     0.1     0.2           1.0           0.2
## SkinThickness                    -0.1     0.1           0.2           1.0
## Insulin                          -0.1     0.3           0.1           0.4
## BMI                               0.0     0.2           0.3           0.4
## DiabetesPedigreeFunction          0.0     0.1           0.0           0.2
## Age                               0.5     0.3           0.2          -0.1
##                           Insulin BMI DiabetesPedigreeFunction  Age
## Pregnancies                  -0.1 0.0                      0.0  0.5
## Glucose                       0.3 0.2                      0.1  0.3
## BloodPressure                 0.1 0.3                      0.0  0.2
## SkinThickness                 0.4 0.4                      0.2 -0.1
## Insulin                       1.0 0.2                      0.2  0.0
## BMI                           0.2 1.0                      0.1  0.0
## DiabetesPedigreeFunction      0.2 0.1                      1.0  0.0
## Age                           0.0 0.0                      0.0  1.0
```

```r
# Plot the correlation matrix
library(ggcorrplot)
ggcorrplot(data_corr)
```



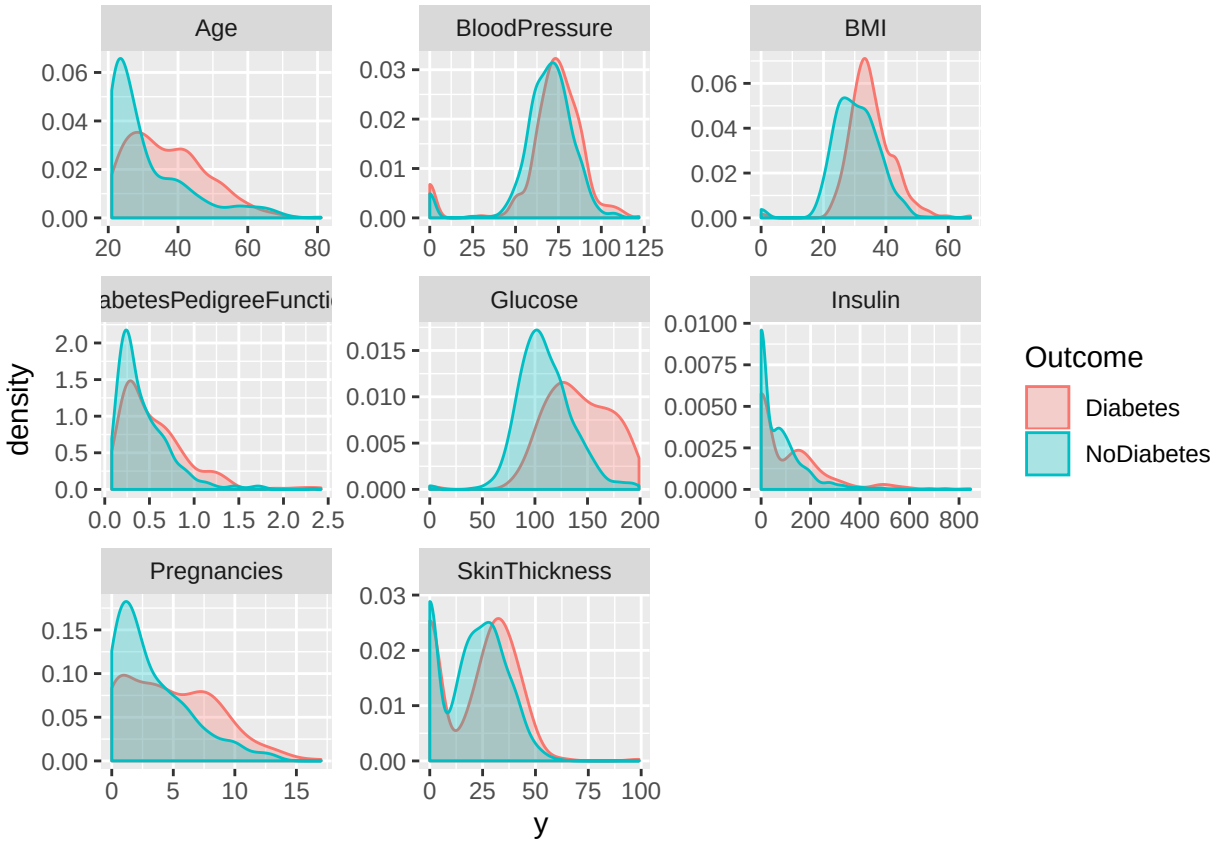
From the correlation matrix, we find that there is no strong correlation between any of the diagnosed factors and they are pretty much independent. Hence we do not have to remove any of the features as none of them are redundant.

2.1.2 Feature Importance

We further try to understand the importance of each feature by plotting the graphs of the features against the diabetes outcome.

```
#Understanding the features
library(tidyr)

gather(diabetes_data, x, y, Pregnancies:Age) %>%
  ggplot(aes(x = y, color = Outcome, fill = Outcome)) +
    geom_density(alpha = 0.3) +
    facet_wrap( ~ x, scales = "free", ncol = 3)
```



We infer from the graphs that Glucose level seems to be the most influential feature in determining if a person is diabetic or not

2.1.3 Split the Dataset

Before starting our modeling, we'll split the data into training (train_data) and testing (test_data) using a 90/10 split as that is considered a good split by experts. The testing data is not used for training the algorithm and is used for evaluating the accuracy of the algorithms. The testing set is chosen as 10% of the diabetes data so that our analysis can take place with the larger dataset of 90% of the diabetes data.

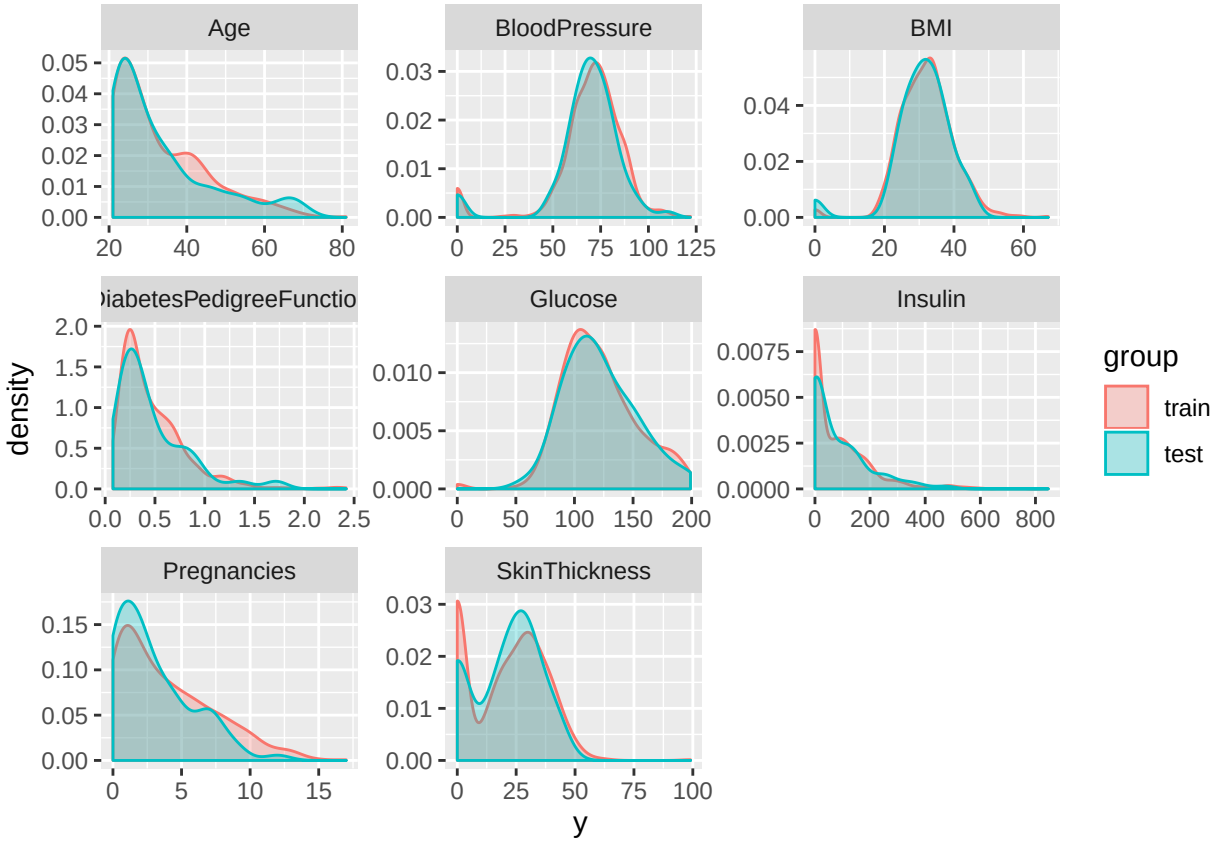
```
library(caret)
set.seed(42) # for reproducibility
test_index <- createDataPartition(diabetes_data$Outcome, p = 0.9, list = FALSE)
train_data <- diabetes_data[test_index, ]
test_data  <- diabetes_data[-test_index, ]
```

To ascertain that we have a good split, we check whether the distribution of feature values is comparable between training, validation and test datasets

```
library(dplyr)

rbind(data.frame(group = "train", train_data),
      data.frame(group = "test", test_data)) %>%
  gather(x, y, Pregnancies:Age) %>%
  ggplot(aes(x = y, color = group, fill = group)) +
```

```
    geom_density(alpha = 0.3) +
    facet_wrap( ~ x, scales = "free", ncol = 3)
```



Since the distribution between the two is similar, we can continue with the modeling part.

2.2 Modeling

We use several different algorithms for building our models beginning with Logistic Regression followed by K-Nearest Neighbors, Decision trees, Random Forest and Gradient Boosting.

The parameters of the below variable ComputationControl are those that we use for controlling the computation of train function in most of the models that we build.

```
ComputationControl <- trainControl(method="repeatedcv",
                                   number = 15,
                                   repeats = 10,
                                   classProbs = TRUE,
                                   summaryFunction = twoClassSummary)
```

2.2.1 Logistic Regression

We start with Logistic Regression which is the most commonly used method for binary classification and prediction problems.

```
# Logistic Regression

model_reg <- caret::train(Outcome ~ .,
                          data = train_data,
                          method = "glm",
                          metric = "ROC",
                          preProcess = c("scale", "center"),
                          trControl =  ComputationControl)
# Prediction
prediction_reg<- predict(model_reg, test_data)

# Confusion matrix
confusionMat_reg <- confusionMatrix(prediction_reg, test_data$Outcome)

confusionMat_reg
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Diabetes NoDiabetes
##   Diabetes         15          4
##   NoDiabetes       11         46
##
##                Accuracy : 0.8026
##                  95% CI : (0.6954, 0.8851)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 0.004209
##
##                   Kappa : 0.5313
##
##  Mcnemar's Test P-Value : 0.121335
##
##             Sensitivity : 0.5769
##             Specificity : 0.9200
##          Pos Pred Value : 0.7895
##          Neg Pred Value : 0.8070
##              Prevalence : 0.3421
##          Detection Rate : 0.1974
##    Detection Prevalence : 0.2500
##       Balanced Accuracy : 0.7485
##
##        'Positive' Class : Diabetes
##
```

```
# Accuracy of model
accuracy_reg <- confusionMat_reg$overall['Accuracy']
```

As we go along, there will be a need to compare different approaches. Hence, starting by creating a results table with this approach:

```
accuracy_results <- tibble(Method = "Logistic Regression", Accuracy = accuracy_reg)
```

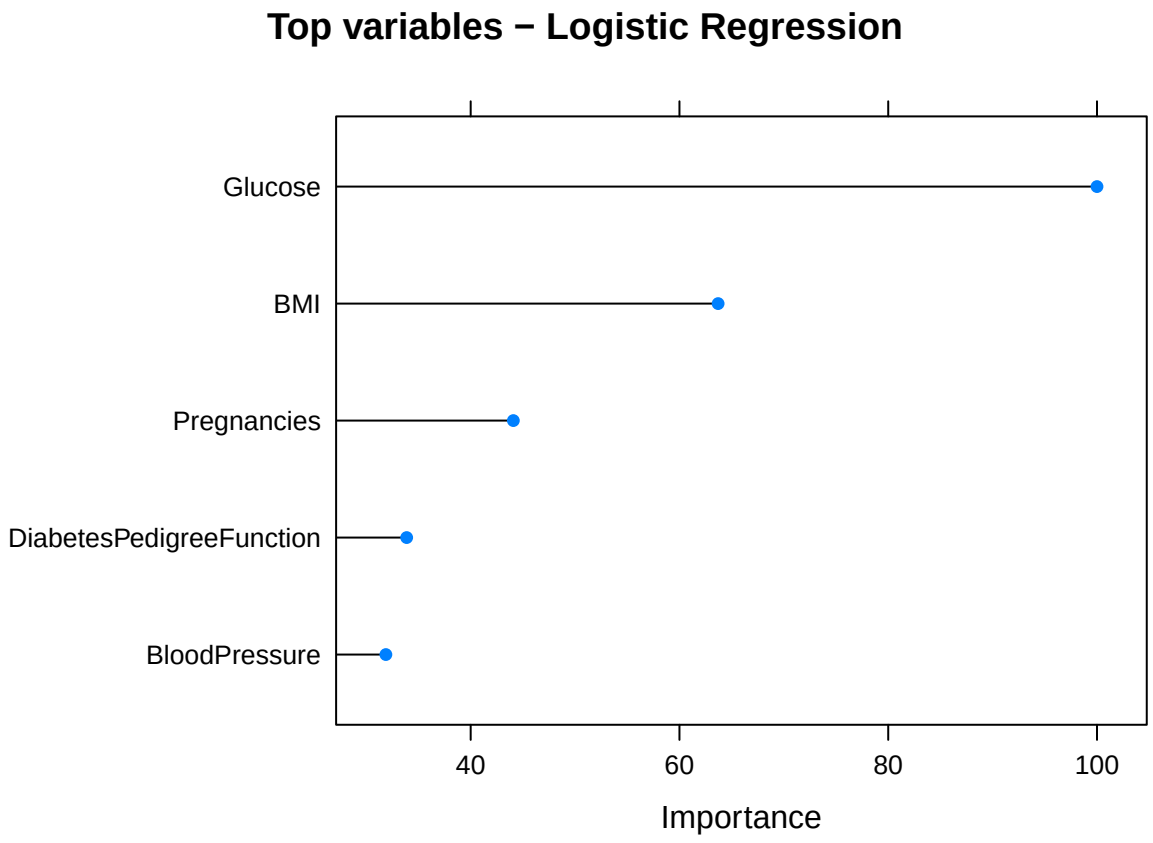
Viewing the results obtained so far:

```
accuracy_results %>% knitr::kable()
```

Method	Accuracy
Logistic Regression	0.8026316

Now, we make a plot to understand which were the most important variables or factors for our regression model.

```
# Plot of top five most important variables
plot(varImp(model_reg), top=5, main="Top variables - Logistic Regression")
```



2.2.2 KNN (K Nearest Neighbor)

The general idea behind KNN is to classify patients by their similarity to other patients. We are going to build our next model based on KNN and see which are the top five most important variables according to this method.

```
# KNN

model_knn <- caret::train(Outcome ~ .,
                          data = train_data,
                          method = "knn",
                          metric = "ROC",
                          preProcess = NULL,
                          trControl =  ComputationControl)

# Prediction
prediction_knn<- predict(model_knn, test_data)

# Confusion matrix
confusionMat_knn <- confusionMatrix(prediction_knn, test_data$Outcome)

confusionMat_knn
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Diabetes NoDiabetes
##   Diabetes         14          5
##   NoDiabetes       12         45
##
##                Accuracy : 0.7763
##                  95% CI : (0.6662, 0.864)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 0.01749
##
##                   Kappa : 0.4687
##
##  Mcnemar's Test P-Value : 0.14561
##
##             Sensitivity : 0.5385
##             Specificity : 0.9000
##          Pos Pred Value : 0.7368
##          Neg Pred Value : 0.7895
##              Prevalence : 0.3421
##          Detection Rate : 0.1842
##    Detection Prevalence : 0.2500
##       Balanced Accuracy : 0.7192
##
##        'Positive' Class : Diabetes
##
```

```
# Accuracy of model
accuracy_knn <- confusionMat_knn$overall['Accuracy']
```

```
accuracy_results <- add_row(accuracy_results, Method = "KNN", Accuracy = accuracy_knn)
```

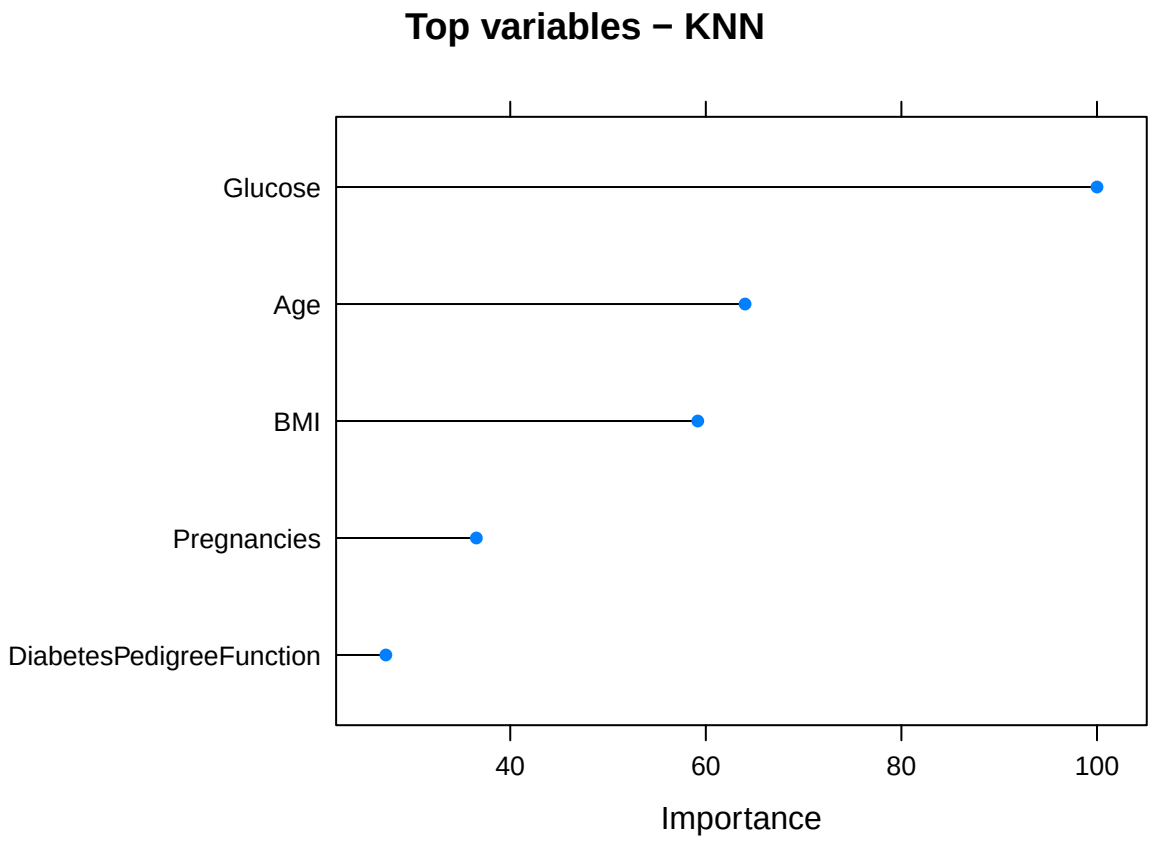
Viewing the results obtained so far:

```
accuracy_results %>% knitr::kable()
```

Method	Accuracy
Logistic Regression	0.8026316
KNN	0.7763158

Plotting the top five most important variables for this model:

```
# Plot of top five most important variables
plot(varImp(model_knn), top=5, main="Top variables - KNN")
```



We see that there is a difference in the variables as KNN gives importance to Age also but for logistic regression, Blood Pressure was more important.

2.2.3 Decision tree

Our next classification model will use decision tree by constructing nodes at which data is separated and terminating in leaves at which we find the model's assigned class.

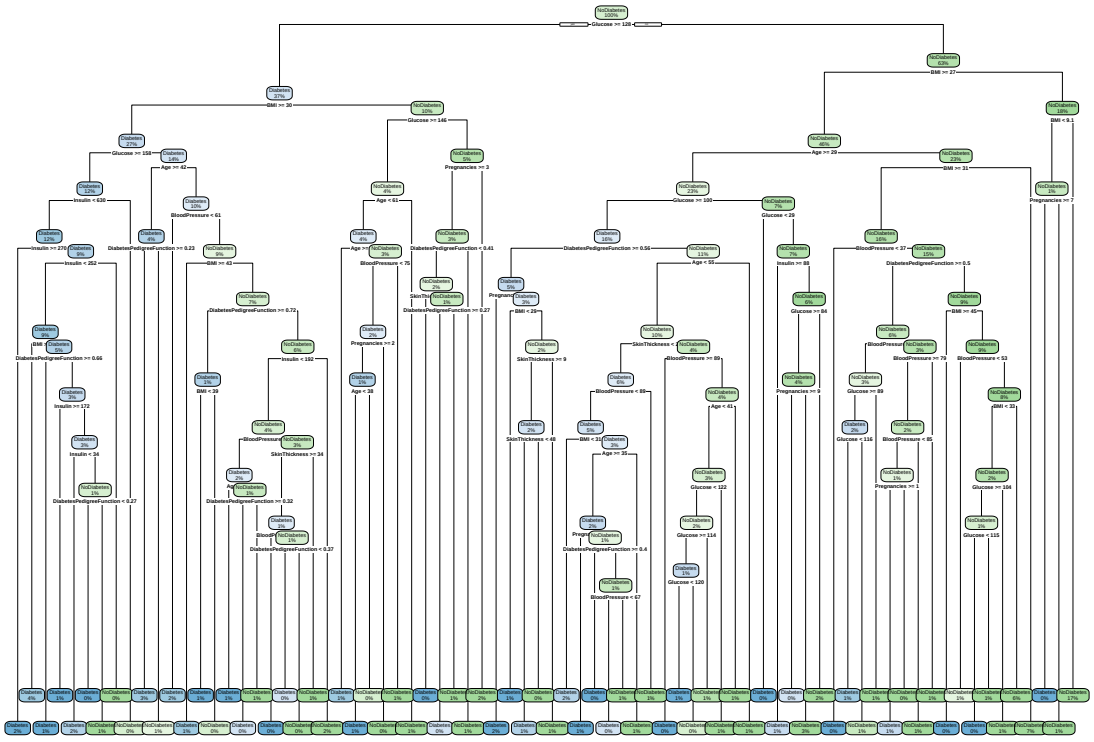
```
# Decision Tree

library(rpart)
library(rpart.plot)

model_dt <- rpart(Outcome ~ .,
            data = train_data,
            method = "class",
            control = rpart.control(xval = 10,
                                    minbucket = 2,
                                    cp = 0),
             parms = list(split = "information"))

# Plot the tree
rpart.plot(model_dt, extra = 100)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```r
# Prediction
prediction_dt<- predict(model_dt, test_data, type = "class")

# Confusion matrix
confusionMat_dt <- confusionMatrix(prediction_dt, test_data$Outcome)

confusionMat_dt
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Diabetes NoDiabetes
##   Diabetes         16         12
##   NoDiabetes       10         38
##
##                Accuracy : 0.7105
##                  95% CI : (0.5951, 0.8089)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 0.1998
##
##                   Kappa : 0.3686
##
##  Mcnemar's Test P-Value : 0.8312
##
##             Sensitivity : 0.6154
##             Specificity : 0.7600
##          Pos Pred Value : 0.5714
##          Neg Pred Value : 0.7917
##              Prevalence : 0.3421
##          Detection Rate : 0.2105
##    Detection Prevalence : 0.3684
##       Balanced Accuracy : 0.6877
##
##        'Positive' Class : Diabetes
##
```

```r
# Accuracy of model
accuracy_dt <- confusionMat_dt$overall['Accuracy']

accuracy_results <- add_row(accuracy_results, Method = "Decision Tree", Accuracy = accuracy_dt)
```

Viewing the results obtained so far:

```r
accuracy_results %>% knitr::kable()
```

Method	Accuracy
Logistic Regression	0.8026316
KNN	0.7763158
Decision Tree	0.7105263

2.2.4 Random Forest

Our next model is built using Random Forest which aggregates multiple decorrelated decision trees in order to yield a prediction.

```
# Random Forest

model_rf <- caret::train(Outcome ~ .,
                         data = train_data,
                         method = "rf",
                         metric = "ROC",
                         preProcess = c("scale", "center"),
                         trControl = ComputationControl)

# Prediction
prediction_rf<- predict(model_rf, test_data)

# Confusion matrix
confusionMat_rf <- confusionMatrix(prediction_rf, test_data$Outcome)

confusionMat_rf
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Diabetes NoDiabetes
##   Diabetes         18          5
##   NoDiabetes        8         45
##
##                Accuracy : 0.8289
##                  95% CI : (0.7253, 0.9057)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 0.0007472
##
##                   Kappa : 0.6092
##
##  Mcnemar's Test P-Value : 0.5790997
##
##             Sensitivity : 0.6923
##             Specificity : 0.9000
##          Pos Pred Value : 0.7826
##          Neg Pred Value : 0.8491
##              Prevalence : 0.3421
##          Detection Rate : 0.2368
##    Detection Prevalence : 0.3026
##       Balanced Accuracy : 0.7962
##
##        'Positive' Class : Diabetes
##
```

```
# Accuracy of model
accuracy_rf <- confusionMat_rf$overall['Accuracy']

accuracy_results <- add_row(accuracy_results, Method = "Random Forest", Accuracy = accuracy_rf)
```

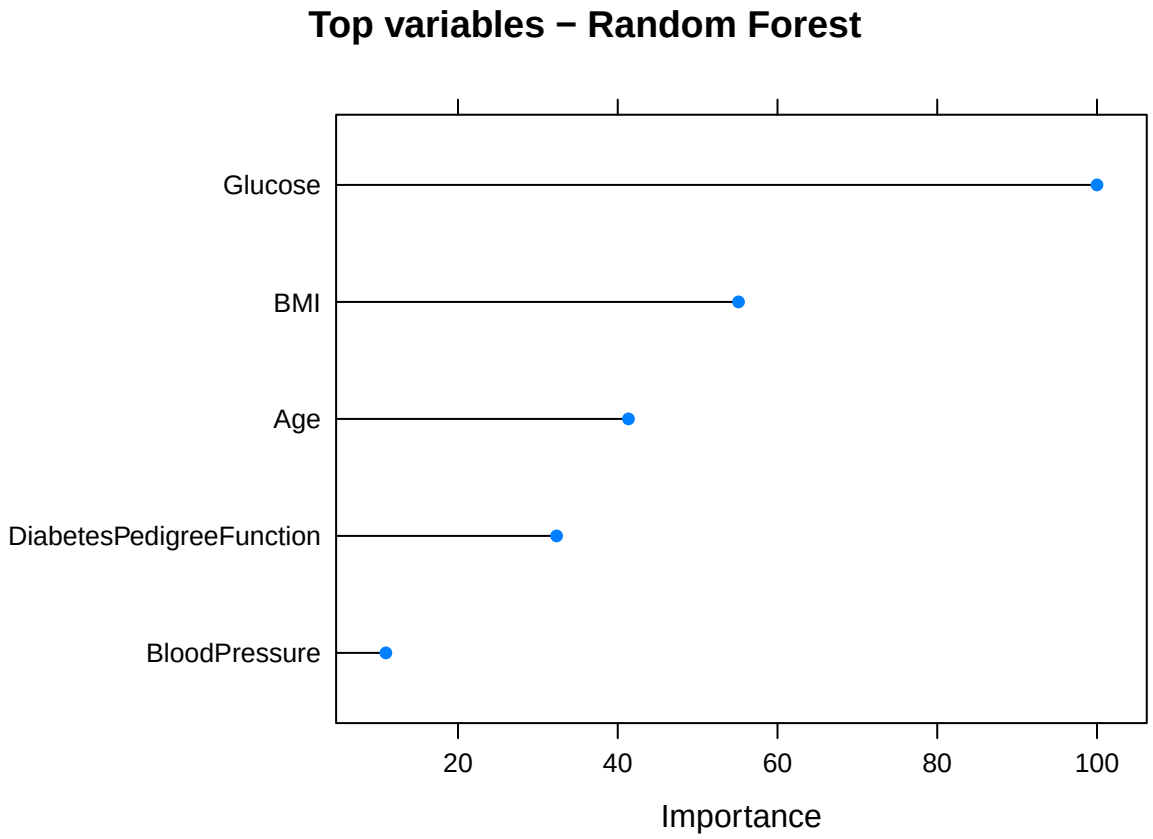
Viewing the results obtained so far:

```
accuracy_results %>% knitr::kable()
```

Method	Accuracy
Logistic Regression	0.8026316
KNN	0.7763158
Decision Tree	0.7105263
Random Forest	0.8289474

Plot of the top five most important variables for random forest:

```
# Plot of top five most important variables
plot(varImp(model_rf), top=5, main="Top variables - Random Forest")
```



2.2.5 Gradient Boosting

We next use Gradient Boosting which ia an ensemble learning method that trains many models in a gradual, additive and sequential manner.

```
# Gradient Boosting

model_gb <- caret::train(Outcome ~ .,
                         data = train_data,
                         method = "gbm",
                         metric = "ROC",
                         trControl = ComputationControl,
                         verbose = 0)

# Prediction
prediction_gb<- predict(model_gb, test_data)

# Confusion matrix
confusionMat_gb <- confusionMatrix(prediction_gb, test_data$Outcome)

confusionMat_gb
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Diabetes NoDiabetes
##   Diabetes         14          4
##   NoDiabetes       12         46
##
##                Accuracy : 0.7895
##                  95% CI : (0.6808, 0.8746)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 0.008892
##
##                   Kappa : 0.495
##
##  Mcnemar's Test P-Value : 0.080118
##
##             Sensitivity : 0.5385
##             Specificity : 0.9200
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.7931
##              Prevalence : 0.3421
##          Detection Rate : 0.1842
##    Detection Prevalence : 0.2368
##       Balanced Accuracy : 0.7292
##
##        'Positive' Class : Diabetes
##
```

```
# Accuracy of model
accuracy_gb <- confusionMat_gb$overall['Accuracy']

accuracy_results <- add_row(accuracy_results, Method = "Gradient Boosting", Accuracy = accuracy_gb)
```

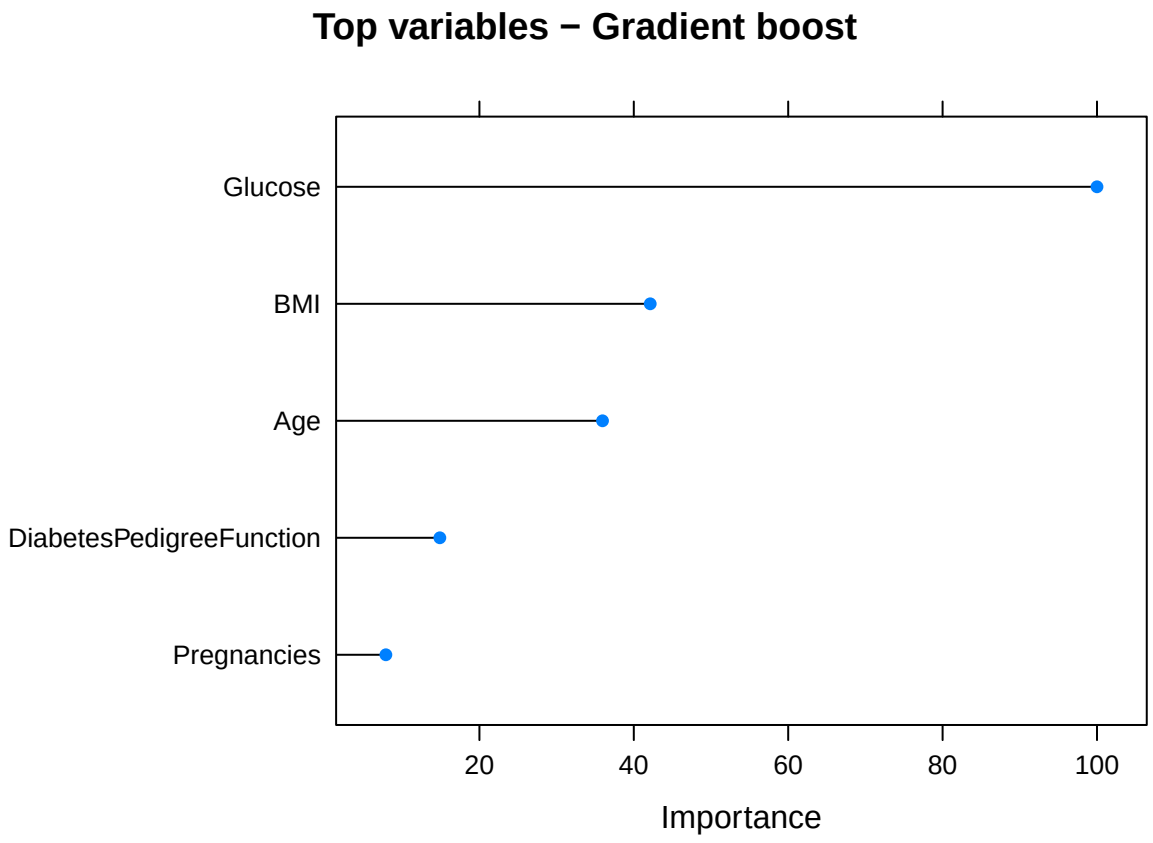
Viewing the results obtained so far:

```
accuracy_results %>% knitr::kable()
```

Method	Accuracy
Logistic Regression	0.8026316
KNN	0.7763158
Decision Tree	0.7105263
Random Forest	0.8289474
Gradient Boosting	0.7894737

Plot of top five most important variables for this model:

```
# Plot of top five most important variables
plot(varImp(model_gb), top=5, main="Top variables - Gradient boost")
```



3 Results

These are the final accuracy values of all the models constructed:

```
accuracy_results %>% knitr::kable()
```

Method	Accuracy
Logistic Regression	0.8026316
KNN	0.7763158
Decision Tree	0.7105263
Random Forest	0.8289474
Gradient Boosting	0.7894737

From the accuracy results, the best performing model is random forest. Now, we re-check the sensitivity of the Random Forest model using its confusion matrix.

```
# Random Forest Confusion matrix

confusionMat_rf
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Diabetes NoDiabetes
##   Diabetes         18          5
##   NoDiabetes        8         45
##
##                Accuracy : 0.8289
##                  95% CI : (0.7253, 0.9057)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 0.0007472
##
##                   Kappa : 0.6092
##
##  Mcnemar's Test P-Value : 0.5790997
##
##             Sensitivity : 0.6923
##             Specificity : 0.9000
##          Pos Pred Value : 0.7826
##          Neg Pred Value : 0.8491
##              Prevalence : 0.3421
##          Detection Rate : 0.2368
##    Detection Prevalence : 0.3026
##       Balanced Accuracy : 0.7962
##
##        'Positive' Class : Diabetes
##
```

If we go back and check the sensitivity of the other models, we find that Random Forest has the highest sensitivity also.

4 Conclusion

We have successfully constructed multiple models for our diabetes prediction system. Among the models designed, the Random Forest model gives the highest accuracy and sensitivity of 0.8289474 and 0.6923 respectively. Therefore, we conclude that Random Forest performs best among all the models constructed for this particular dataset.

As is the case with any disease related dataset, there is always a limitation of having lesser number of positive cases and hence an imbalance. For further improvement, there are methods that can be used to balance out the dataset or we can pick a much larger dataset that would have many more cases, both positives and negatives.

Also, we see that in all the models glucose level is the topmost variable and we can identify this directly from the feature importance graphs that we had plotted earlier which further strengthens the need to understand the features in a dataset and its impact on the results before starting any modeling.

5 References

1. https://rafalab.github.io/dsbook
2. https://towardsdatascience.com
3. https://www.kaggle.com