Diabetes Prediction Model
HarvardX's Professional Certificate in Data Science
Machine Learning Capstone Project

Usha Saravanakumar

October 2022

Contents

1 Introduction

Diabetes is a chronic (long-lasting) disease that affects how our body turns food into energy. According to the ***CDC***, More than 37 million US adults have diabetes, and 1 in 5 do not know they have diabetes. In the last 20 years, the number of adults diagnosed with diabetes has more than doubled. Diabetes is the seventh leading cause of death in the United States. Diabetes is the No. 1 cause of kidney failure, lower-limb amputations, and adult blindness. This project aims at developing a machine-learning model that accurately predicts if a person has diabetes or not.

1.1 Diabetes Dataset

The diabetes dataset used in this project is from ***Kaggle***, containing nine attributes for 768 entries. All patients in the dataset are females of Pima Indian heritage at least 21 years old.

1.1.1 Predictor Variables:

- Pregnancies: Number of pregnancies
- Glucose: Glucose level in blood
- Blood Pressure: Blood pressure measurement
- SkinThickness: Thickness of the skin
- Insulin: Insulin level in the blood
- BMI: Body mass index
- DiabetesPedigreeFunction: Diabetes percentage
- Age: Age

1.1.2 Target Variable:

- Outcome:
  - 1: The patient has diabetes
  - 0: The patient does not have diabetes

1.2 Process and Workflow

The main steps in the project involve the following:

- Data Extraction: Download the dataset from Kaggle, upload the dataset to Github, import the dataset from GitHub, and prepare the data
- Exploring the dataset's structure.
- Performing Exploratory Data Analysis (EDA).
- Developing a machine-learning classification model for classifying the patients as diabetic or non-diabetic.

2 Methods/Analysis

2.1 Data Extraction

2.1.1 Install and import required packages

```
if(!require(formatR)) install.packages("formatR", repos = "http://cran.us.r-project.org")
if(!require(tinytex)) tinytex::install_tinytex()
if(!require(kableExtra)) install.packages("kableExtra", repos = "http://cran.us.r-project.org")
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(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages('corrplot', repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")

library(tidyverse, warn.conflicts=F, quietly=T)
library(caret, warn.conflicts=F, quietly=T)
library(data.table, warn.conflicts=F, quietly=T)
library(dplyr, warn.conflicts = F, quietly=T)
library(kableExtra, warn.conflicts = F, quietly=T)
library(formatR, warn.conflicts=F, quietly=T)
library(knitr, warn.conflicts=F, quietly=T)
library(corrplot)
library(rpart)
library(xgboost)
```

2.1.2 Download Diabetes Dataset

```
url <- "https://raw.githubusercontent.com/ushaakumaar/diabetes_prediction/main/dataset/diabetes.csv"
diabetes_df <- fread(url)
head(as_tibble(diabetes_df))
```

```
## # A tibble: 6 x 9
##   Pregnancies Glucose BloodPressure SkinTh~1 Insulin   BMI Diabe~2   Age Outcome
##         <int>   <int>         <int>    <int>   <int> <dbl>   <dbl> <int>   <int>
## 1           6     148            72       35       0  33.6   0.627    50       1
## 2           1      85            66       29       0  26.6   0.351    31       0
## 3           8     183            64        0       0  23.3   0.672    32       1
## 4           1      89            66       23      94  28.1   0.167    21       0
## 5           0     137            40       35     168  43.1   2.29     33       1
## 6           5     116            74        0       0  25.6   0.201    30       0
## # ... with abbreviated variable names 1: SkinThickness,
## #   2: DiabetesPedigreeFunction
```

2.2 Data Exploration and Transformation

Let us begin by splitting the dataset into training and testing datasets and then start exploring the structure of the dataset.

2.2.1 Split dataset to train and test datasets

We split the Diabetes dataset into a training dataset called train_set and an evaluation dataset called test_set. The train and test set will have 90% and 10% of Diabetes data, respectively.

```
# Training dataset will be 10% of Diabetes data
set.seed(1)
test_index <- createDataPartition(y = diabetes_df$Outcome, times = 1, p = 0.1, list = FALSE)
train_set <- diabetes_df[-test_index,]
test_set <- diabetes_df[test_index,]

str(train_set)
```

```
## Classes ’data.table’ and ’data.frame’:   691 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 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
str(test_set)
```

```
## Classes ’data.table’ and ’data.frame’:   77 obs. of  9 variables:
##  $ Pregnancies             : int  8 13 11 2 4 7 0 1 2 0 ...
##  $ Glucose                 : int  99 145 138 90 111 159 101 81 85 95 ...
##  $ BloodPressure           : int  84 82 76 68 72 64 65 72 65 85 ...
##  $ SkinThickness           : int  0 19 0 42 47 0 28 18 0 25 ...
##  $ Insulin                 : int  0 110 0 0 207 0 0 40 0 36 ...
##  $ BMI                     : num  35.4 22.2 33.2 38.2 37.1 27.4 24.6 26.6 39.6 37.4 ...
##  $ DiabetesPedigreeFunction: num  0.388 0.245 0.42 0.503 1.39 0.294 0.237 0.283 0.93 0.247 ...
##  $ Age                     : int  50 57 35 27 56 40 22 24 27 24 ...
##  $ Outcome                 : int  0 0 0 1 1 0 0 0 0 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

2.2.2 Structure of dataset

Diabetes dataset has 768 observations, 8 predictor variables and 1 target variable. All variables are numeric.

```
## Classes ’data.table’ and ’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 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

2.2.3 Summary Statistics of dataset

The below statistics show that the minimum value for Glucose, blood pressure, skin thickness, Insulin, and BMI is 0. It is ideally impossible for a person to have 0 glucose level or blood pressure or skin thickness or insulin level, or BMI. It appears to be a data error or missing 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  
```

Now, let us check what percentage of each variable contains missing data. It is evident from the below table that Glucose, blood pressure, and BMI have less than 5% of data missing, while 30% of SkinThickness and almost 50% of Insulin data are incorrect or missing.

Table 1: Percentage of Variable having Missing data

Variable	Percentage
Glucose - Missing Percentage	0.6510
BloodPressure - Missing Percentage	4.5573
SkinThickness - Missing Percentage	29.5573
Insulin - Missing Percentage	48.6979
BMI - Missing Percentage	1.4323

2.2.4 Median Imputation

In statistics, Median imputation is the process of replacing all occurrences of missing values within a variable ("0" in our case) with the median. We can perform Median imputation when data are missing completely at random and when no more than 5% of the variable contains missing data. So, let us perform median imputation by replacing 0s with median values for Glucose, blood pressure, and BMI. The median value should be calculated only in the training set and used to replace missing data in both training and test sets to avoid over-fitting.

```r
# Find median values in training set
median_glucose <- as.integer(median(train_set$Glucose))
```

```r
median_bp <- as.integer(median(train_set$BloodPressure))
median_bmi <- median(train_set$BMI)

# Replace missing data in training set with Median
train_set$Glucose[train_set$Glucose == 0] <- median_glucose
train_set$BloodPressure[train_set$BloodPressure == 0] <- median_bp
train_set$BMI[train_set$BMI == 0] <- median_bmi

# Replace missing data in test set with Median
test_set$Glucose[test_set$Glucose == 0] <- median_glucose
test_set$BloodPressure[test_set$BloodPressure == 0] <- median_bp
test_set$BMI[test_set$BMI == 0] <- median_bmi

# Print summary statistics
summary(train_set)
```

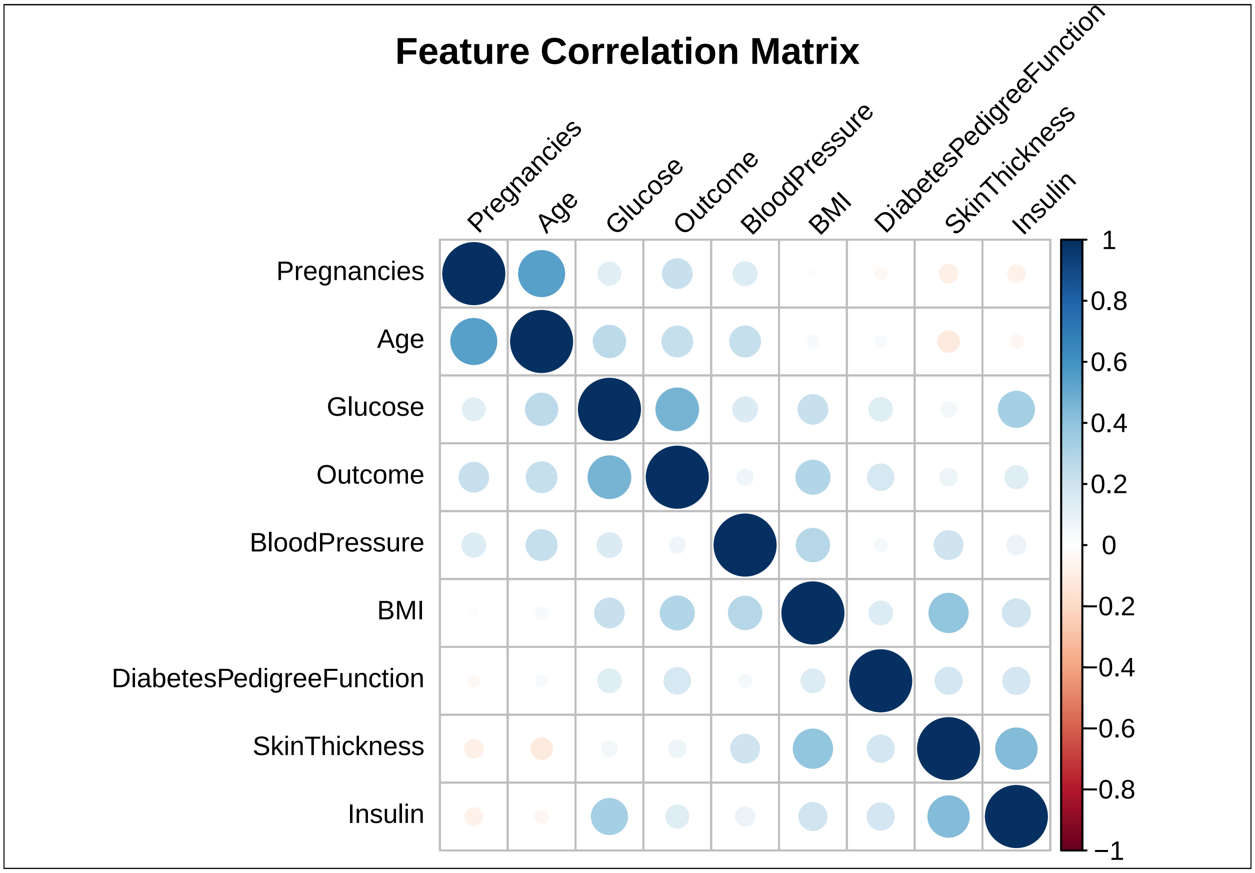
```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.:100.0   1st Qu.: 64.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.812   Mean   :121.7   Mean   : 72.21   Mean   :20.71
##  3rd Qu.: 6.000   3rd Qu.:140.0   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.   :  0.00   Min.   :18.20   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.00   1st Qu.:27.60   1st Qu.:0.2450           1st Qu.:24.00
##  Median : 41.00   Median :32.10   Median :0.3710           Median :29.00
##  Mean   : 80.61   Mean   :32.46   Mean   :0.4723           Mean   :33.19
##  3rd Qu.:129.50   3rd Qu.:36.55   3rd Qu.:0.6300           3rd Qu.:41.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.3488
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

```r
summary(test_set)
```

```
##   Pregnancies        Glucose    BloodPressure SkinThickness      Insulin
##  Min.   : 0.000   Min.   : 67   Min.   :50    Min.   : 0.00   Min.   :  0.00
##  1st Qu.: 1.000   1st Qu.: 99   1st Qu.:68    1st Qu.: 0.00   1st Qu.:  0.00
##  Median : 3.000   Median :115   Median :72    Median :21.00   Median :  0.00
##  Mean   : 4.143   Mean   :121   Mean   :74    Mean   :18.95   Mean   : 72.49
##  3rd Qu.: 7.000   3rd Qu.:145   3rd Qu.:82    3rd Qu.:33.00   3rd Qu.:110.00
##  Max.   :14.000   Max.   :195   Max.   :98    Max.   :56.00   Max.   :680.00
##       BMI        DiabetesPedigreeFunction      Age           Outcome
##  Min.   :20.00   Min.   :0.1220           Min.   :21.00   Min.   :0.0000
##  1st Qu.:25.80   1st Qu.:0.2370           1st Qu.:24.00   1st Qu.:0.0000
##  Median :32.00   Median :0.3830           Median :28.00   Median :0.0000
##  Mean   :32.37   Mean   :0.4678           Mean   :33.71   Mean   :0.3506
##  3rd Qu.:36.60   3rd Qu.:0.6070           3rd Qu.:42.00   3rd Qu.:1.0000
##  Max.   :53.20   Max.   :2.1370           Max.   :60.00   Max.   :1.0000
```

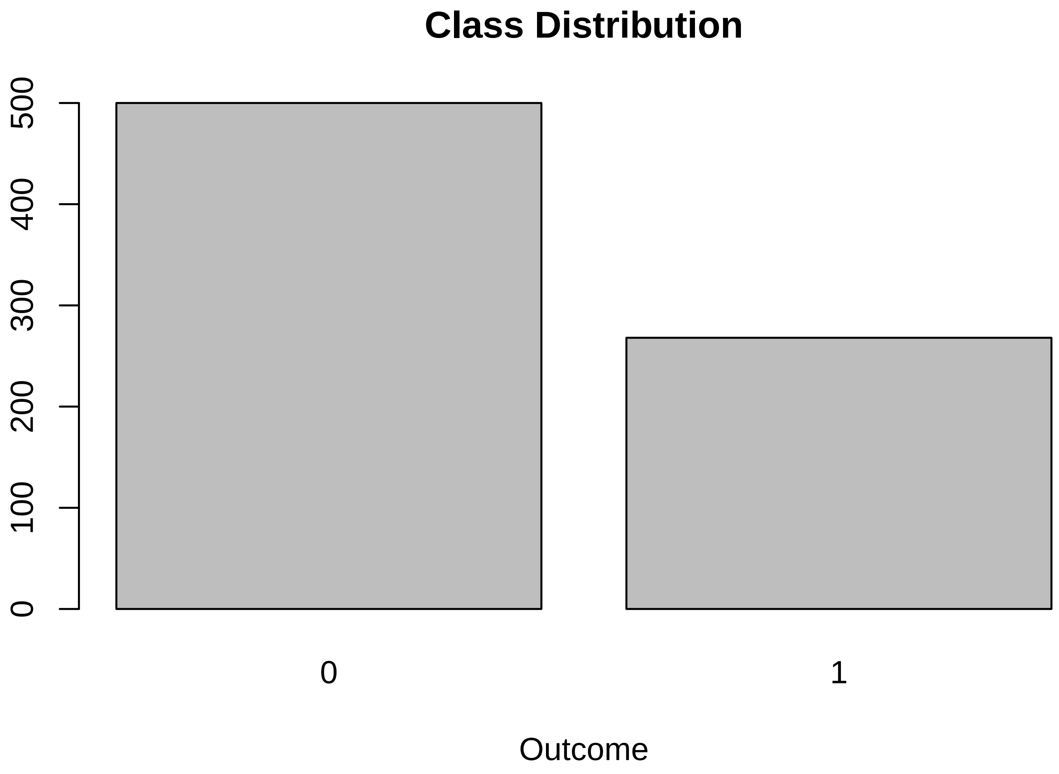
2.2.5 Feature Correlation Matrix

We could observe from the below correlation matrix that Glucose, Age, Pregnancies, and BMI have the most direct correlation with Outcome. In contrast, Insulin, SkinThickness, and Blood Pressure have the most negligible direct correlation with Outcome. SkinThickness and Insulin have an inverse correlation with Age and Pregnancies.



2.2.6 Class Distribution

Out of 768 entries, there are 500 with Outcome ‘0’ and 268 with Outcome ‘1’. Class Imbalance is not defined formally, but a ratio of 1 to 10 is usually considered imbalanced enough to benefit from balancing techniques.



3 Machine Learning Models

We noticed during data exploration that SkinThickness and Insulin have approximately 30% and 50% of data missing, respectively. The correlation Matrix also revealed that these variables have the most negligible direct correlation, So we will not use SkinThickness and Insulin in our Models.

3.1 Multi Linear Regression Model

We could see from the summary that the P-values of Glucose, BMI and Pregnancies are the least so they are the top three most relevant features.

```
# Create the model
mlr_model <- lm(Outcome ~ . - SkinThickness - Insulin, data=train_set)

# Print the model summary statistics
summary(mlr_model)
```

```
##
## Call:
## lm(formula = Outcome ~ . - SkinThickness - Insulin, data = train_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.13551 -0.28343 -0.07239  0.28359  0.99709
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -1.0235220  0.1076751  -9.506  < 2e-16 ***
## Pregnancies               0.0223077  0.0053168   4.196 3.08e-05 ***
## Glucose                   0.0065300  0.0005314  12.287  < 2e-16 ***
## BloodPressure            -0.0009046  0.0013588  -0.666  0.50580
## BMI                       0.0133056  0.0023770   5.598 3.15e-08 ***
## DiabetesPedigreeFunction  0.1276376  0.0462052   2.762  0.00589 **
## Age                       0.0019720  0.0015994   1.233  0.21801
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.3916 on 684 degrees of freedom
## Multiple R-squared:  0.3317, Adjusted R-squared:  0.3259
## F-statistic: 56.59 on 6 and 684 DF,  p-value: < 2.2e-16
```

3.1.1 Cross Validation

Let us test the model against test dataset and check the model accuracy.

```r
# Predict outcome in the testing set
mlr_prediction <- predict(mlr_model,newdata=test_set,type='response')
mlr_prediction <- ifelse(mlr_prediction > 0.5,1,0)

# Print confusion matrix
(confusion_matrix_mlr<-table(mlr_prediction, test_set$Outcome))
```

```
##
## mlr_prediction  0  1
##              0 42 11
##              1  8 16
```

Table 2: MODEL PERFORMANCE RESULTS

Method	Accuracy
Model 1: Multi Linear Regression Model	0.7532

3.2 Logistic Regression Model

Let us try the Logistic regression model on our dataset. Since our Outcome is binary (zeroes and ones), we will use binomial distribution. We could see from the summary again that the P-values of Glucose, BMI, and Pregnancy are the least, so they are the top three most relevant features.

```r
# Create the model
logistic_reg_model <- glm(Outcome ~ . - SkinThickness - Insulin, data=train_set, family="binomial")

# Print the model summary statistics
summary(logistic_reg_model)
```

```
##
## Call:
## glm(formula = Outcome ~ . - SkinThickness - Insulin, family = "binomial",
##     data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8381  -0.7048  -0.3874   0.6786   2.4280
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -9.220252   0.861187 -10.706  < 2e-16 ***
## Pregnancies               0.137227   0.034669   3.958 7.55e-05 ***
## Glucose                   0.037927   0.003892   9.745  < 2e-16 ***
## BloodPressure            -0.007415   0.009072  -0.817  0.41375
## BMI                       0.089865   0.016606   5.412 6.25e-08 ***
## DiabetesPedigreeFunction  0.877586   0.315309   2.783  0.00538 **
## Age                       0.012459   0.010011   1.245  0.21331
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 893.71  on 690  degrees of freedom
## Residual deviance: 631.97  on 684  degrees of freedom
## AIC: 645.97
##
## Number of Fisher Scoring iterations: 5
```

3.2.1 Cross Validation

Let us test the model against test dataset and check the model accuracy.

```
# Predict outcome in the testing set
logistic_reg_prediction <- predict(logistic_reg_model,newdata=test_set,type='response')
logistic_reg_prediction <- ifelse(logistic_reg_prediction > 0.5,1,0)

# Print confusion matrix
(confusion_matrix_logistic_reg<-table(logistic_reg_prediction, test_set$Outcome))
```

```
##
## logistic_reg_prediction  0  1
##                       0 42 11
##                       1  8 16
```

Table 3: MODEL PERFORMANCE RESULTS

Method	Accuracy
Model 1: Multi Linear Regression Model	0.7532
Model 2: Logistic Regression	0.7532

3.3 Decision Tree

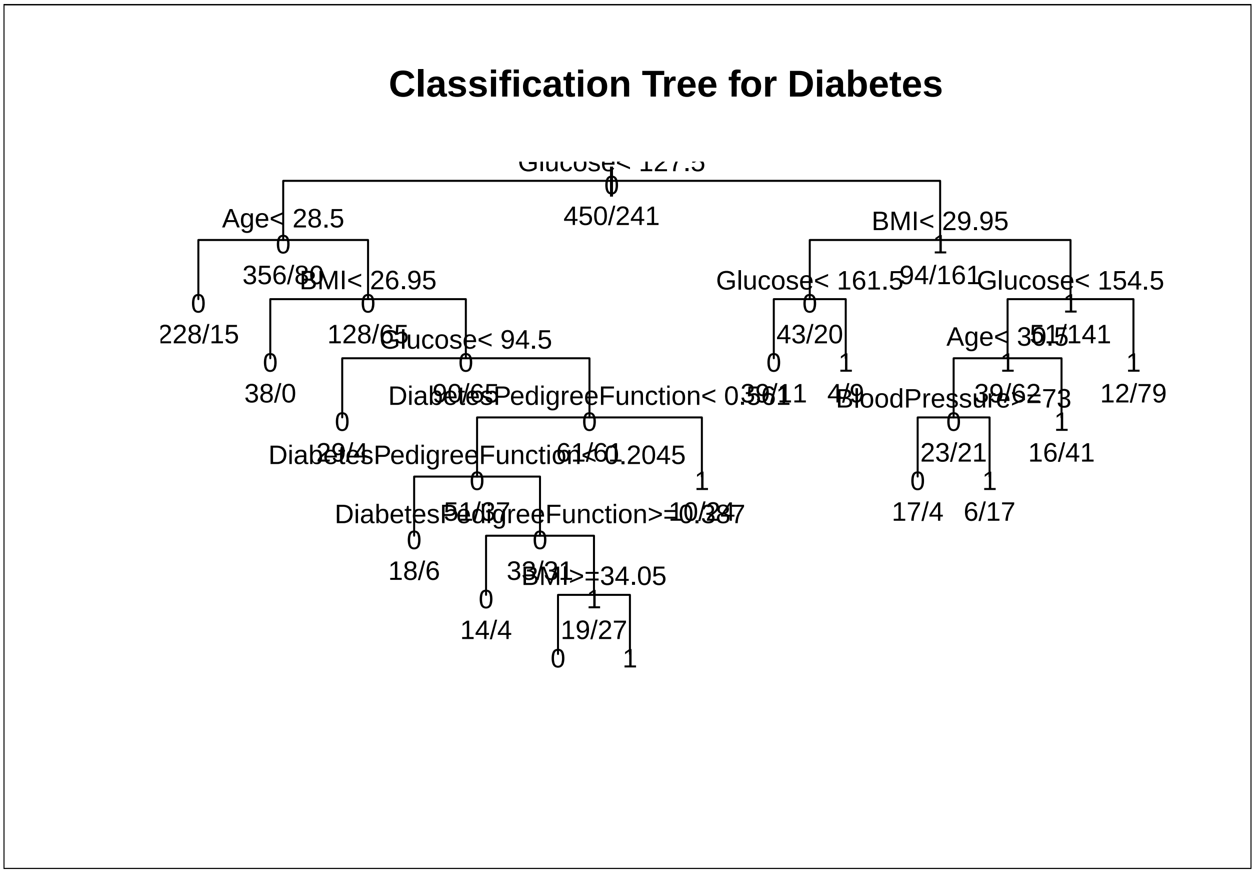
Let us try Decision Tree model on our dataset.

```
# Create decision tree model
decision_tree_model <- rpart(Outcome ~ . - SkinThickness - Insulin, data=train_set,method="class")

# Plot the tree
plot(decision_tree_model, uniform=TRUE, main="Classification Tree for Diabetes")

text(decision_tree_model, use.n=TRUE, all=TRUE, cex=.8)

box(which = "outer", lty = "solid")
```



3.3.1 Cross validation

Let us test the model against test dataset and check the model accuracy. Decision tree model did not yield better accuracy than Linear regression and Logistic regression models.

```
decision_tree_prediction <- predict(decision_tree_model, test_set, type = 'class')
(confusion_matrix_decision_tree<-table(decision_tree_prediction, test_set$Outcome))
```

```
##
## decision_tree_prediction  0  1
##                         0 41 15
##                         1  9 12
```

```
# Find model accuracy
decision_tree_accuracy <- mean(decision_tree_prediction == test_set$Outcome)

# Print the results
results <- bind_rows(results, tibble(Method = "Model 3: Decision Tree", Accuracy = decision_tree_accura
results %>%
  kbl(caption="MODEL PERFORMANCE RESULTS", digits=4) %>%
  kable_classic_2(full_width = T, latex_options = "hold_position")
```

Table 4: MODEL PERFORMANCE RESULTS

Method	Accuracy
Model 1: Multi Linear Regression Model	0.7532
Model 2: Logistic Regression	0.7532
Model 3: Decision Tree	0.6883

3.4 XGBoost

Extreme Gradient Boosting (XGBoost) is one of the Boosting technique in machine learning that has been shown to produce models with high predictive accuracy.

```
#define predictor and response variables in training set
train_x = data.matrix(train_set[,-9])
train_y = as.matrix(train_set[,9])

#define predictor and response variables in testing set
test_x = data.matrix(test_set[, -9])
test_y = as.matrix(test_set[, 9])

#define final training and testing sets
xgb_train = xgb.DMatrix(data = train_x, label = train_y)
xgb_test = xgb.DMatrix(data = test_x, label = test_y)

#define watchlist
watchlist = list(train=xgb_train, test=xgb_test)

#fit XGBoost model and display training and testing data at each round
model = xgb.train(data = xgb_train, max.depth = 3, watchlist=watchlist, nrounds = 70)
```

```
## [1]  train-rmse:0.440839 test-rmse:0.473975
## [2]  train-rmse:0.407452 test-rmse:0.461621
## [3]  train-rmse:0.385690 test-rmse:0.461263
## [4]  train-rmse:0.370349 test-rmse:0.457088
## [5]  train-rmse:0.360050 test-rmse:0.453717
## [6]  train-rmse:0.351687 test-rmse:0.453215
## [7]  train-rmse:0.346246 test-rmse:0.451741
## [8]  train-rmse:0.340312 test-rmse:0.448894
## [9]  train-rmse:0.334845 test-rmse:0.453818
## [10] train-rmse:0.331112 test-rmse:0.453075
## [11] train-rmse:0.325774 test-rmse:0.453872
## [12] train-rmse:0.323575 test-rmse:0.451254
## [13] train-rmse:0.320716 test-rmse:0.453884
## [14] train-rmse:0.316676 test-rmse:0.456031
## [15] train-rmse:0.314068 test-rmse:0.455481
## [16] train-rmse:0.312237 test-rmse:0.455180
## [17] train-rmse:0.310615 test-rmse:0.456795
## [18] train-rmse:0.306387 test-rmse:0.455613
## [19] train-rmse:0.303806 test-rmse:0.458489
## [20] train-rmse:0.302132 test-rmse:0.458906
## [21] train-rmse:0.299457 test-rmse:0.458679
## [22] train-rmse:0.296040 test-rmse:0.457977
## [23] train-rmse:0.294741 test-rmse:0.458179
## [24] train-rmse:0.293785 test-rmse:0.459462
## [25] train-rmse:0.291294 test-rmse:0.460096
## [26] train-rmse:0.290404 test-rmse:0.460212
## [27] train-rmse:0.288078 test-rmse:0.462193
## [28] train-rmse:0.284648 test-rmse:0.462962
## [29] train-rmse:0.282121 test-rmse:0.461553
## [30] train-rmse:0.280600 test-rmse:0.462426
## [31] train-rmse:0.278472 test-rmse:0.462785
## [32] train-rmse:0.275987 test-rmse:0.464976
## [33] train-rmse:0.273212 test-rmse:0.465918
## [34] train-rmse:0.272128 test-rmse:0.465326
## [35] train-rmse:0.269446 test-rmse:0.466639
## [36] train-rmse:0.267376 test-rmse:0.467791
## [37] train-rmse:0.266737 test-rmse:0.466464
## [38] train-rmse:0.263883 test-rmse:0.464453
## [39] train-rmse:0.261399 test-rmse:0.463930
## [40] train-rmse:0.258791 test-rmse:0.464994
## [41] train-rmse:0.257019 test-rmse:0.465242
## [42] train-rmse:0.255284 test-rmse:0.465506
## [43] train-rmse:0.253583 test-rmse:0.467238
## [44] train-rmse:0.251826 test-rmse:0.467352
## [45] train-rmse:0.251345 test-rmse:0.466013
## [46] train-rmse:0.249414 test-rmse:0.465174
## [47] train-rmse:0.248102 test-rmse:0.467093
## [48] train-rmse:0.244814 test-rmse:0.468264
## [49] train-rmse:0.242760 test-rmse:0.468483
## [50] train-rmse:0.241044 test-rmse:0.469389
## [51] train-rmse:0.240333 test-rmse:0.468980
## [52] train-rmse:0.238594 test-rmse:0.469344
## [53] train-rmse:0.237652 test-rmse:0.469240
## [54] train-rmse:0.236747 test-rmse:0.468975
## [55] train-rmse:0.236143 test-rmse:0.467128
## [56] train-rmse:0.233656 test-rmse:0.467326
## [57] train-rmse:0.232020 test-rmse:0.467729
## [58] train-rmse:0.231135 test-rmse:0.466318
## [59] train-rmse:0.229992 test-rmse:0.465008
## [60] train-rmse:0.228856 test-rmse:0.464639
## [61] train-rmse:0.227500 test-rmse:0.464865
## [62] train-rmse:0.225129 test-rmse:0.466394
## [63] train-rmse:0.224755 test-rmse:0.466547
## [64] train-rmse:0.223430 test-rmse:0.469109
## [65] train-rmse:0.221614 test-rmse:0.473023
## [66] train-rmse:0.221371 test-rmse:0.473038
## [67] train-rmse:0.220894 test-rmse:0.473198
## [68] train-rmse:0.219665 test-rmse:0.473110
## [69] train-rmse:0.217128 test-rmse:0.472918
## [70] train-rmse:0.215713 test-rmse:0.471472
```

From the output we can see that the minimum test-RMSE is achieved at 24 rounds. Beyond this point, the test-RMSE begins to increase, which is a sign that we are overfitting the training data.

So, we will define our final XGBoost model to use 24 rounds.

```
#define final model
final_xgboost = xgboost(data = xgb_train, max.depth = 3, nrounds = 24, verbose = 0)
```

3.4.1 Cross validation

Let us test the model against test dataset and check the model accuracy.

```
xgboost_prediction <- predict(final_xgboost, xgb_test)
xgboost_prediction <- ifelse(xgboost_prediction > 0.5,1,0)
```

```
# Find model accuracy
xgboost_accuracy <- mean(xgboost_prediction == test_set$Outcome)

# Print the results
results <- bind_rows(results, tibble(Method = "Model 4: XGBoost", Accuracy = xgboost_accuracy))
results %>%
  kbl(caption="MODEL PERFORMANCE RESULTS", digits=4) %>%
  kable_classic_2(full_width = T, latex_options = "hold_position")
```

Table 5: MODEL PERFORMANCE RESULTS

Method	Accuracy
Model 1: Multi Linear Regression Model	0.7532
Model 2: Logistic Regression	0.7532
Model 3: Decision Tree	0.6883
Model 4: XGBoost	0.7143

4 Results

Accuracy of Linear Regression and Logistic Regression models are the highest among the four machine learning models we tried our dataset on.

```
# Print the results
results %>%
  kbl(caption="MODEL PERFORMANCE RESULTS", digits=4) %>%
  kable_classic_2(full_width = T, latex_options = "hold_position")
```

Table 6: MODEL PERFORMANCE RESULTS

Method	Accuracy
Model 1: Multi Linear Regression Model	0.7532
Model 2: Logistic Regression	0.7532
Model 3: Decision Tree	0.6883
Model 4: XGBoost	0.7143

5 Conclusion

We started with the Multi Linear Regression model and tried three other models Logistic Regression, Decision Tree, and XGBoost. The Multi Linear Regression and Logistic Regression models performed better than the Decision Tree and XGBoost. The accuracy of Multi Linear Regression and Logistic model was 75.3246753% followed by XGBoost with 71.4285714% accuracy and the decision tree with the least accuracy of 68.8311688%.

6 Limitations

Insulin and SkinThickness variables were missing data. Our models could have predicted the Outcome more accurately if we had data on these variables. The algorithms' precision increases when the dataset's size is large. Hence, more data will make the model more accurate in predicting if a person has diabetes.

7 Future Work

As with any such project, there is always room for improvement. As part of this project, we trained and evaluated only four predictive models on the dataset. We could train more predictive models and improve their performance with hyperparameter tuning. Also, the models' performance increases when the size of the dataset increases. So having more data will make the model more accurate in predicting if a person has diabetes.

8 Acknowledgments

I want to thank the instructor, Rafael A. Irizarry of HarvardX's Professional Certificate in Data Science, for the detailed and clear explanation in his lectures throughout the series. I also want to thank and appreciate the staff and peers for providing their support, help, and guidance in the discussion forums.

9 Reference

- Diabetes Basics | CDC
- Diabetes Dataset | Kaggle
- Median Imputaion
- Decision Tree
- Binary Classification Models