Diabetes Risk Prediction - CYO Project

Poonam Quraishy

2/22/2022

Contents

Overview **1**

Introduction **2**

Dataset **2**

Exploratory Analysis **2**

Modeling Approach **17**

Model 1 - Logistic Regression. 18

Model 2 - Classification Tree. 18

Model 3 - XGBoost - eXtreme Gradient Boosting 22

Model 4 - K Nearest Neighbors (KNN) 23

Model 5 - Support vector machine (SVM) 25

Comparing the Test results of all the models. **31**

Results **31**

Comparing the accuracy of all the models **31**

Conclusion **32**

Enviroment **32**

Overview

This project is the Choose your own section of the Data Science Capstone offered by HarvardX. The aim of this project is to apply machine learning techniques on any chosen dataset and present the insights and analysis in a structured report.

Introduction

Diabetes mellitus is a chronic health condition that occurs when the pancreas do not produce enough insulin or the body cannot efficiently use the insulin it produces. Chronic diabeteic conditions include Type 1 diabetes, Type 2 diabetes, prediabetes, and gestational diabetes. The incidence and prevalence of diabetes mellitus is rapidly growing and has already affected 422 million people as stated by a report by the World Health Organization (WHO), in 2018. According to the World Health Organization, diabetes can be treated and its consequences avoided or delayed with diet, physical activity, medication, regular screening, and treatment for complications. Early detection of diabetes is ideally desired for a clinically meaningful outcome. Diabetes has a relatively long asymptomatic phase which poses challenges to early detection and diagnosis. This project attempts to create multiple machine learning models to predict the risk of developing diabetes. The modeling techniques performed on this dataset include Logistic Regression, Classification Trees, eXtreme Gradient Boosing (XGBoost), K - Nearest Neighbors, and Support Vector Machines (SVM).

Dataset

The dataset used for this project is the Pima Indians Diabetes dataset available in the mLbench package. This dataset contains 786 observations and 9 variables. The observations are PIMA Indian females near Pheonix Arizona. The 9 variables are as follows - 1.Pregnant 2.Glucose 3.Pressure 4.Triceps 5.Insulin 6.Mass 7.Pedigree 8.Age 9.Diabetes

'Diabetes' will be the response/target variable. The diabetes variable contains 500 negative and 268 positive outcomes which indicate if the person was diagnosed with diabetes or not. The data set will be split into an 80-20 train and test data set.

Exploratory Analysis

The Pima Indians Diabetes dataset is available in the package mLbench.

```
## 'data.frame':    768 obs. of  9 variables:
##  $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
##  $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
##  $ pressure: num  72 66 64 66 40 74 50 0 70 96 ...
##  $ triceps : num  35 29 0 23 35 0 32 0 45 0 ...
##  $ insulin : num  0 0 0 94 168 0 88 0 543 0 ...
##  $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
##  $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

First few rows

pregnant	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
6	148	72	35	0	33.6	0.627	50	pos
1	85	66	29	0	26.6	0.351	31	neg
8	183	64	0	0	23.3	0.672	32	pos
1	89	66	23	94	28.1	0.167	21	neg
0	137	40	35	168	43.1	2.288	33	pos
5	116	74	0	0	25.6	0.201	30	neg

Names of the columns

```
## [1] "pregnant" "glucose"  "pressure" "triceps"  "insulin"  "mass"     "pedigree"
## [8] "age"      "diabetes"
```

Summary of the dataset

```
##     pregnant         glucose         pressure         triceps
##  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           mass          pedigree           age        diabetes
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00   neg:500
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00   pos:268
##  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
```

Dimensions

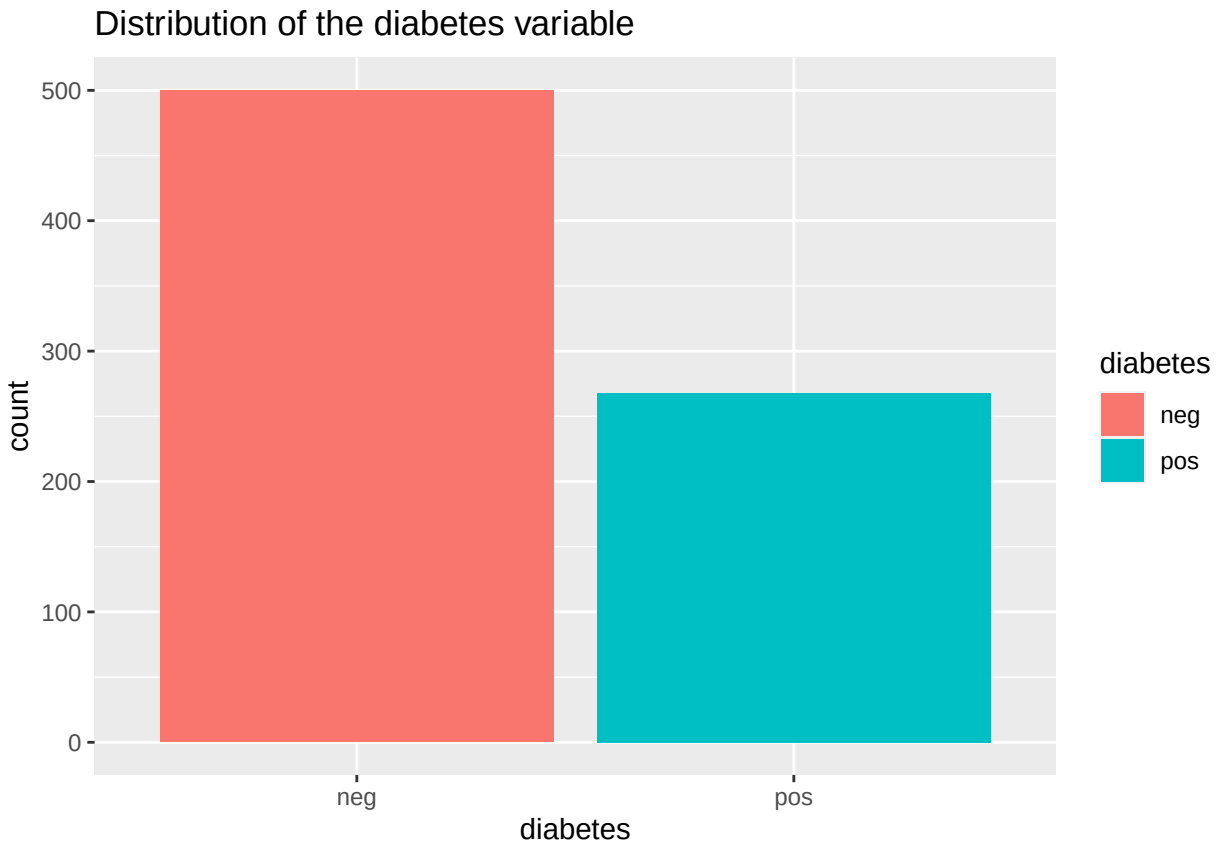
```
## [1] 768   9
```

Check for NA's or missing values

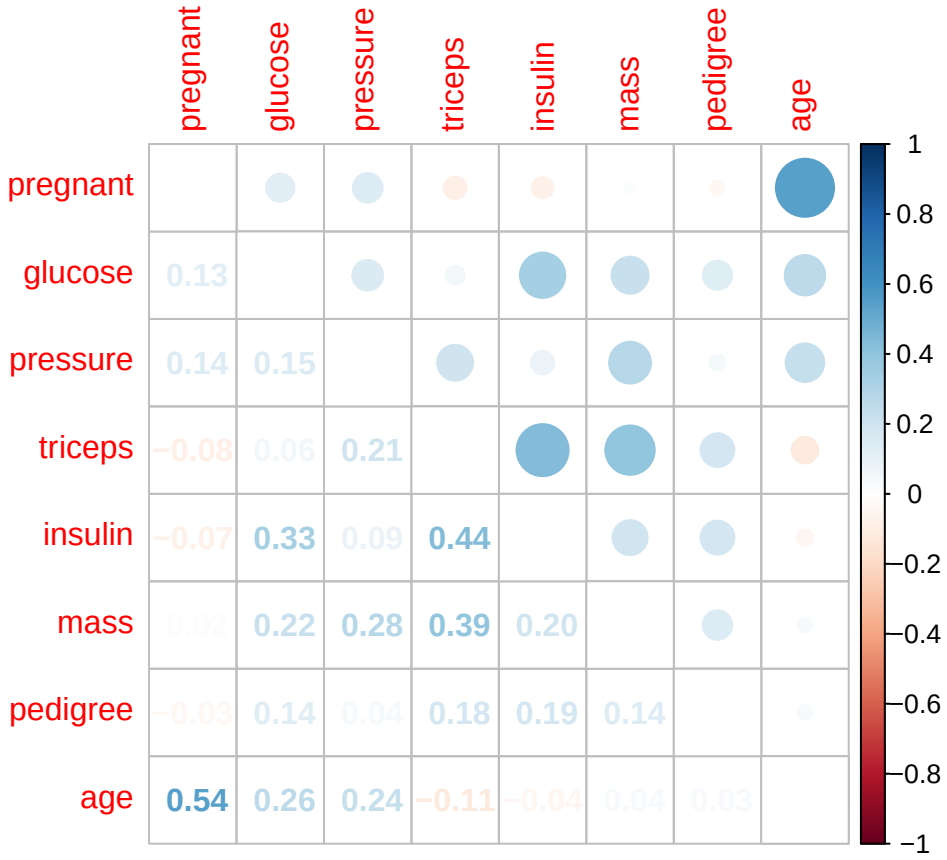
```
## pregnant  glucose pressure  triceps  insulin     mass pedigree      age
##        0        0        0        0        0        0        0        0
## diabetes
##        0
```

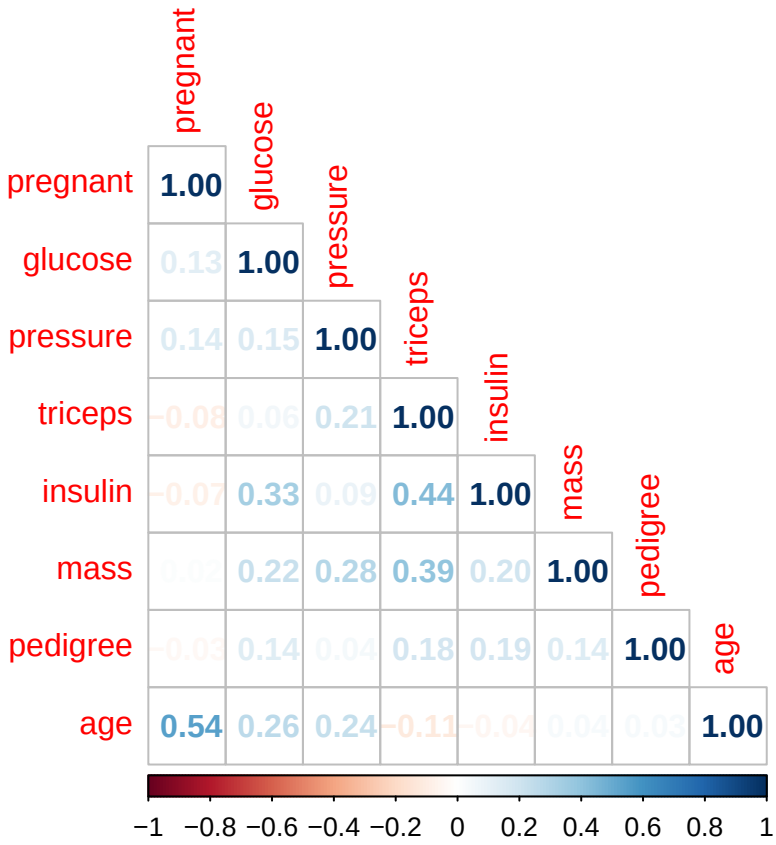
Exploring the response variable Diabetes

Diabetes - The sample has a high occurrence of positive diabetes diagnosis.



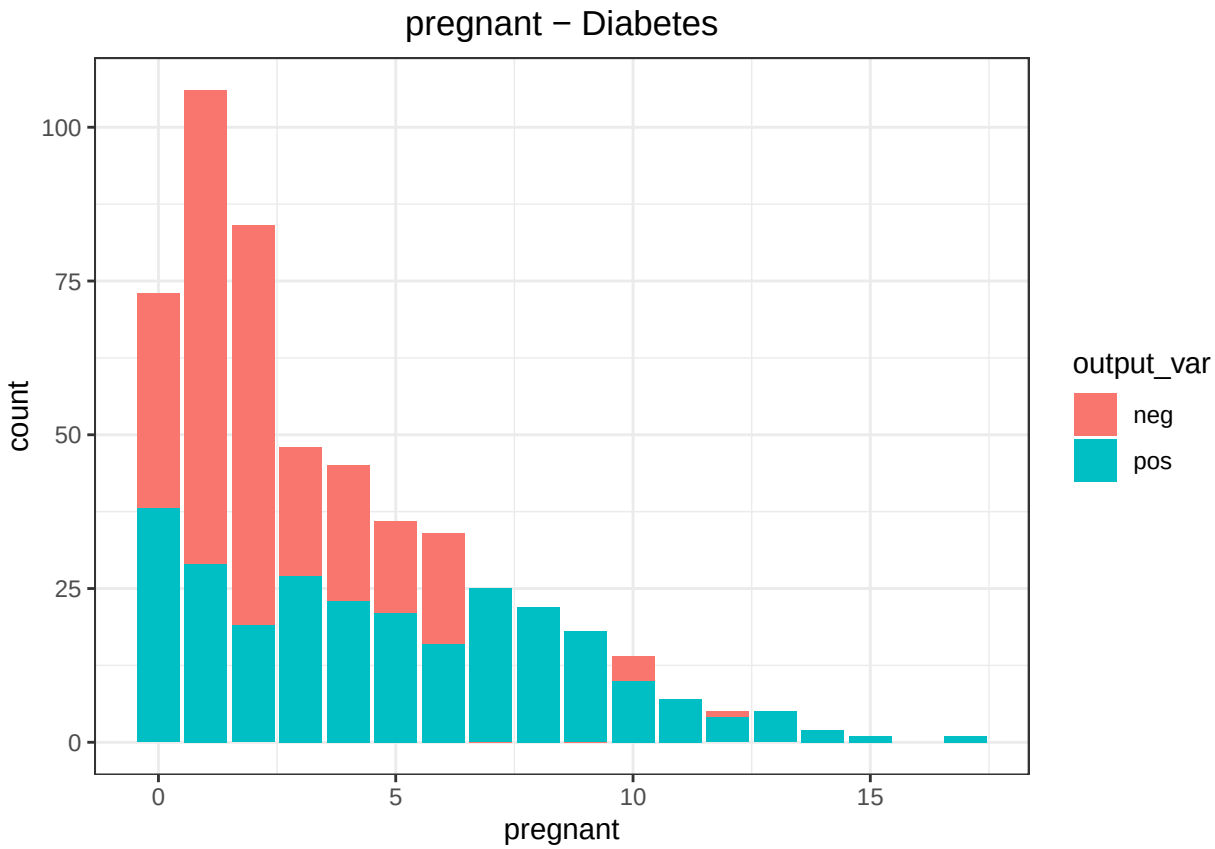
Plot of Correlations between all the predictor variables

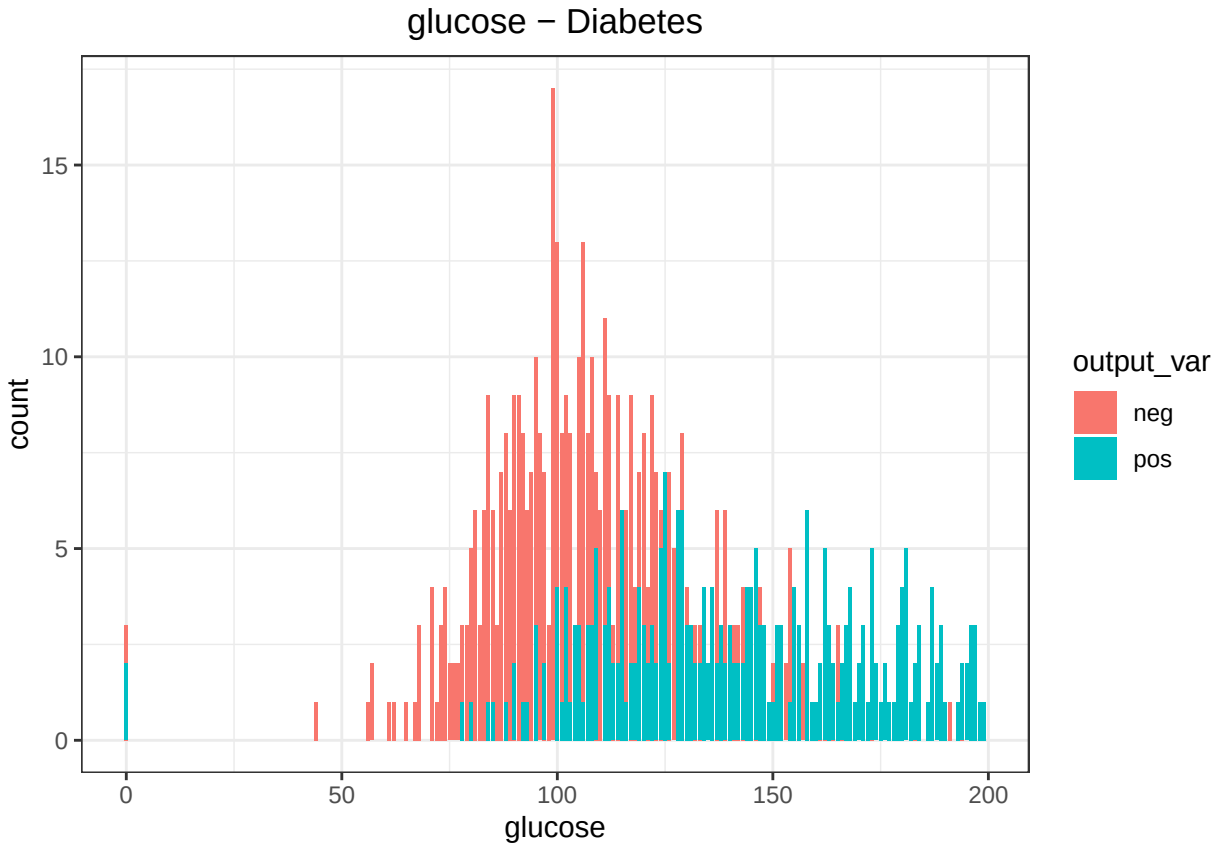

pregnant
glucose
pressure
triceps
insulin
mass
pedigree
age
0.13
0.14 0.15
−0.08 0.06 0.21
−0.07 0.33 0.09 0.44
0.02 0.22 0.28 0.39 0.20
−0.03 0.14 0.04 0.18 0.19 0.14
0.54 0.26 0.24 −0.11 −0.04 0.04 0.03
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

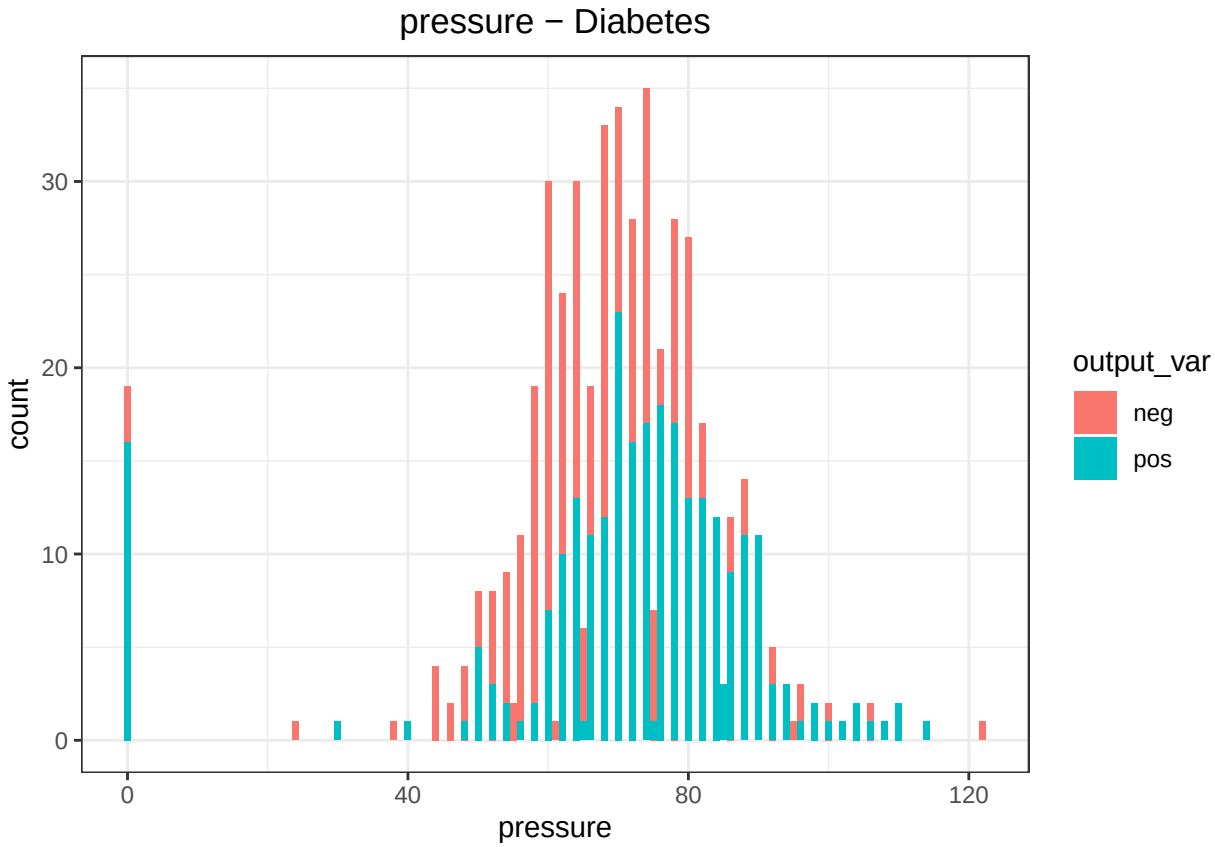


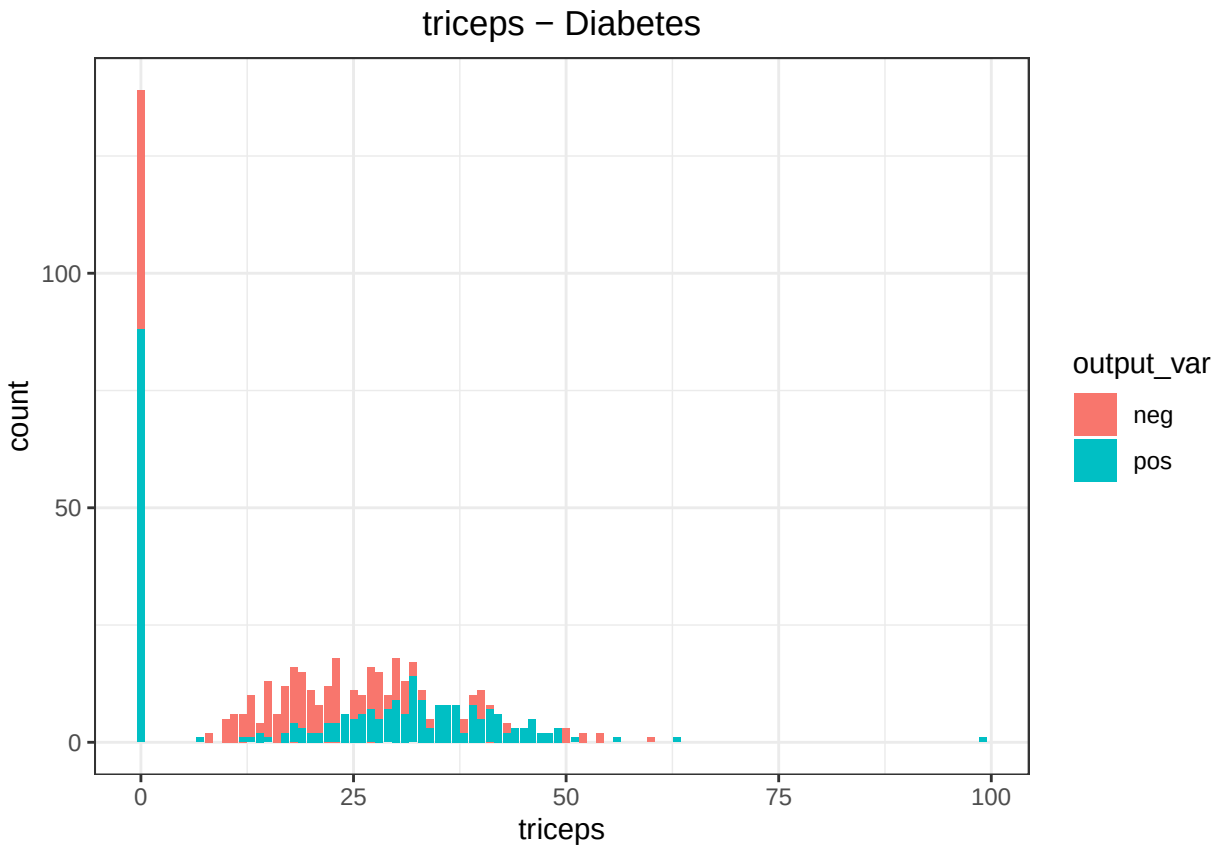
Numerical Representation

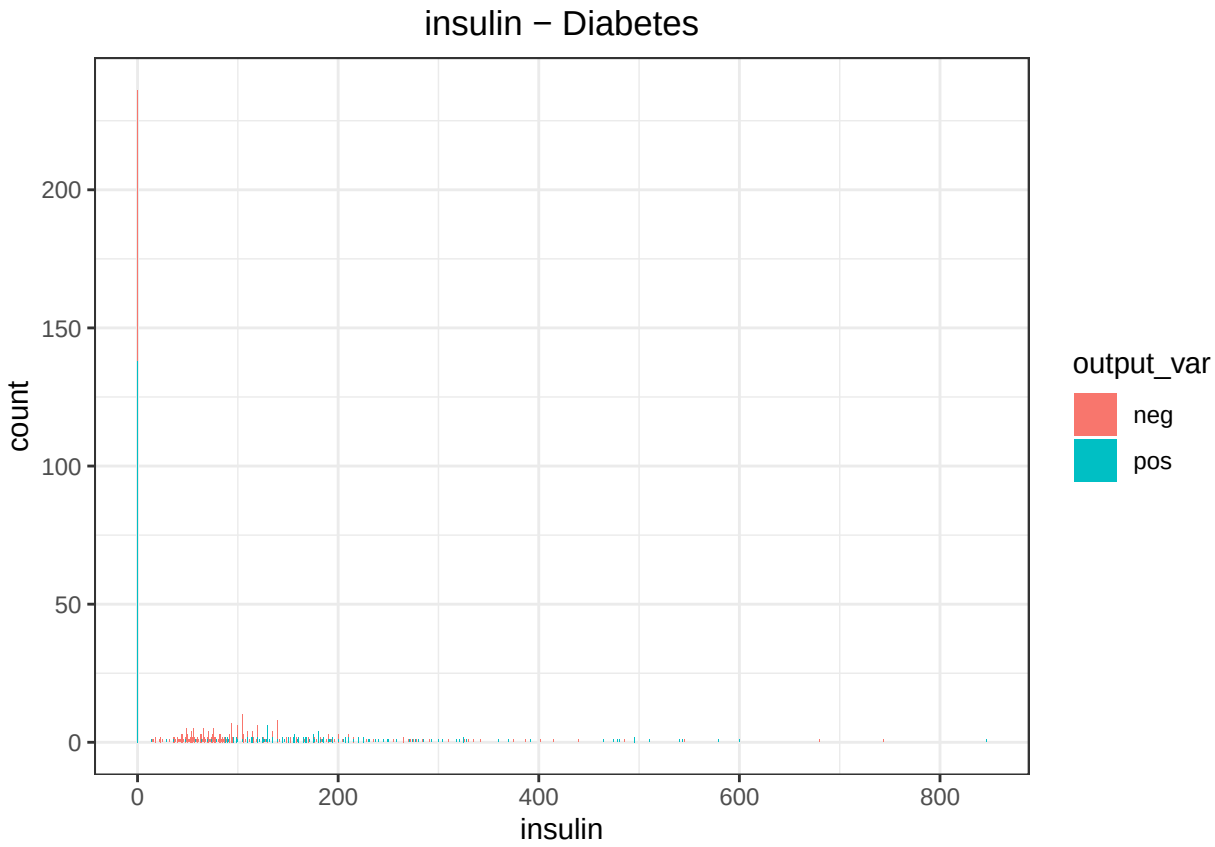
Univariate Analysis

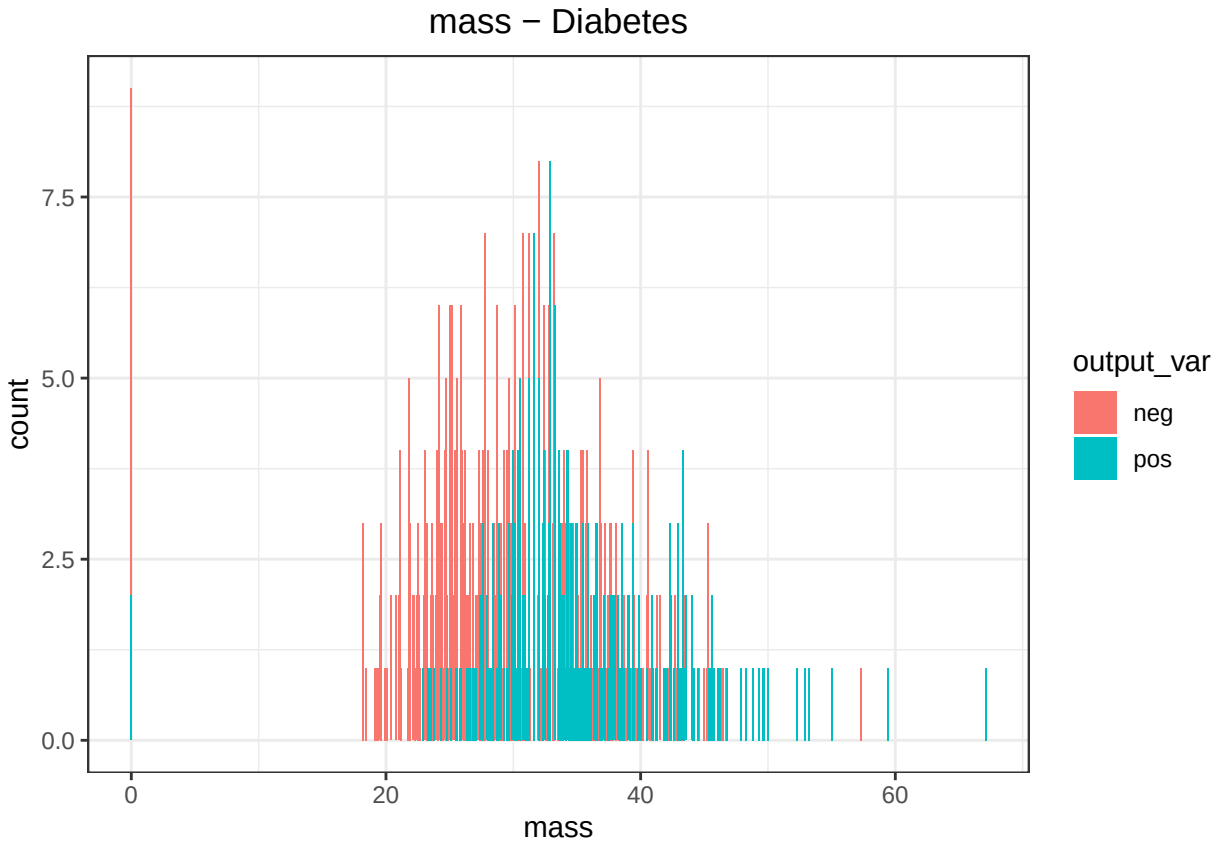

pregnant − Diabetes
count
100
75
50
25
0
0
5
10
15
pregnant
output_var
neg
pos

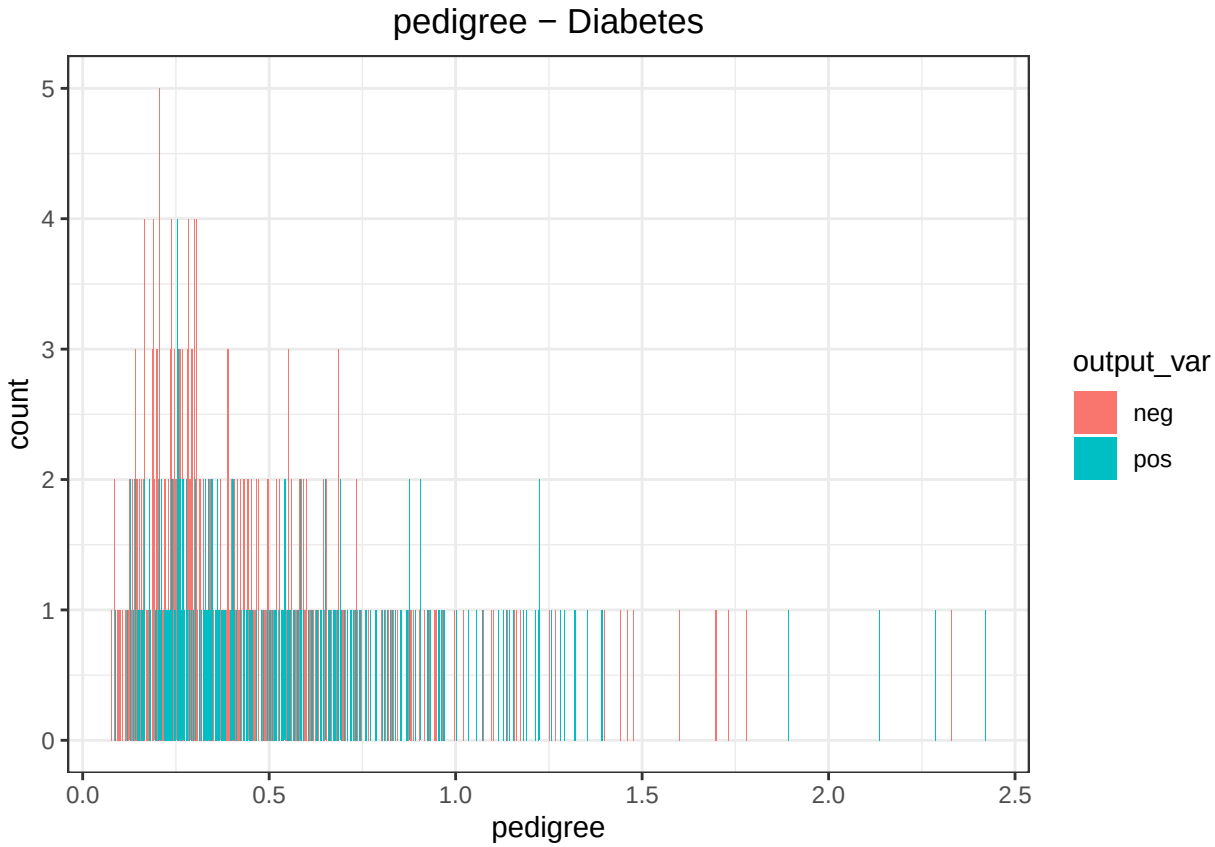

glucose − Diabetes
count
15
10
5
0
0
50
100
150
200
glucose
output_var
neg
pos

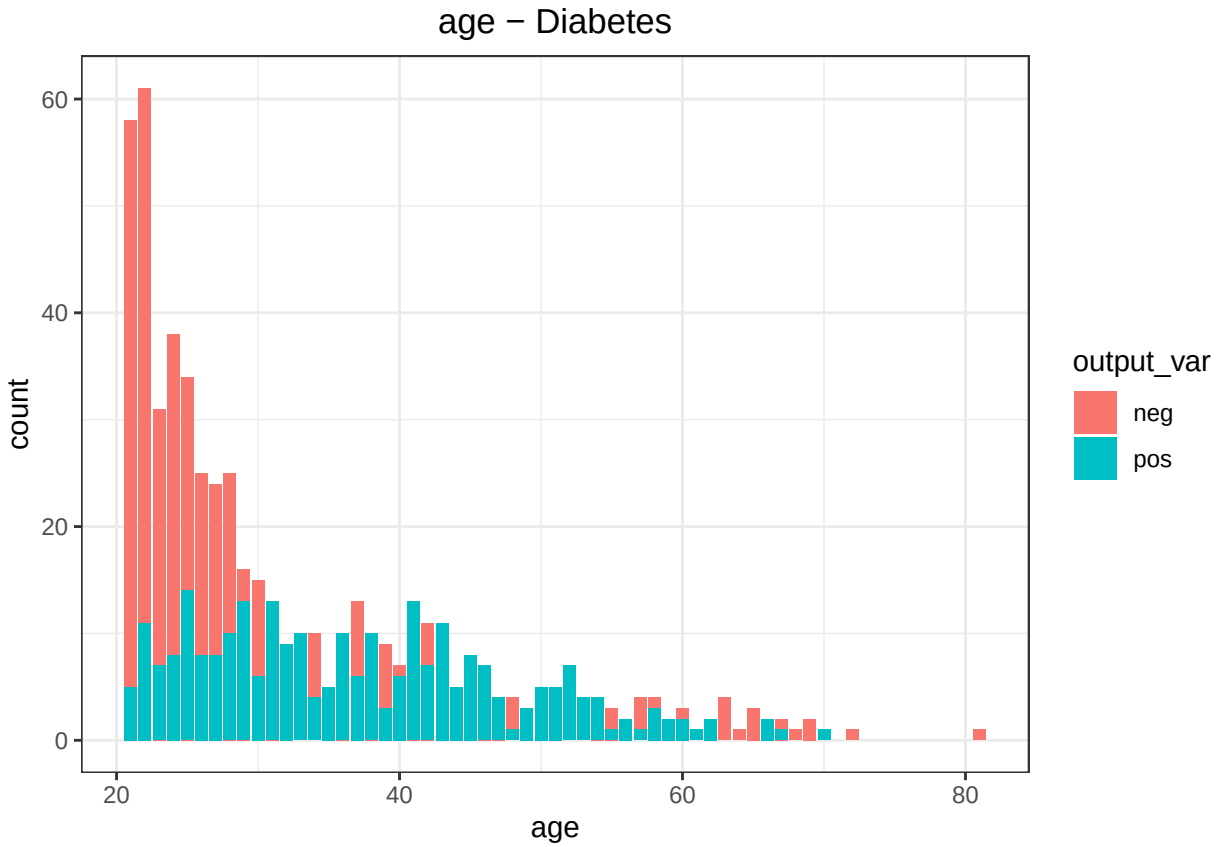

pressure − Diabetes
count
30
20
10
0
0
40
80
120
pressure
output_var
neg
pos


triceps − Diabetes
count
100
50
0
0
25
50
75
100
triceps
output_var
neg
pos


insulin − Diabetes
count
200
150
100
50
0
0
200
400
600
800
insulin
output_var
neg
pos

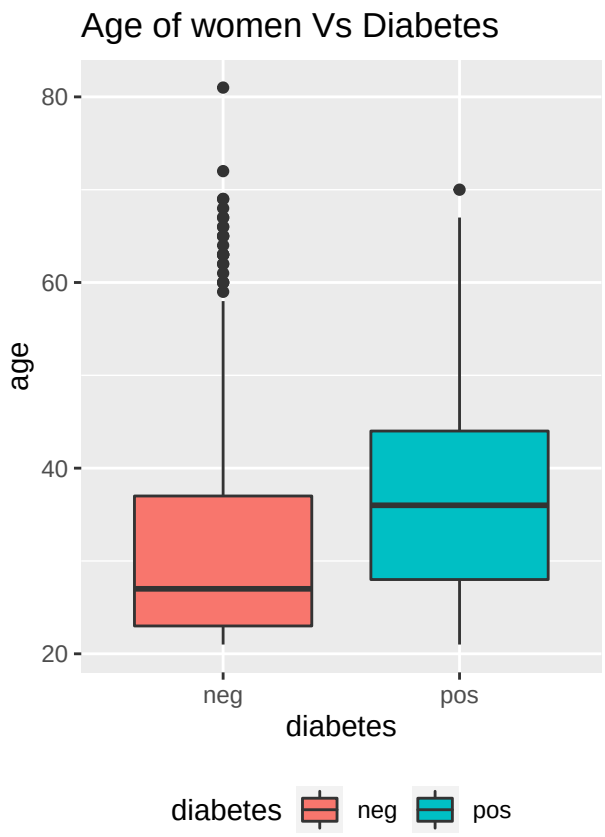

mass − Diabetes
count
7.5
5.0
2.5
0.0
0
20
40
60
mass
output_var
neg
pos

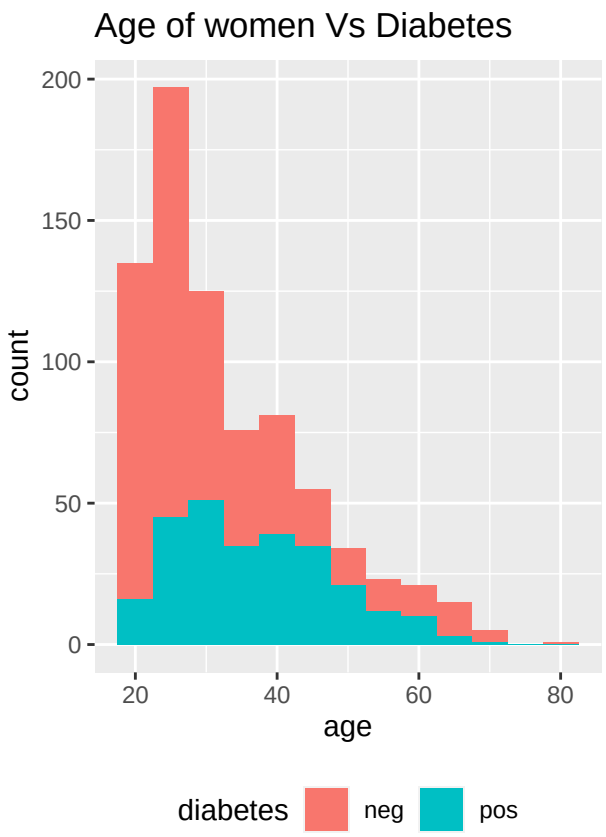

pedigree − Diabetes
count
pedigree
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
2.5
output_var
neg
pos



It is evident that high glucose levels lead to a higher chance of positive diabetes diagnosis. Mass/BMI increases also increase the chance of a positive diabetes diagnosis. Age over the age of 25 also an indicator that increases the chance of a diabetes diagnosis. There is no notable significant distinction among other variables to warrant further exploration.

Exploring the Age variable

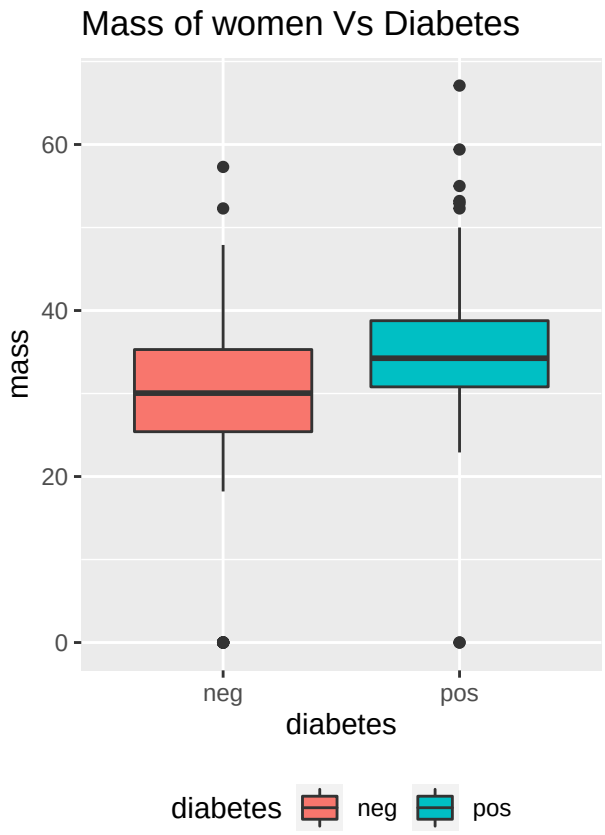


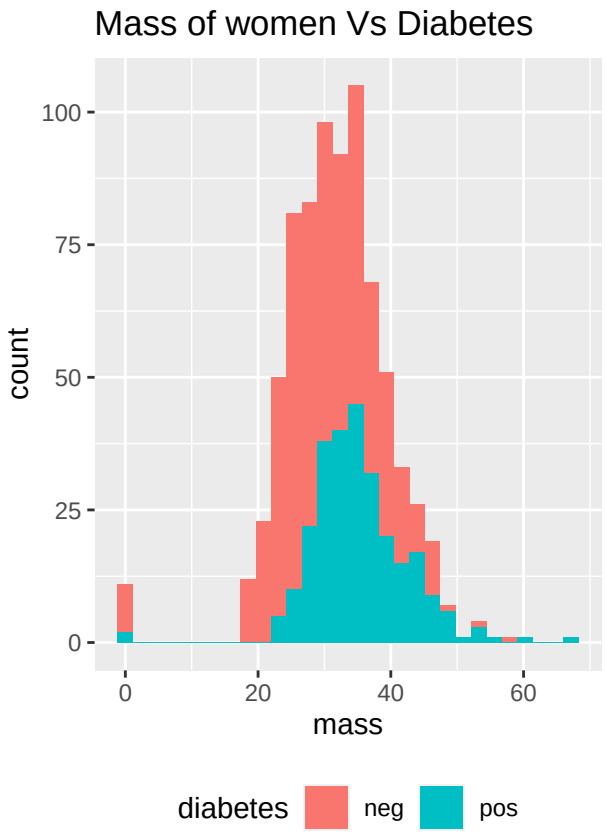


Exploring the mass variable

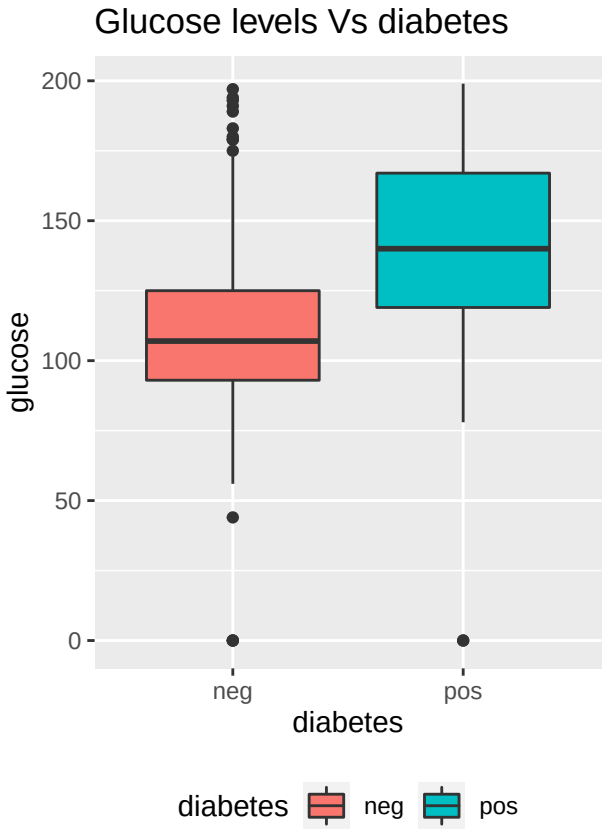
```
## Warning: Ignoring unknown parameters: binwidth
```

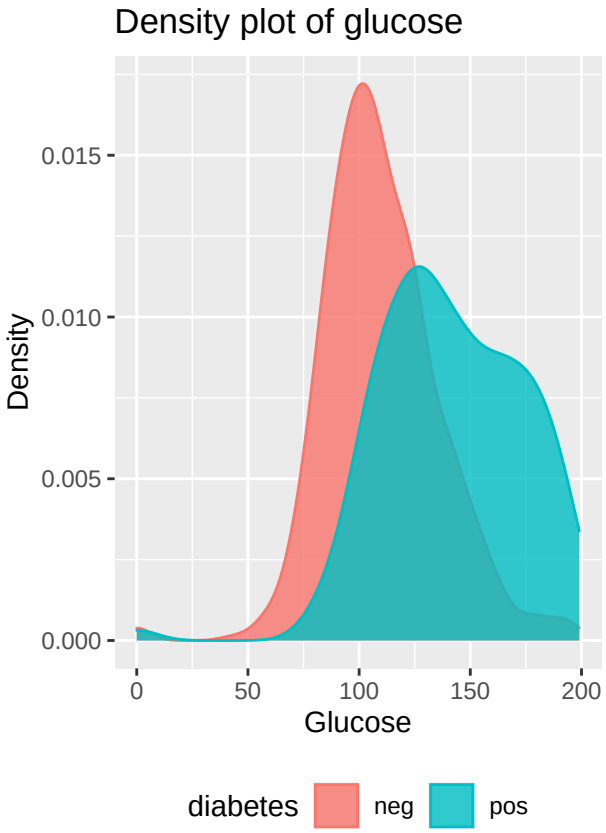
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```





Exploring the Glucose variable





Modeling Approach

Split the Data into Training set consisting of 80% of the data and Testing set consisting of 20% of the data.

Training set and Test set

```
## 'data.frame':    615 obs. of  9 variables:
##  $ pregnant: num  6 1 8 1 0 5 3 2 8 4 ...
##  $ glucose : num  148 85 183 89 137 116 78 197 125 110 ...
##  $ pressure: num  72 66 64 66 40 74 50 70 96 92 ...
##  $ triceps : num  35 29 0 23 35 0 32 45 0 0 ...
##  $ insulin : num  0 0 0 94 168 0 88 543 0 0 ...
##  $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 30.5 0 37.6 ...
##  $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age     : num  50 31 32 21 33 30 26 53 54 30 ...
##  $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 2 2 1 ...
```

```
##     pregnant         glucose         pressure         triceps
##  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   : 3.782   Mean   :121.6   Mean   : 69.54   Mean   :20.67
##  3rd Qu.: 6.000   3rd Qu.:142.0   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
```

```
##     insulin           mass          pedigree           age        diabetes
##  Min.   :  0.00   Min.   : 0.00   Min.   :0.0780   Min.   :21.00   neg:400
##  1st Qu.:  0.00   1st Qu.:27.30   1st Qu.:0.2385   1st Qu.:24.00   pos:215
##  Median : 29.00   Median :32.00   Median :0.3640   Median :29.00
##  Mean   : 79.89   Mean   :32.02   Mean   :0.4624   Mean   :33.44
##  3rd Qu.:126.00   3rd Qu.:36.75   3rd Qu.:0.6215   3rd Qu.:41.00
##  Max.   :846.00   Max.   :59.40   Max.   :2.4200   Max.   :81.00

##     pregnant         glucose         pressure         triceps
##  Min.   : 0.000   Min.   : 44.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.: 0.00
##  Median : 3.000   Median :114.0   Median : 70.00   Median :23.00
##  Mean   : 4.098   Mean   :117.9   Mean   : 67.37   Mean   :19.98
##  3rd Qu.: 6.000   3rd Qu.:136.0   3rd Qu.: 78.00   3rd Qu.:32.00
##  Max.   :14.000   Max.   :197.0   Max.   :110.00   Max.   :52.00
##     insulin           mass          pedigree           age        diabetes
##  Min.   :  0.00   Min.   : 0.00   Min.   :0.0840   Min.   :21.00   neg:100
##  1st Qu.:  0.00   1st Qu.:27.10   1st Qu.:0.2590   1st Qu.:25.00   pos: 53
##  Median : 37.00   Median :32.30   Median :0.4110   Median :29.00
##  Mean   : 79.45   Mean   :31.87   Mean   :0.5101   Mean   :32.42
##  3rd Qu.:135.00   3rd Qu.:35.50   3rd Qu.:0.6470   3rd Qu.:38.00
##  Max.   :540.00   Max.   :67.10   Max.   :2.1370   Max.   :72.00
```

Model 1 - Logistic Regression.

Logistic Regression is an introductory classification algorithm used to find the probability of event success and event failure. Logistic regression is used when the dependent variable is binary in nature.

```
glm_model <- caret::train(diabetes ~., data = train_set,
                          method = "glm",
                          metric = "ROC",
                          tuneLength = 10,
                          trControl = trainControl(method = "cv", number = 10,
                                                   classProbs = T, summaryFunction = twoClassSummary),
                          preProcess = c("center","scale","pca"))
```

Final ROC for the Logistic Regression

```
## [1] 0.8308225
```

Model 2 - Classification Tree.

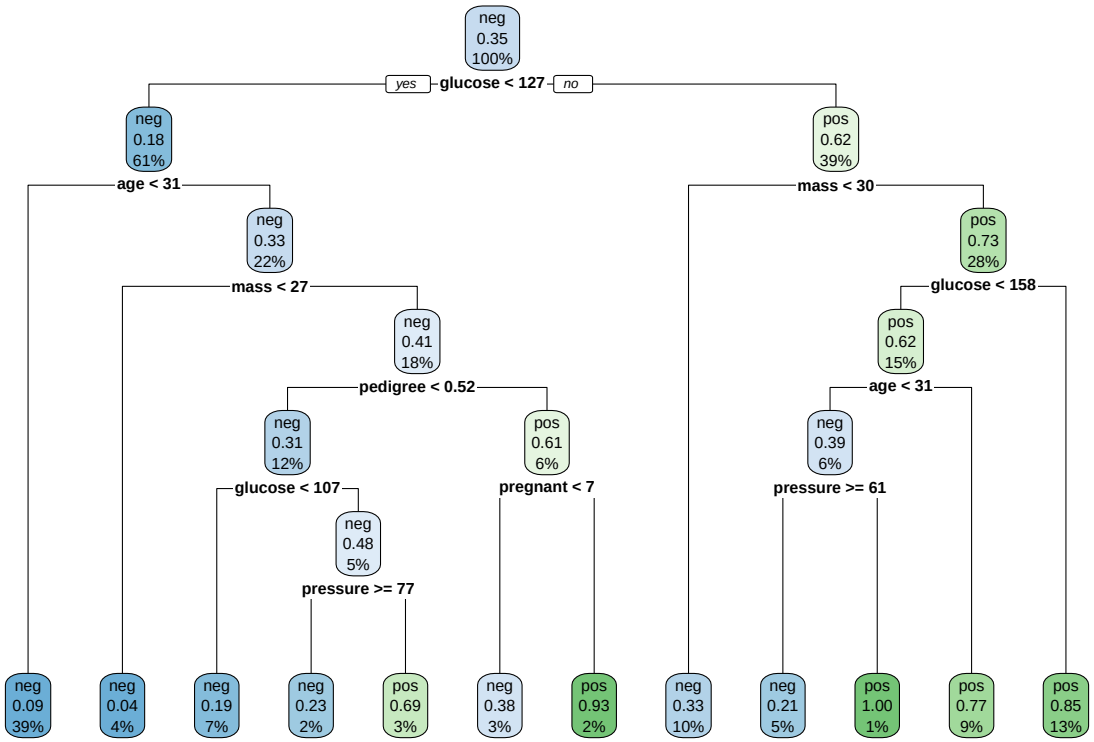
A classification tree is used for modeling data when the response variable is categorical. The tree splits the data into two or more homogeneous sets based on the most significant differentiator in the predictor variables-value set. We will build a classification tree using the binary response variable 'diabetes'.

```
tree_model <- rpart(diabetes~., data=train_set, method="class")
tree_model
```

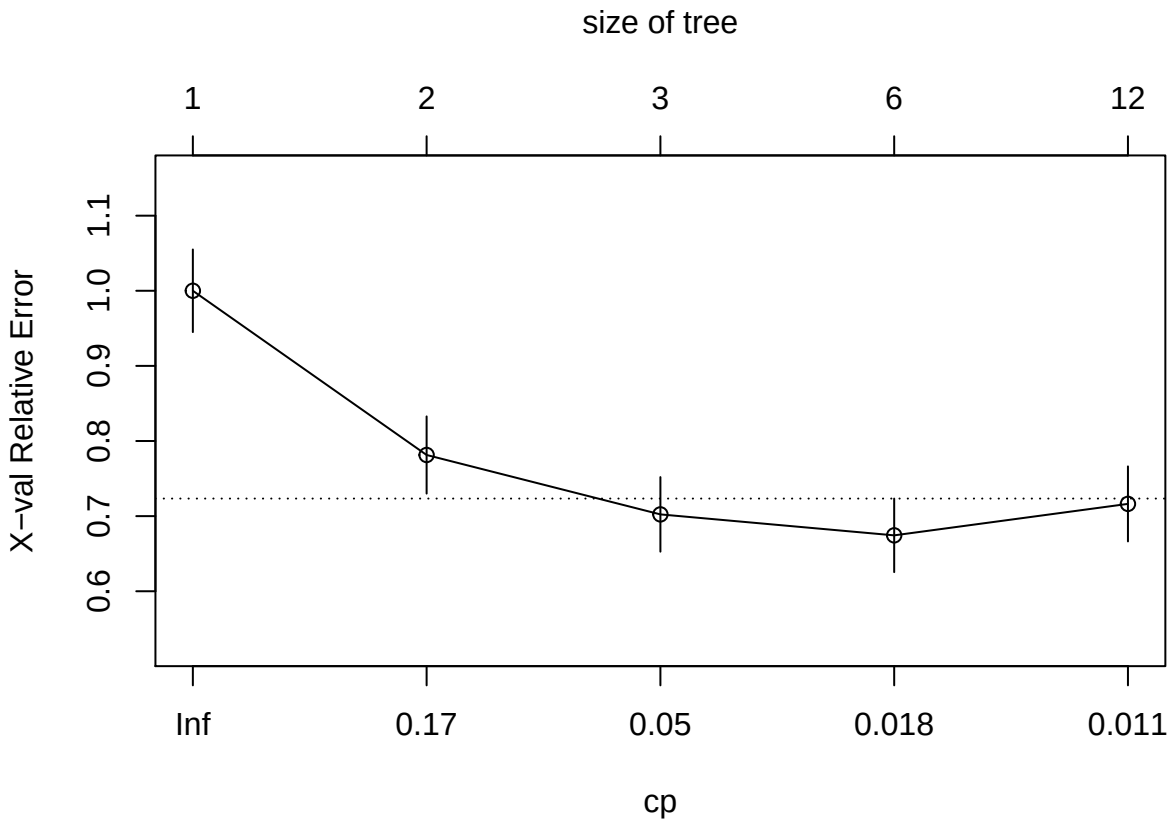
```
## n= 615
##
```

```
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 615 215 neg (0.65040650 0.34959350)
##    2) glucose< 127 377  67 neg (0.82228117 0.17771883)
##      4) age< 30.5 242  22 neg (0.90909091 0.09090909) *
##      5) age>=30.5 135  45 neg (0.66666667 0.33333333)
##       10) mass< 26.8 27   1 neg (0.96296296 0.03703704) *
##       11) mass>=26.8 108  44 neg (0.59259259 0.40740741)
##         22) pedigree< 0.5205 72  22 neg (0.69444444 0.30555556)
##           44) glucose< 106.5 43   8 neg (0.81395349 0.18604651) *
##           45) glucose>=106.5 29  14 neg (0.51724138 0.48275862)
##             90) pressure>=77 13   3 neg (0.76923077 0.23076923) *
##             91) pressure< 77 16   5 pos (0.31250000 0.68750000) *
##         23) pedigree>=0.5205 36  14 pos (0.38888889 0.61111111)
##           46) pregnant< 6.5 21   8 neg (0.61904762 0.38095238) *
##           47) pregnant>=6.5 15   1 pos (0.06666667 0.93333333) *
##    3) glucose>=127 238  90 pos (0.37815126 0.62184874)
##      6) mass< 29.95 64  21 neg (0.67187500 0.32812500) *
##      7) mass>=29.95 174  47 pos (0.27011494 0.72988506)
##       14) glucose< 157.5 93  35 pos (0.37634409 0.62365591)
##         28) age< 30.5 36  14 neg (0.61111111 0.38888889)
##           56) pressure>=61 28   6 neg (0.78571429 0.21428571) *
##           57) pressure< 61 8   0 pos (0.00000000 1.00000000) *
##         29) age>=30.5 57  13 pos (0.22807018 0.77192982) *
##       15) glucose>=157.5 81  12 pos (0.14814815 0.85185185) *
```

```
rpart.plot(tree_model)
```



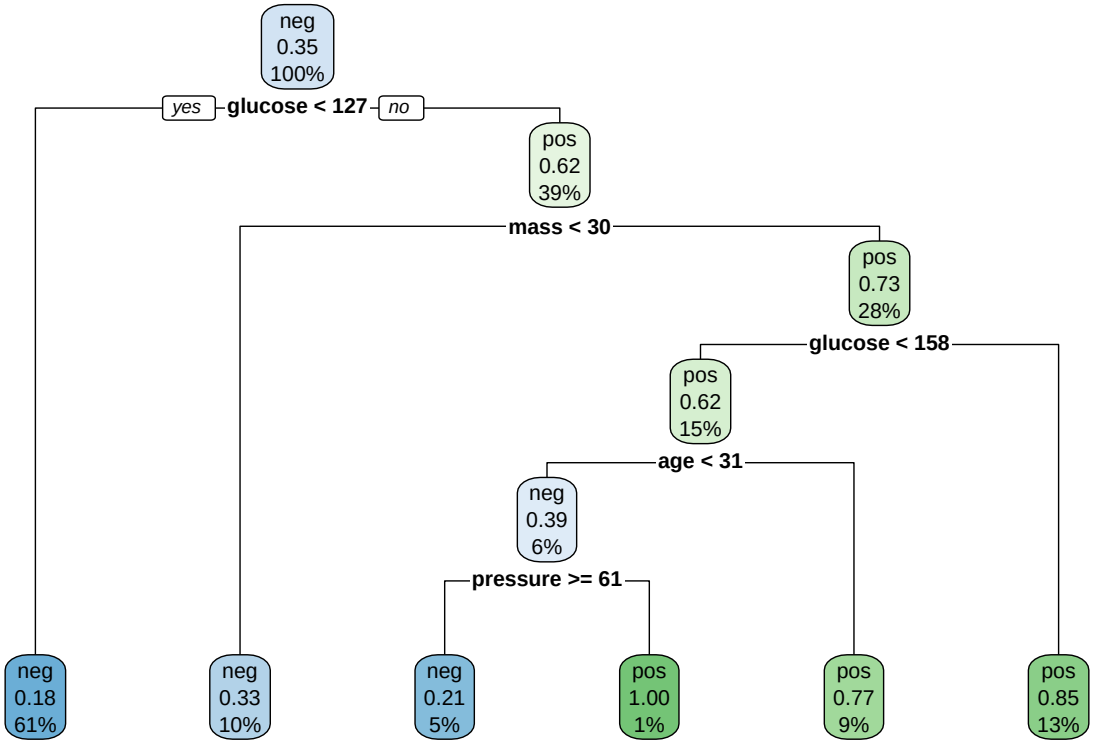
Best Model Complexity Parameter - CP



The complexity parameter value of 0.015 was chosen since the relative error does not decrease significantly after this value.

Model after Pruning the tree

```
tree_model <- rpart(diabetes~., data=train_set, method="class",cp=0.015)
rpart.plot(tree_model)
```



Model 3 - XGBoost - eXtreme Gradient Boosting

eXtreme Gradient Boosting Machine (XGBoost) is a popular machine learning algorithm that can be used for both Regression and Classification. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

```
xgb_grid  <-  expand.grid(
                        nrounds = 50,
                        eta = c(0.03),
                        max_depth = 1,
                        gamma = 0,
                        colsample_bytree = 0.6,
                        min_child_weight = 1,
                        subsample = 0.5
)

xgb_model <- caret::train(diabetes ~., data = train_set,
                         method = "xgbTree",
                         metric = "ROC",
                         tuneGrid=xgb_grid,
                         trControl = trainControl(method = "cv", number = 10,
                                                  classProbs = T, summaryFunction = twoClassSummary),
```

```
                           preProcess = c("center","scale","pca"))
xgb_model
```

```
## eXtreme Gradient Boosting
##
## 615 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (8), scaled (8), principal component signal
##  extraction (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 553, 553, 554, 553, 554, 554, ...
## Resampling results:
##
##   ROC        Sens    Spec
##   0.7924892  0.9175  0.3766234
##
## Tuning parameter 'nrounds' was held constant at a value of 50
## Tuning
##  held constant at a value of 1
## Tuning parameter 'subsample' was held
##  constant at a value of 0.5
```

```
xgb_model$results["ROC"]
```

ROC
0.7924892

Model 4 - K Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm is a supervised machine learning algorithm that can be used to solve both classification and regression problems.The principle behind this technique is that known data are arranged in a space defined by the selected features. When a new data is supplied to the algorithm, the algorithm will compare the classes of the k closest data to determine the class of the new data.

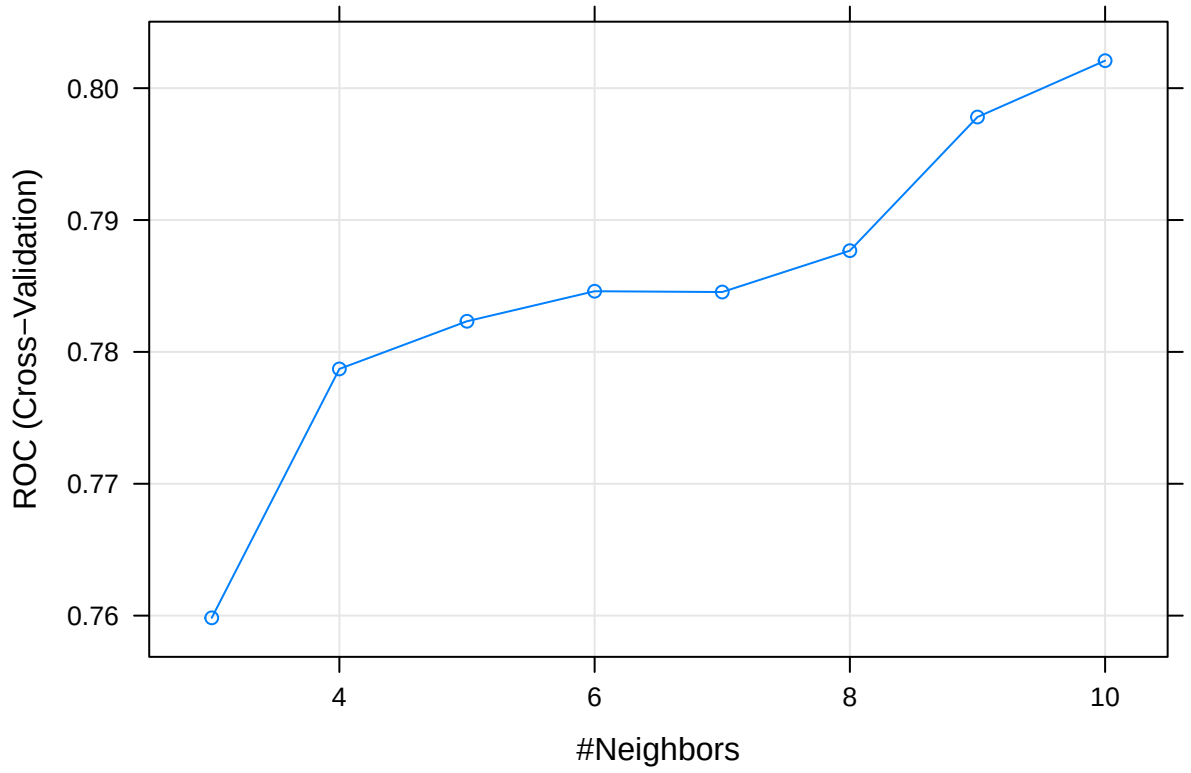
```
knn_model <- caret::train(diabetes ~., data = train_set,
                          method = "knn",
                          metric = "ROC",
                          tuneGrid = expand.grid(.k = c(3:10)),
                          trControl = trainControl(method = "cv", number = 10,
                                                   classProbs = T, summaryFunction = twoClassSummary),
                          preProcess = c("center","scale","pca"))

knn_model
```

```
## k-Nearest Neighbors
##
## 615 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
```

```
##
## Pre-processing: centered (8), scaled (8), principal component signal
##  extraction (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 553, 554, 553, 553, 554, 554, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens    Spec
##    3  0.7598323  0.8350  0.5956710
##    4  0.7787067  0.8325  0.5913420
##    5  0.7823241  0.8575  0.5766234
##    6  0.7845996  0.8475  0.5625541
##    7  0.7845400  0.8500  0.5484848
##    8  0.7876759  0.8425  0.5387446
##    9  0.7978111  0.8600  0.5632035
##   10  0.8020833  0.8675  0.5402597
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 10.
```

```
plot(knn_model)
```




```
knn_model$results[7,2]
```

```
## [1] 0.7978111
```

Model 5 - Support vector machine (SVM)

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. SVM focuses on the dual aspects of maximizing the minimum margin between the hyperplane and support vectors; and minimizing the misclassification rate.

Selecting the best parameters using the tune() function to do a grid search over the supplied parameter ranges (C - cost, gamma), using the train set. The range to gamma parameter is between 0.000001 and 0.1. For cost parameter the range is from 0.1 until 10.

```
model_svmtune <- tune.svm(diabetes ~., data = train_set, gamma = 10^(-6:-1), cost = 10^(-1:1))
summary(model_svmtune) # to show the results
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##   0.01    1
##
## - best performance: 0.2147805
##
## - Detailed performance results:
##    gamma cost     error dispersion
## 1  1e-06  0.1 0.3497091 0.04952648
## 2  1e-05  0.1 0.3497091 0.04952648
## 3  1e-04  0.1 0.3497091 0.04952648
## 4  1e-03  0.1 0.3497091 0.04952648
## 5  1e-02  0.1 0.3513485 0.05019586
## 6  1e-01  0.1 0.2407985 0.03685025
## 7  1e-06  1.0 0.3497091 0.04952648
## 8  1e-05  1.0 0.3497091 0.04952648
## 9  1e-04  1.0 0.3497091 0.04952648
## 10 1e-03  1.0 0.3464569 0.05064470
## 11 1e-02  1.0 0.2147805 0.02859493
## 12 1e-01  1.0 0.2195928 0.03152126
## 13 1e-06 10.0 0.3497091 0.04952648
## 14 1e-05 10.0 0.3497091 0.04952648
## 15 1e-04 10.0 0.3464569 0.05064470
## 16 1e-03 10.0 0.2163670 0.02148552
## 17 1e-02 10.0 0.2228715 0.02947868
## 18 1e-01 10.0 0.2308831 0.03134915
```

```
# As we can see the result show that the best parameters are Cost=10 and gamma=0.01.

svm_model  <- svm(diabetes ~., data = train_set, kernel = "radial", gamma = 0.01, cost = 10, probability
summary(svm_model)
```

```
##
## Call:
```

```
## svm(formula = diabetes ~ ., data = train_set, kernel = "radial",
##     gamma = 0.01, cost = 10, probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  320
##
##  ( 159 161 )
##
##
## Number of Classes:  2
##
## Levels:
##  neg pos
```

Prediction on the Test data set.

Model 1 - Logistic Regression

Prediction on Test data set

```
pred_glm <- predict(glm_model, test_set)

# Confusion Matrix
cmx_glm <- confusionMatrix(pred_glm, test_set$diabetes, positive="pos")


pred_prob_glm <- predict(glm_model, test_set, type="prob")
# ROC value
roc_glm <- roc(test_set$diabetes, pred_prob_glm$pos)
```

```
## Setting levels: control = neg, case = pos
```

```
## Setting direction: controls < cases
```

```
# Confusion matrix
cmx_glm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  87  28
##        pos  13  25
##
##                Accuracy : 0.732
##                  95% CI : (0.6545, 0.8003)
##     No Information Rate : 0.6536
```

```
##     P-Value [Acc > NIR] : 0.02367
##
##                   Kappa : 0.366
##
##  Mcnemar's Test P-Value : 0.02878
##
##             Sensitivity : 0.4717
##             Specificity : 0.8700
##          Pos Pred Value : 0.6579
##          Neg Pred Value : 0.7565
##              Prevalence : 0.3464
##          Detection Rate : 0.1634
##    Detection Prevalence : 0.2484
##       Balanced Accuracy : 0.6708
##
##        'Positive' Class : pos
##
```

```
# ROC

roc_glm
```

```
##
## Call:
## roc.default(response = test_set$diabetes, predictor = pred_prob_glm$pos)
##
## Data: pred_prob_glm$pos in 100 controls (test_set$diabetes neg) < 53 cases (test_set$diabetes pos).
## Area under the curve: 0.8158
```

Model 2 - Classification tree

Prediction on the test data set.

```
pred_rpart <- predict(tree_model, test_set, type = "class")

# prediction probabilities
tree_prob_pred <- predict(tree_model, test_set, type = "prob")

# confusion matrix
cmx_tree <- confusionMatrix(test_set$diabetes, pred_rpart)

cmx_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  93   7
##        pos  37  16
##
##                Accuracy : 0.7124
##                  95% CI : (0.6338, 0.7826)
```

```
##     No Information Rate : 0.8497
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2675
##
##  Mcnemar's Test P-Value : 1.232e-05
##
##             Sensitivity : 0.7154
##             Specificity : 0.6957
##          Pos Pred Value : 0.9300
##          Neg Pred Value : 0.3019
##              Prevalence : 0.8497
##          Detection Rate : 0.6078
##    Detection Prevalence : 0.6536
##       Balanced Accuracy : 0.7055
##
##        'Positive' Class : neg
##
```

Model 3 - XGBoost

Prediction on Test data set

```
pred_xgb <- predict(xgb_model, test_set)

# Confusion Matrix
cmx_xgb <- confusionMatrix(pred_xgb, test_set$diabetes, positive="pos")


pred_prob_xgb <- predict(xgb_model, test_set, type="prob")
# ROC value
roc_xgb <- roc(test_set$diabetes, pred_prob_xgb$pos)
```

```
## Setting levels: control = neg, case = pos
```

```
## Setting direction: controls < cases
```

```
# Confusion matrix
cmx_xgb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  96  46
##        pos   4   7
##
##                Accuracy : 0.6732
##                  95% CI : (0.5928, 0.7468)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.3382
##
```

```
##                   Kappa : 0.1131
##
##  Mcnemar's Test P-Value : 6.7e-09
##
##             Sensitivity : 0.13208
##             Specificity : 0.96000
##          Pos Pred Value : 0.63636
##          Neg Pred Value : 0.67606
##              Prevalence : 0.34641
##          Detection Rate : 0.04575
##    Detection Prevalence : 0.07190
##       Balanced Accuracy : 0.54604
##
##        'Positive' Class : pos
##
```

Model 4 - KNN

Prediction on Test data set

```
pred_knn <- predict(knn_model, test_set)

# Confusion Matrix
cmx_knn <- confusionMatrix(pred_knn, test_set$diabetes, positive="pos")

pred_prob_knn <- predict(knn_model, test_set, type="prob")
# ROC value
roc_knn <- roc(test_set$diabetes, pred_prob_knn$pos)
```

```
## Setting levels: control = neg, case = pos
```

```
## Setting direction: controls < cases
```

```
# Confusion matrix
cmx_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  89  29
##        pos  11  24
##
##                Accuracy : 0.7386
##                  95% CI : (0.6615, 0.8062)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 0.01536
##
##                   Kappa : 0.3726
##
##  Mcnemar's Test P-Value : 0.00719
##
```

```
##             Sensitivity : 0.4528
##             Specificity : 0.8900
##          Pos Pred Value : 0.6857
##          Neg Pred Value : 0.7542
##              Prevalence : 0.3464
##          Detection Rate : 0.1569
##    Detection Prevalence : 0.2288
##       Balanced Accuracy : 0.6714
##
##        'Positive' Class : pos
##
```

Model 5 - SVM

Prediction on the test data

```
par(mfrow = c(1,2))
pred_svm <- predict(svm_model, test_set)


svm_prob_pred <- predict(svm_model,test_set,probability = TRUE)

# Confusion matrix
cmx_svm <- confusionMatrix(test_set$diabetes, svm_prob_pred)

cmx_svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  88  12
##        pos  30  23
##
##                Accuracy : 0.7255
##                  95% CI : (0.6476, 0.7945)
##     No Information Rate : 0.7712
##     P-Value [Acc > NIR] : 0.923228
##
##                   Kappa : 0.3412
##
##  Mcnemar's Test P-Value : 0.008712
##
##             Sensitivity : 0.7458
##             Specificity : 0.6571
##          Pos Pred Value : 0.8800
##          Neg Pred Value : 0.4340
##              Prevalence : 0.7712
##          Detection Rate : 0.5752
##    Detection Prevalence : 0.6536
##       Balanced Accuracy : 0.7015
##
##        'Positive' Class : neg
##
```

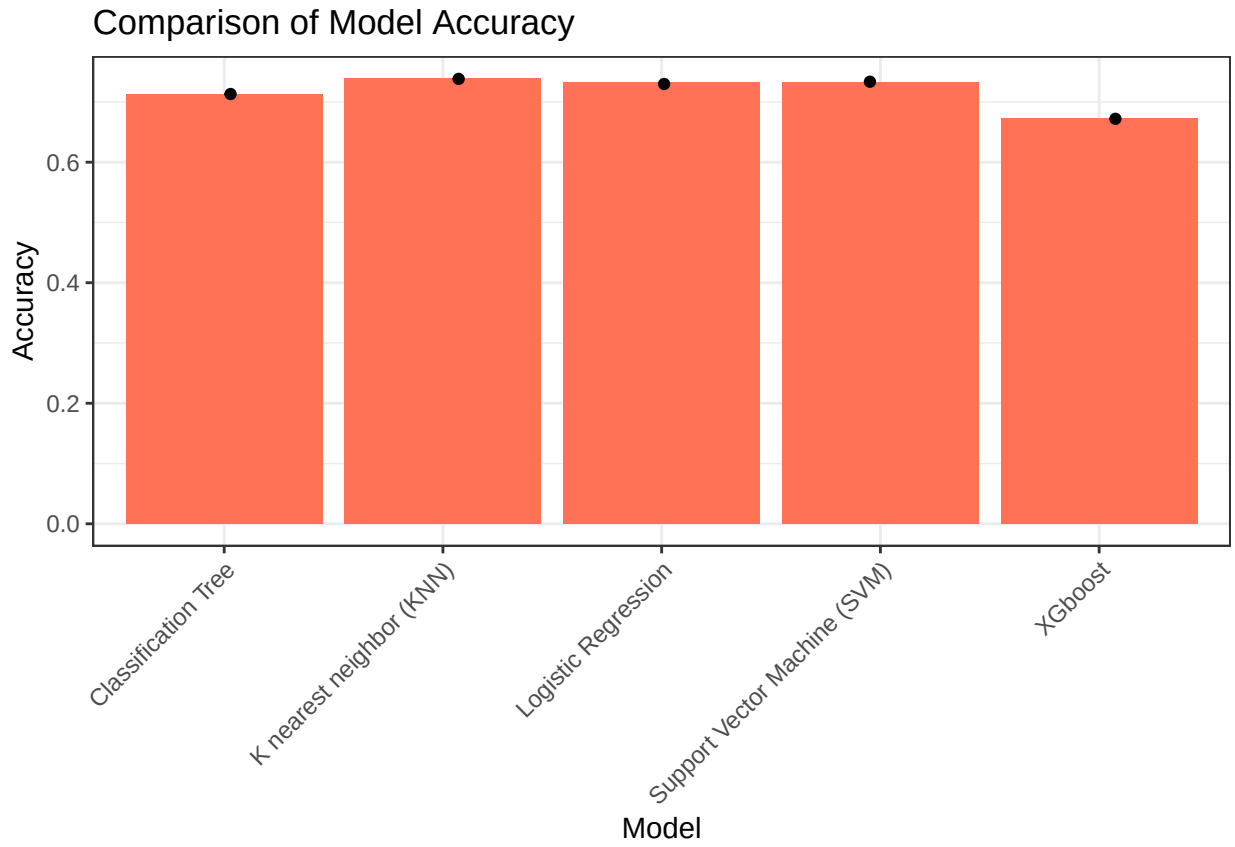
Comparing the Test results of all the models.

Results

	Sensitivity	Specificity	Precision	Recall	F1
test_glm	0.4716981	0.8700000	0.6578947	0.4716981	0.5494505
test_tree	0.7153846	0.6956522	0.9300000	0.7153846	0.8086957
test_knn	0.4528302	0.8900000	0.6857143	0.4528302	0.5454545
test_xgb	0.1320755	0.9600000	0.6363636	0.1320755	0.2187500
test_svm	0.7457627	0.6571429	0.8800000	0.7457627	0.8073394

Comparing the accuracy of all the models

Model	Accuracy
Logistic Regression	0.7320261
Classification Tree	0.7124183
XGboost	0.6732026
K nearest neighbor (KNN)	0.7385621
Support Vector Machine (SVM)	0.7320261



After comparing the results it is evident that all the models have pros and cons and no single model has a perfect combination that makes it superior to the other models.

Conclusion

The Pima Indian dataset was explored and analyzed in detail. Multiple machine learning models were built and tested in order to identify the best performing model to predict the occurrence of diabetes in Pima Indian women. According to the results the Logistic Regression, and SVM performed similarly in terms of accuracy. Accuracy can be defined as the percentage of correct predictions for the test data. The Support vector machine model and Classification tree performed best in terms of Recall which can be defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class. The SVM model also performed best on Precision which tells us about the percentage of positive instances out of the total predicted positive instances. The SVM and Classification tree model performed best in terms of Sensitivity which is the ability of the test to correctly identify the true positive rate. The Extreme gradient boosting model performed the best in terms of Specificity which is the ability of the test to correctly identify the true negative rate. The SVM model also had the highest F1 score which can be defined as the harmonic mean of precision and recall. The findings suggest that Support vector machine modeling is a promising classification approach for detecting persons with diabetes in the population. This approach should be further explored in other complex diseases using common variables.

Enviroment

```
## [1] "Operating System:"

##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          0.2
## year           2020
## month          06
## day            22
## svn rev        78730
## language       R
## version.string R version 4.0.2 (2020-06-22)
## nickname       Taking Off Again
```

‘‘‘