

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.	30
Results	30
Comparing the accuracy of all the models	30
Conclusion	31
Enviroment	31

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 diabetic 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 Boosting (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

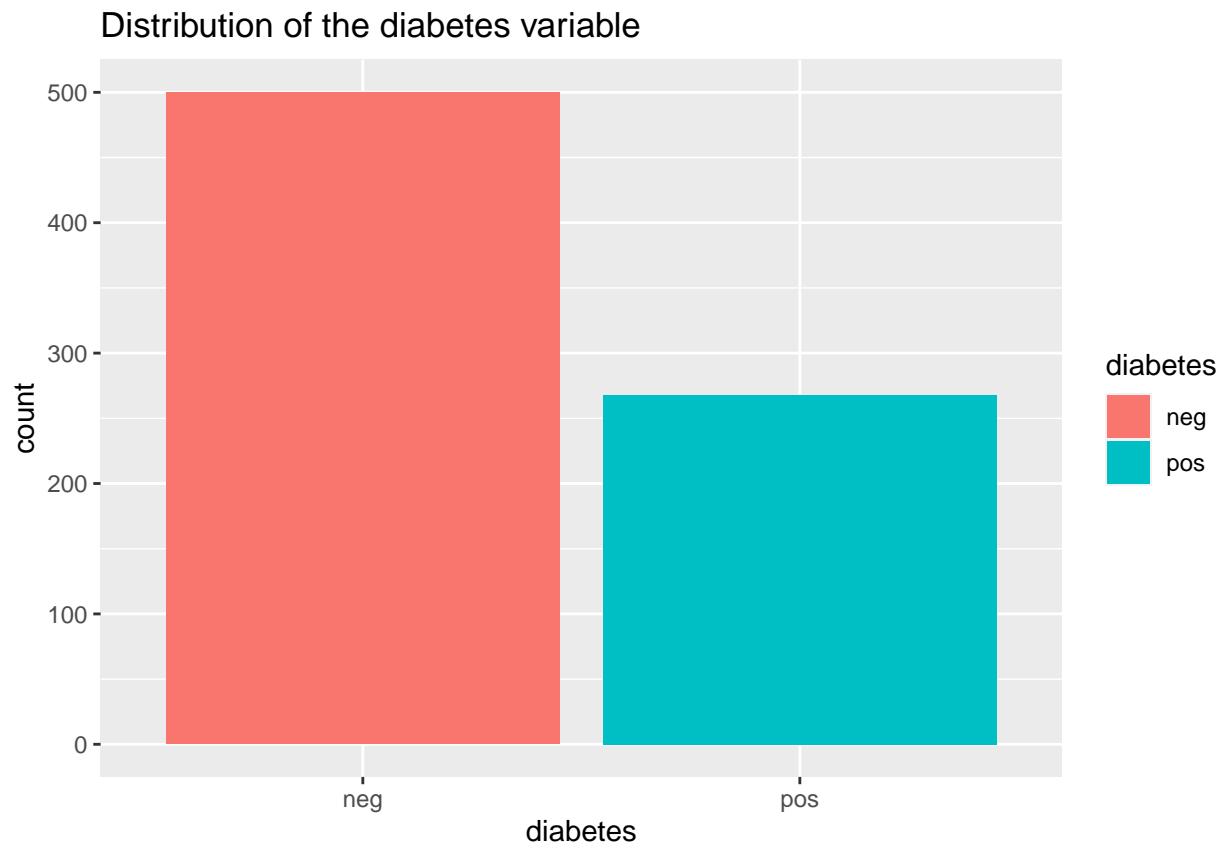
```
## [1] 768 9
```

Check for NA's or missing values

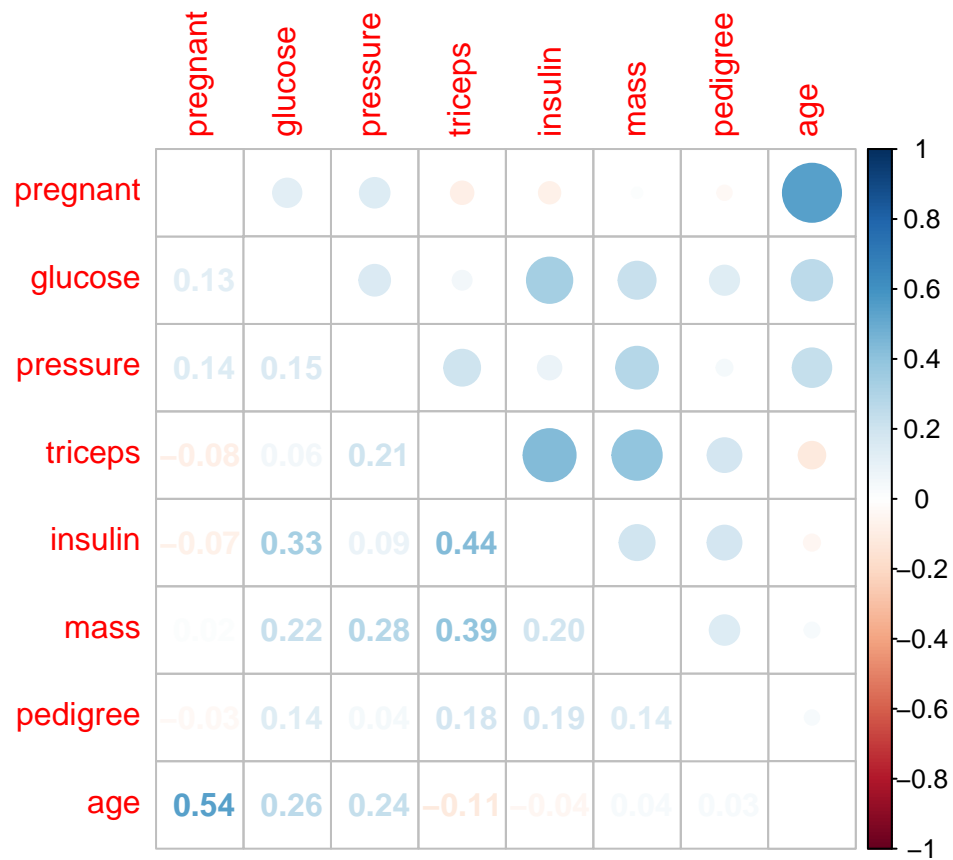
```
## pregnant glucose pressure triceps insulin mass pedigree age
##      0      0      0      0      0      0      0      0
## diabetes
##      0
```

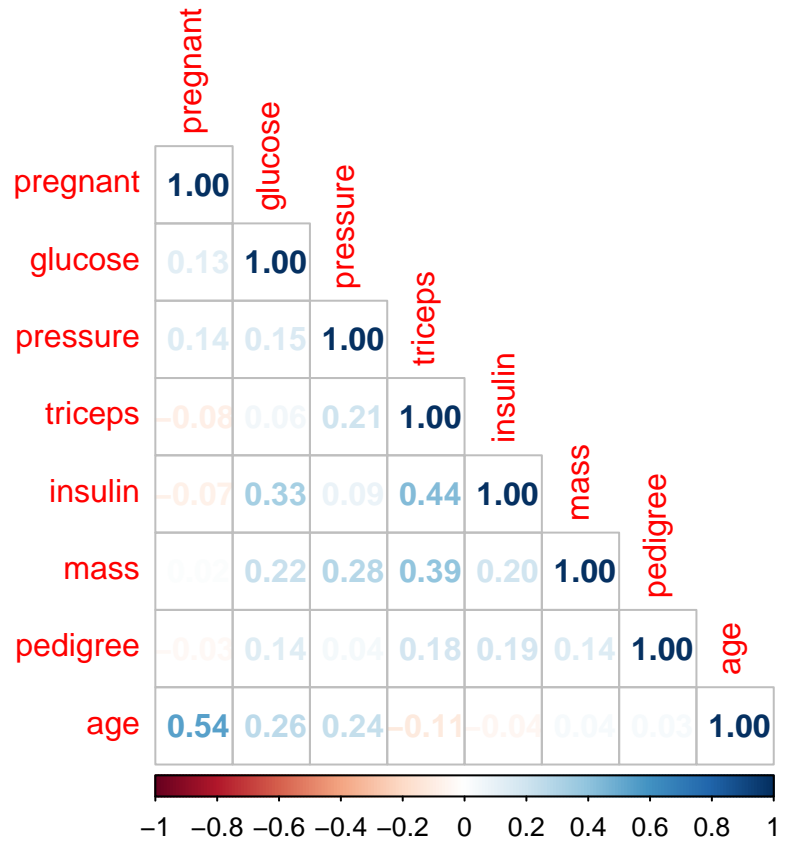
Exploring the response variable Diabetes

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



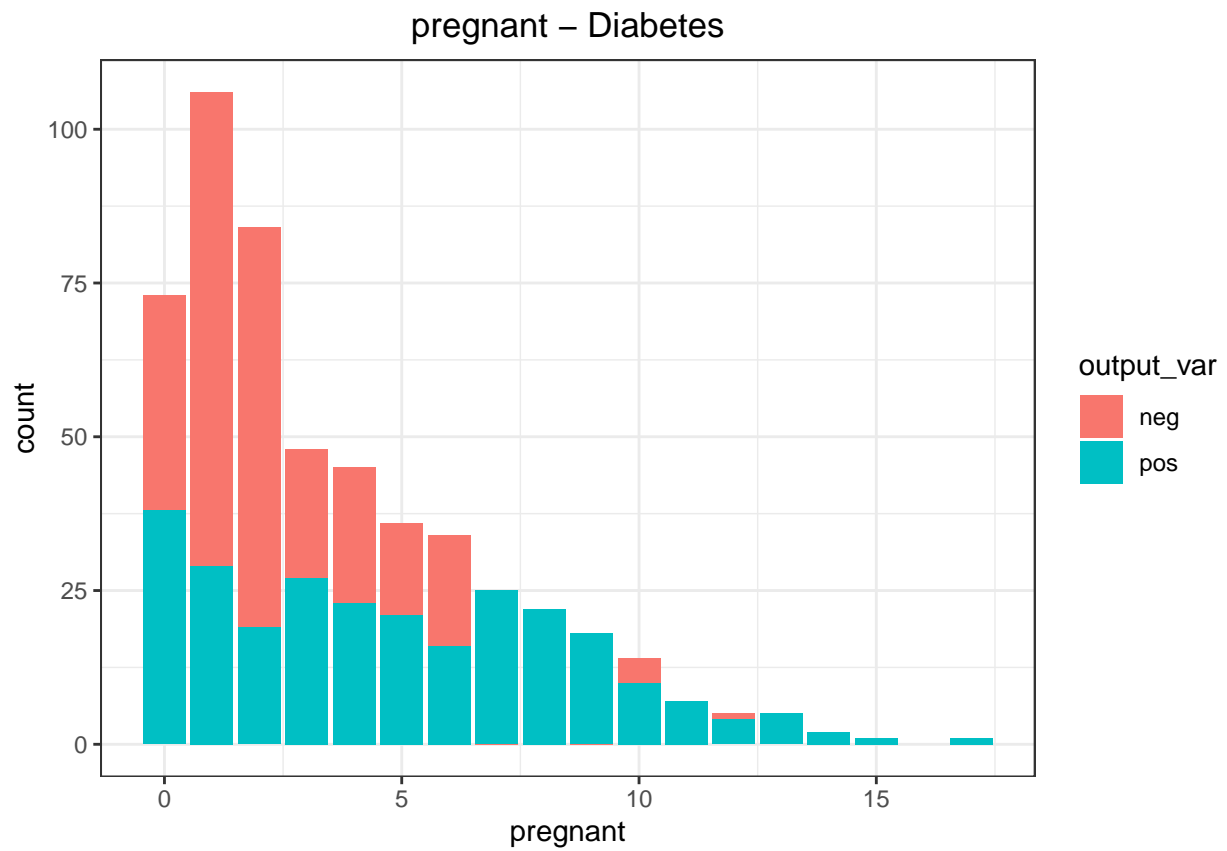
Plot of Correlations between all the predictor variables

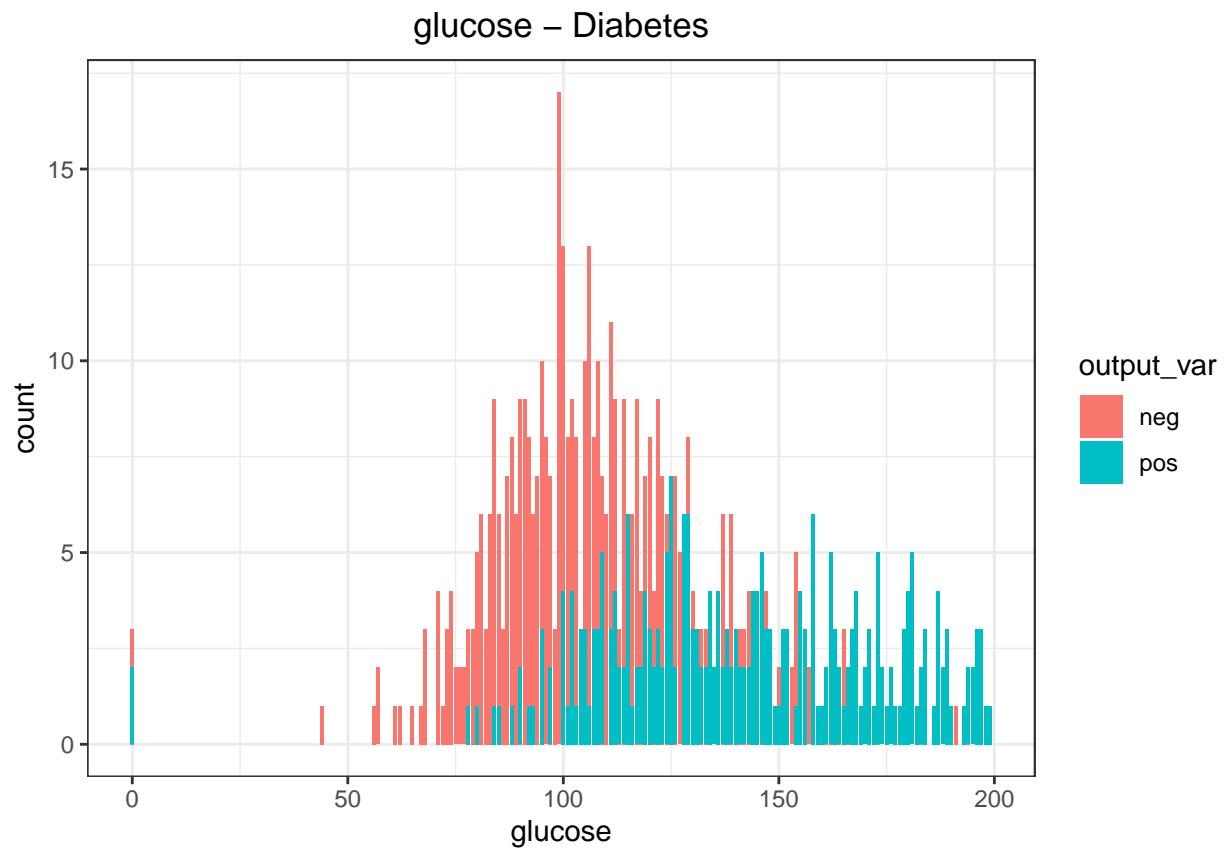


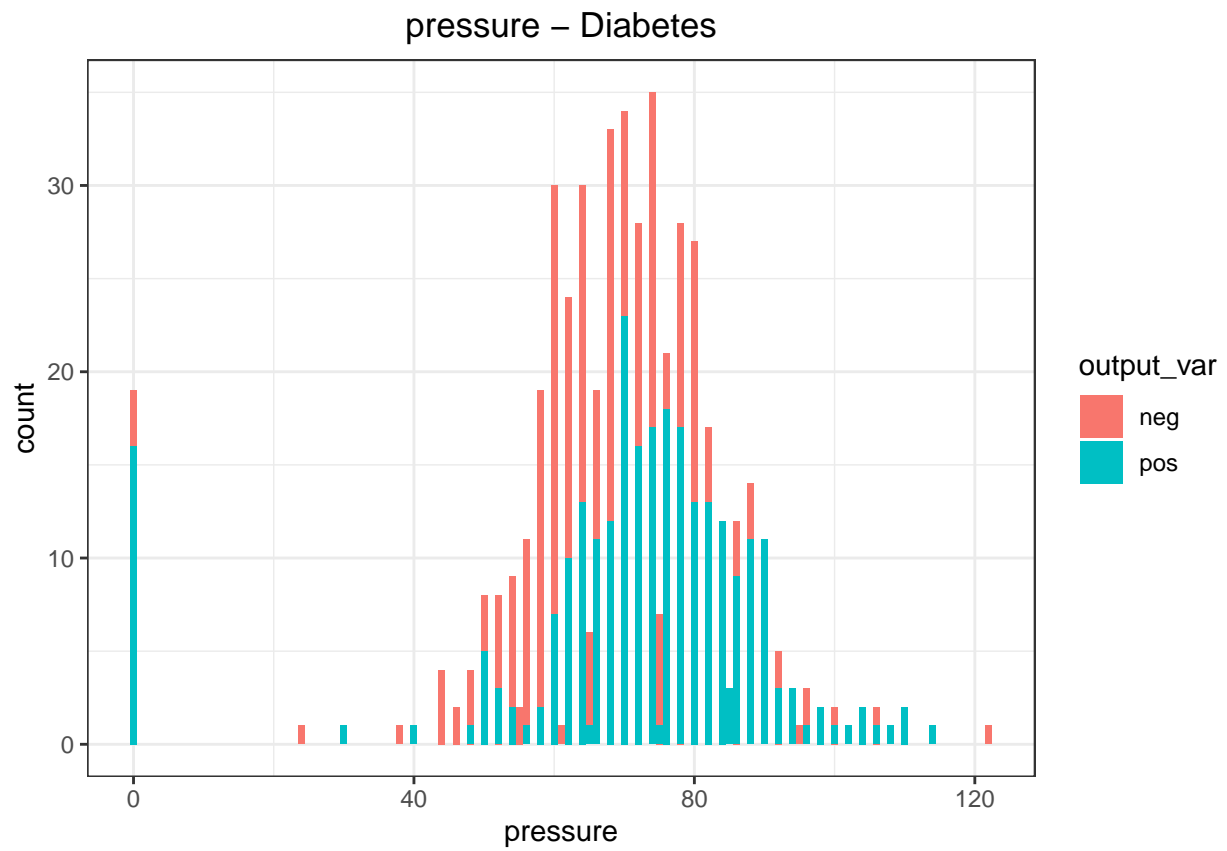


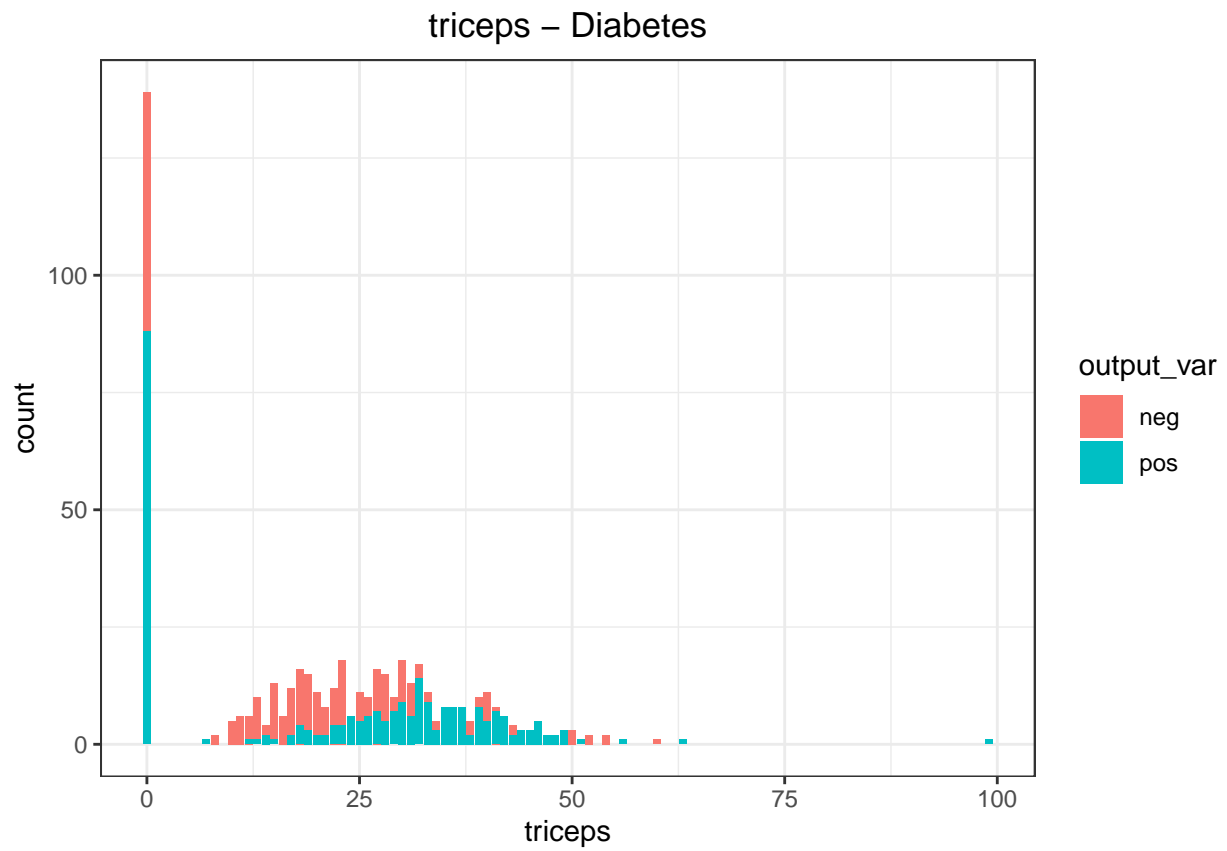
Numerical Representation

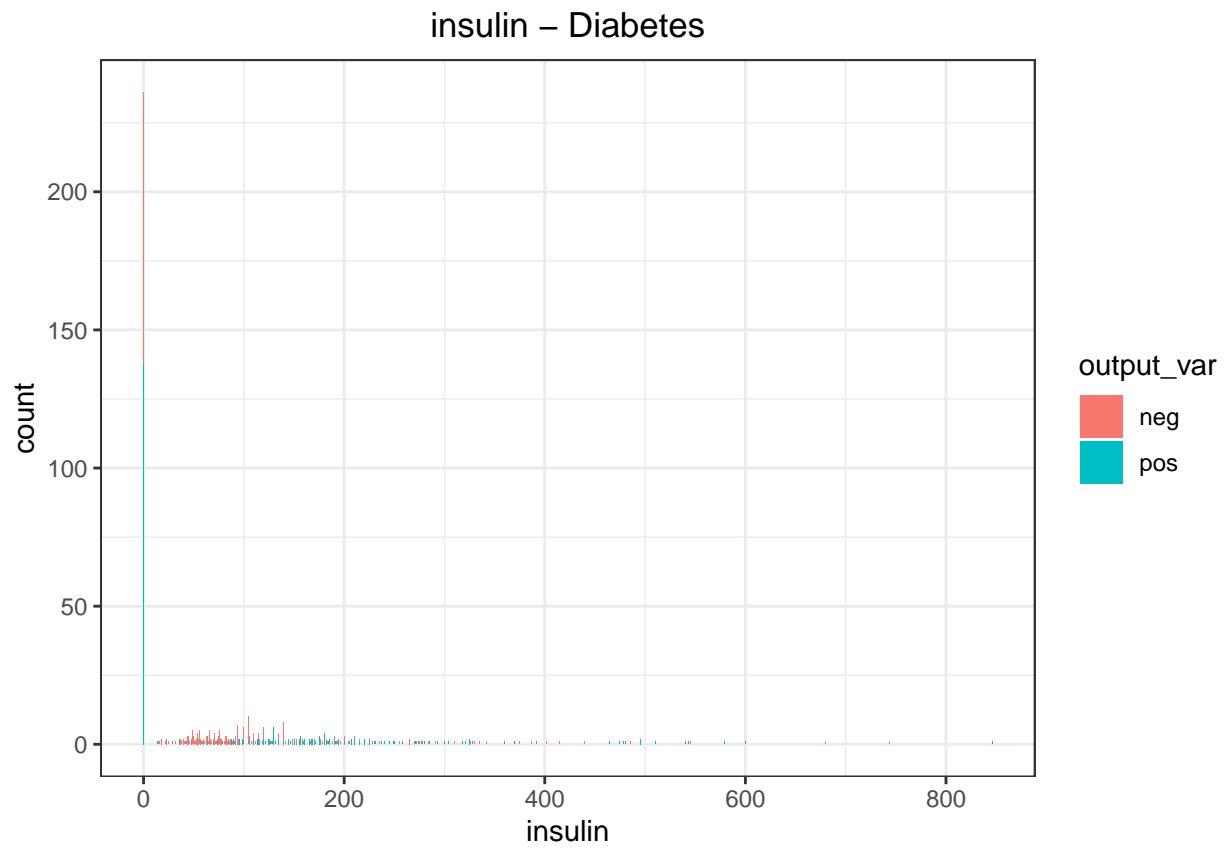
Univariate Analysis

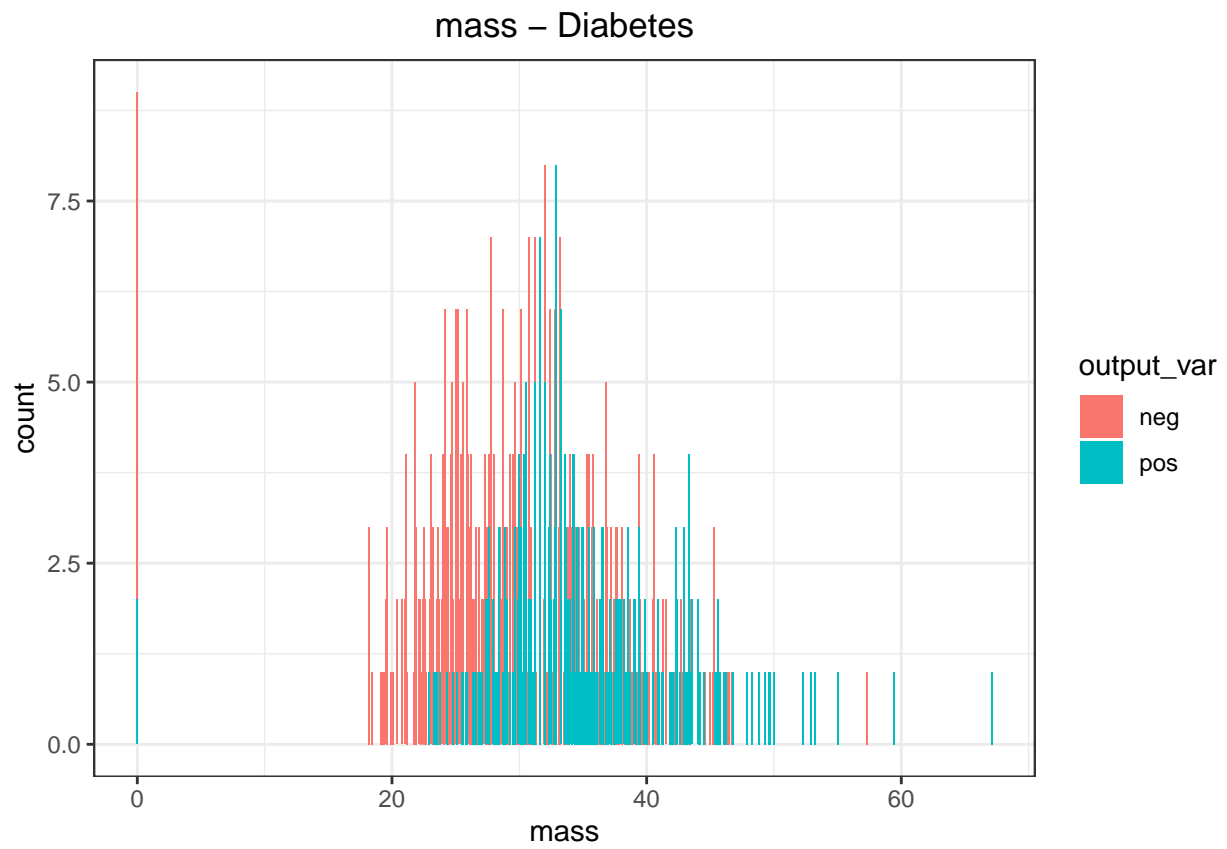


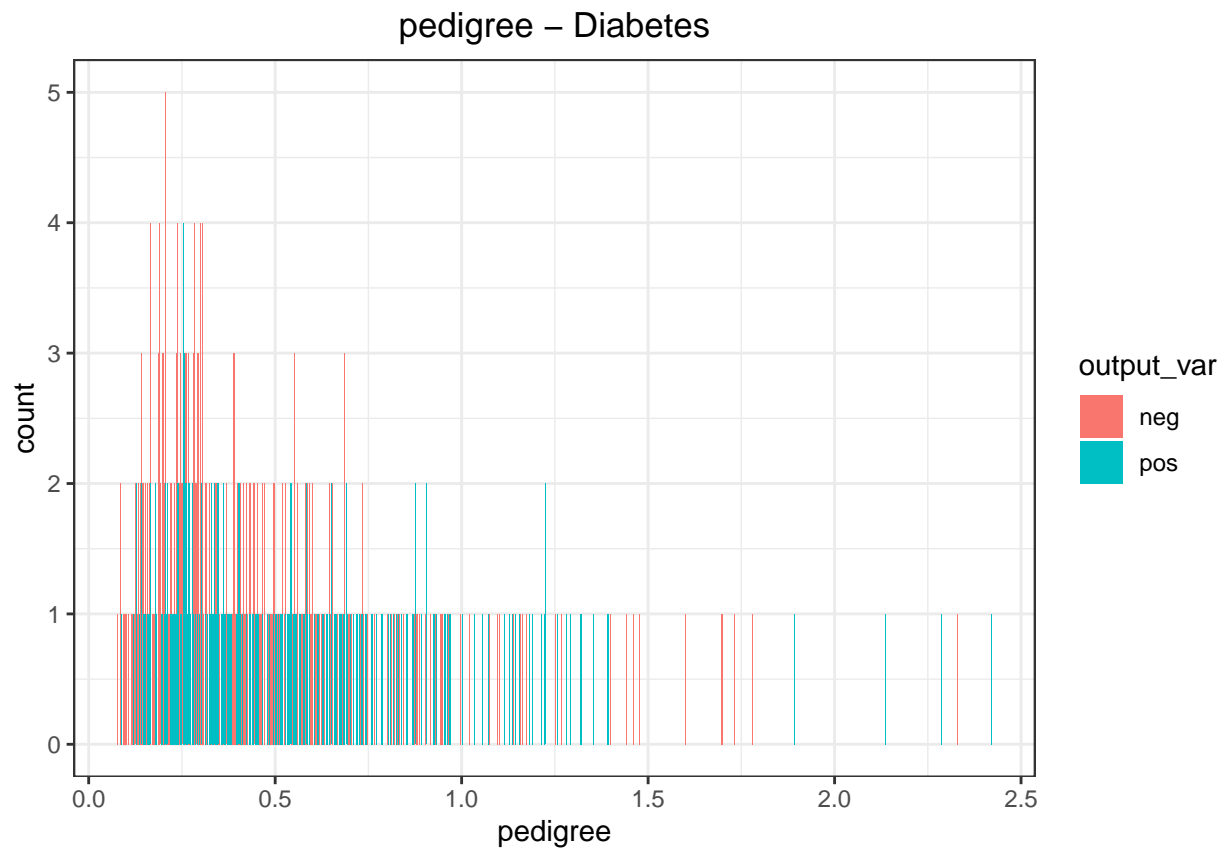


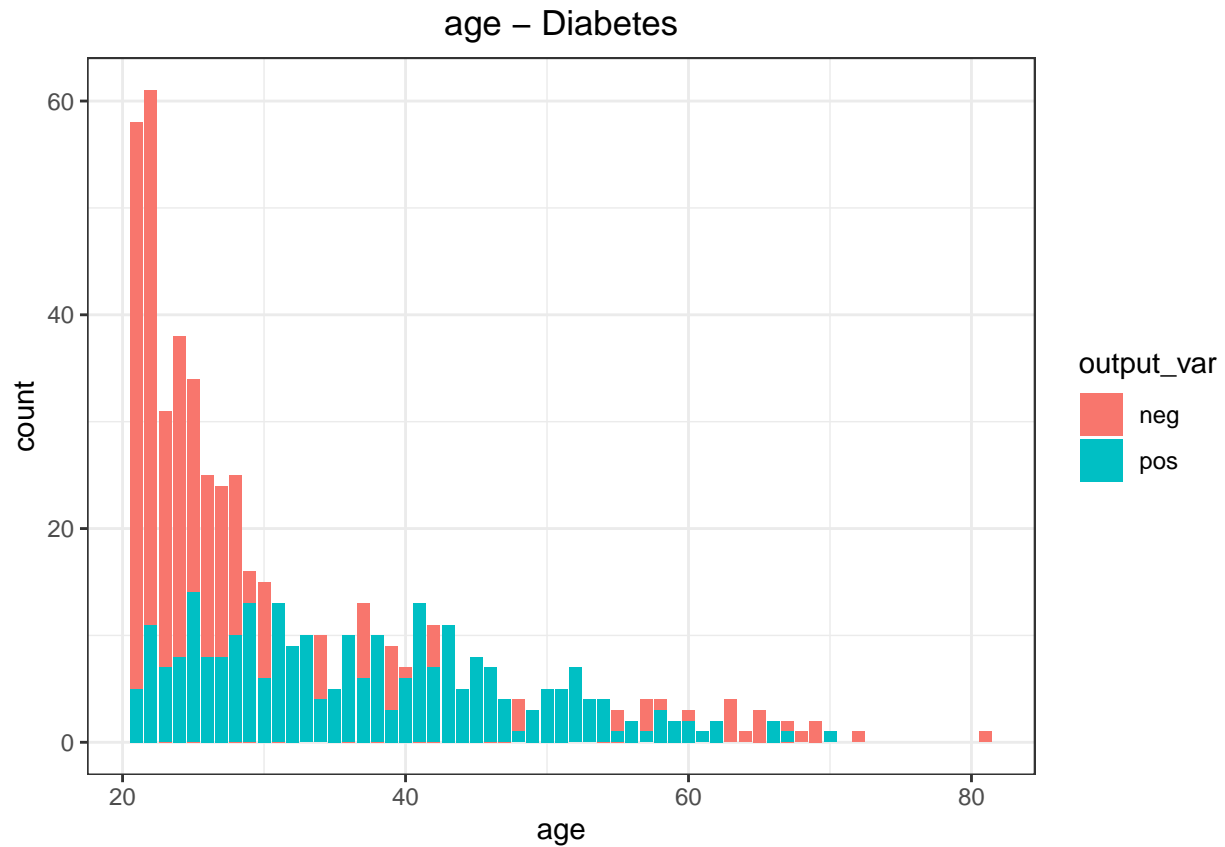






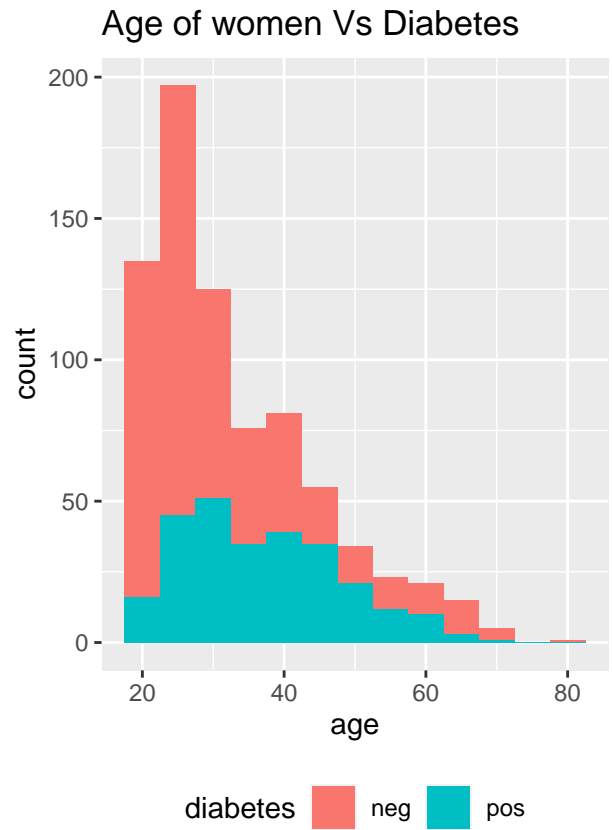
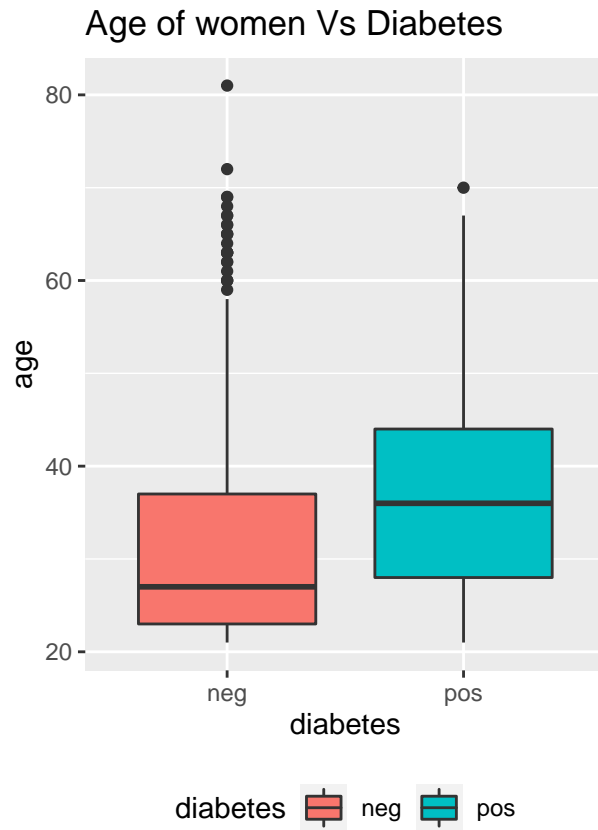






It is evident that high glucose levels lead to a higher chance of positive diabetes diagnosis. Mass/BMI increases also increase the chance of a positive diabetes diagnosis. Age over the age of 25 also an indicator that increases the chance of a diabetes diagnosis. There is no notable significant distinction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

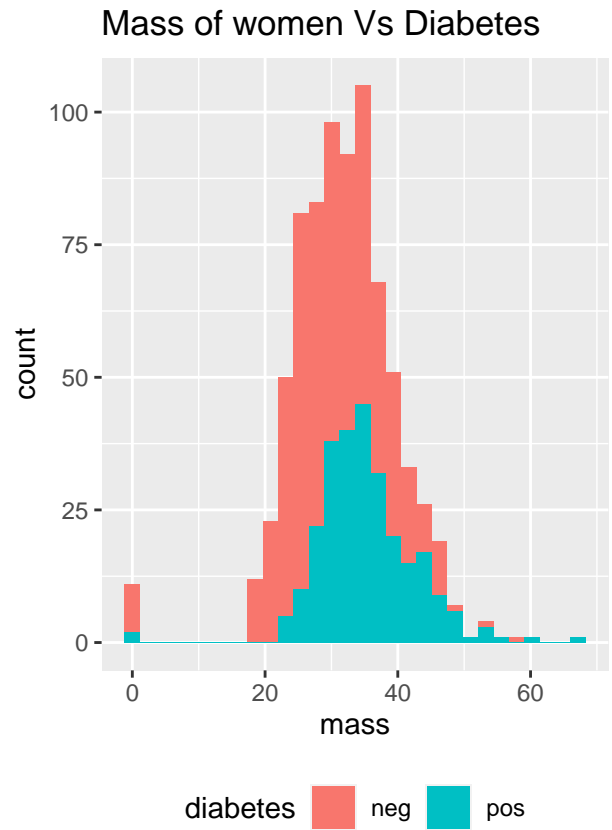
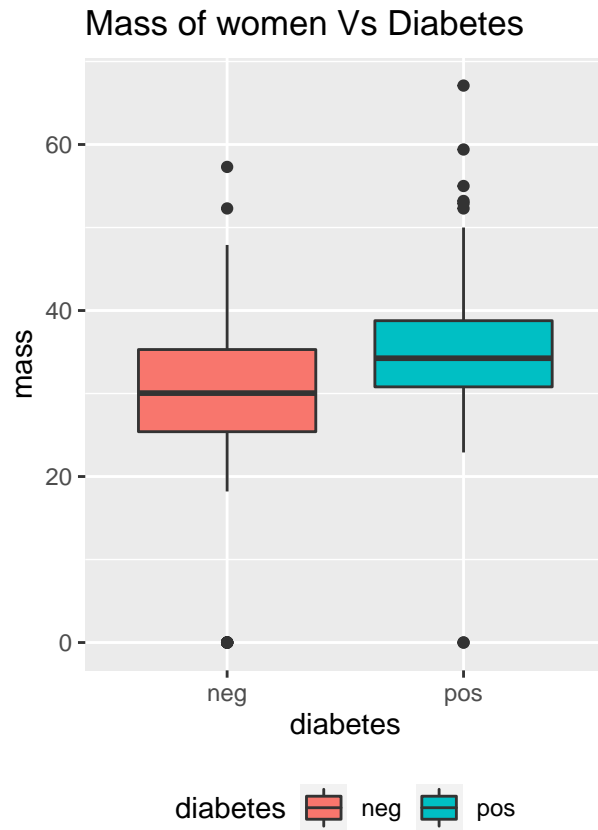
Exploring the Age variable



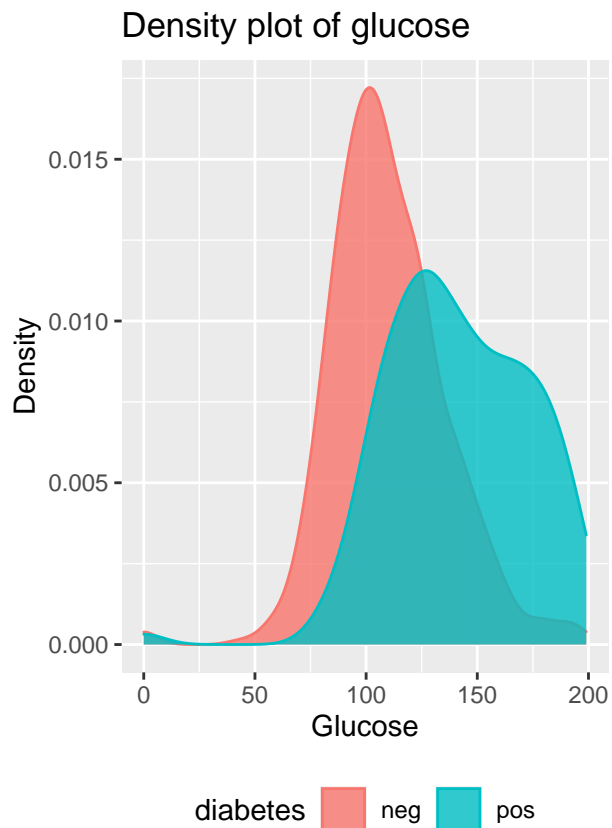
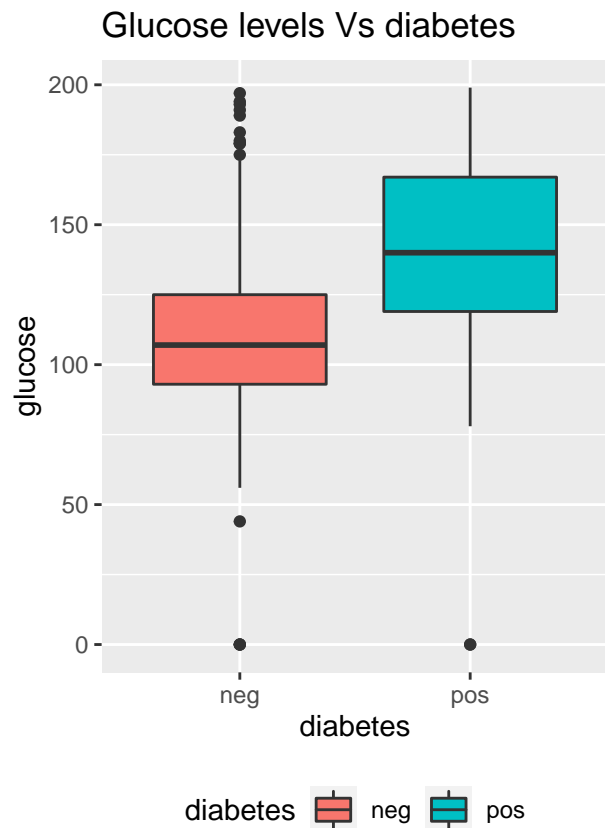
Exploring the mass variable

```
## 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 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 ...
```

	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 : 70.00	Median :23.00
## Mean :	3.813	Mean :120.9	Mean : 68.93	Mean :20.74
## 3rd Qu.:	6.000	3rd Qu.:141.0	3rd Qu.: 79.00	3rd Qu.:32.00
## Max. :	17.000	Max. :197.0	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.:0.2380   1st Qu.:24.00   pos:215
## Median : 23.00   Median :32.00   Median :0.3700   Median :29.00
## Mean   : 79.12   Mean   :31.95   Mean   :0.4661   Mean   :33.46
## 3rd Qu.:130.00   3rd Qu.:36.15   3rd Qu.:0.6135   3rd Qu.:41.00
## Max.   :846.00   Max.   :67.10   Max.   :2.4200   Max.   :81.00

##      pregnant      glucose      pressure      triceps
## Min.   : 0.000   Min.   : 67.0   Min.   : 0.0   Min.   : 0.00
## 1st Qu.: 1.000   1st Qu.:100.0   1st Qu.: 64.0   1st Qu.: 0.00
## Median : 3.000   Median :115.0   Median : 74.0   Median :23.00
## Mean   : 3.974   Mean   :120.7   Mean   : 69.8   Mean   :19.71
## 3rd Qu.: 6.000   3rd Qu.:137.0   3rd Qu.: 82.0   3rd Qu.:32.00
## Max.   :14.000   Max.   :199.0   Max.   :114.0   Max.   :50.00
##      insulin      mass      pedigree      age      diabetes
## Min.   : 0.00   Min.   : 0.00   Min.   :0.0880   Min.   :21.00   neg:100
## 1st Qu.: 0.00   1st Qu.:27.40   1st Qu.:0.2570   1st Qu.:24.00   pos: 53
## Median : 37.00   Median :32.90   Median :0.3890   Median :29.00
## Mean   : 82.54   Mean   :32.16   Mean   :0.4952   Mean   :32.36
## 3rd Qu.:120.00   3rd Qu.:37.80   3rd Qu.:0.6470   3rd Qu.:39.00
## Max.   :579.00   Max.   :57.30   Max.   :2.1370   Max.   :66.00
```

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] 0.8209903
```

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
```

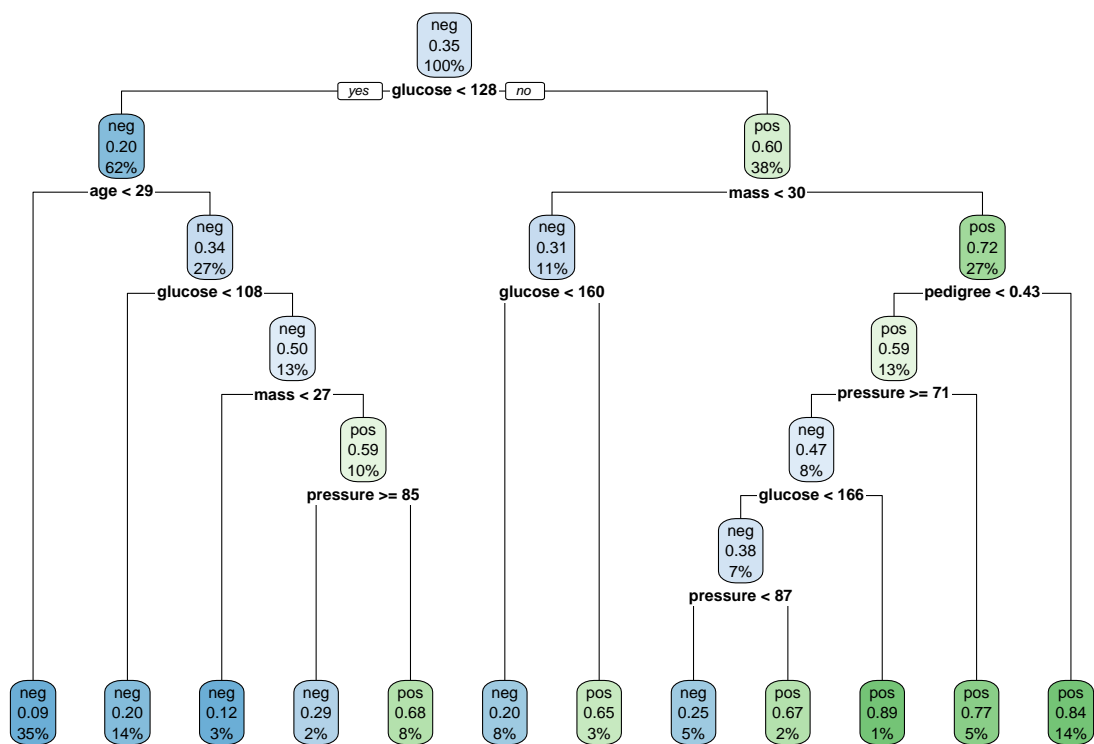
```
## n= 615
##
```

```

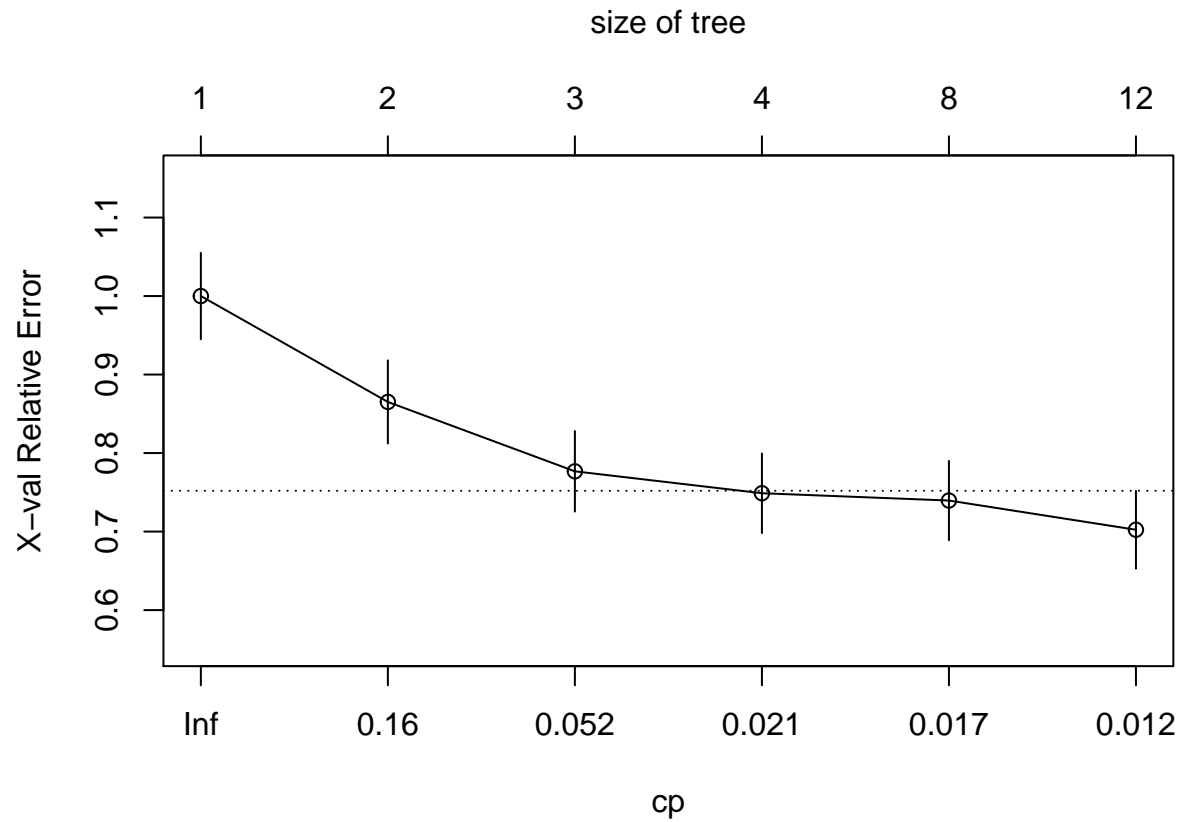
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
## 1) root 615 215 neg (0.65040650 0.34959350)
##    2) glucose< 127.5 383 76 neg (0.80156658 0.19843342)
##      4) age< 28.5 216 19 neg (0.91203704 0.08796296) *
##      5) age>=28.5 167 57 neg (0.65868263 0.34131737)
##        10) glucose< 107.5 87 17 neg (0.80459770 0.19540230) *
##        11) glucose>=107.5 80 40 neg (0.50000000 0.50000000)
##          22) mass< 26.75 16 2 neg (0.87500000 0.12500000) *
##          23) mass>=26.75 64 26 pos (0.40625000 0.59375000)
##            46) pressure>=85 14 4 neg (0.71428571 0.28571429) *
##            47) pressure< 85 50 16 pos (0.32000000 0.68000000) *
##    3) glucose>=127.5 232 93 pos (0.40086207 0.59913793)
##      6) mass< 29.95 67 21 neg (0.68656716 0.31343284)
##        12) glucose< 160 50 10 neg (0.80000000 0.20000000) *
##        13) glucose>=160 17 6 pos (0.35294118 0.64705882) *
##    7) mass>=29.95 165 47 pos (0.28484848 0.71515152)
##      14) pedigree< 0.434 80 33 pos (0.41250000 0.58750000)
##        28) pressure>=71 49 23 neg (0.53061224 0.46938776)
##          56) glucose< 165.5 40 15 neg (0.62500000 0.37500000)
##            112) pressure< 87 28 7 neg (0.75000000 0.25000000) *
##            113) pressure>=87 12 4 pos (0.33333333 0.66666667) *
##          57) glucose>=165.5 9 1 pos (0.11111111 0.88888889) *
##      29) pressure< 71 31 7 pos (0.22580645 0.77419355) *
##    15) pedigree>=0.434 85 14 pos (0.16470588 0.83529412) *

```

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



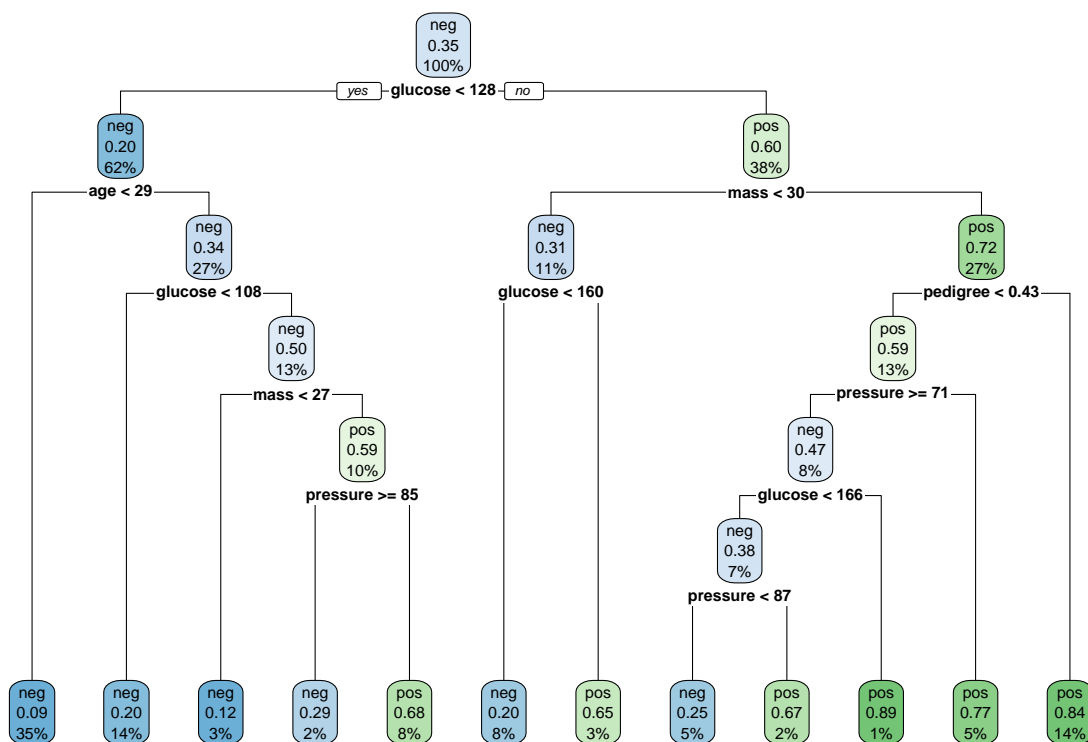
Best Model Complexity Parameter - CP



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, 554, 554, 553, 553, 554, ...
## Resampling results:
##
## ROC      Sens   Spec
## 0.782665 0.9175 0.3764069
##
## 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
0.782665

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

```

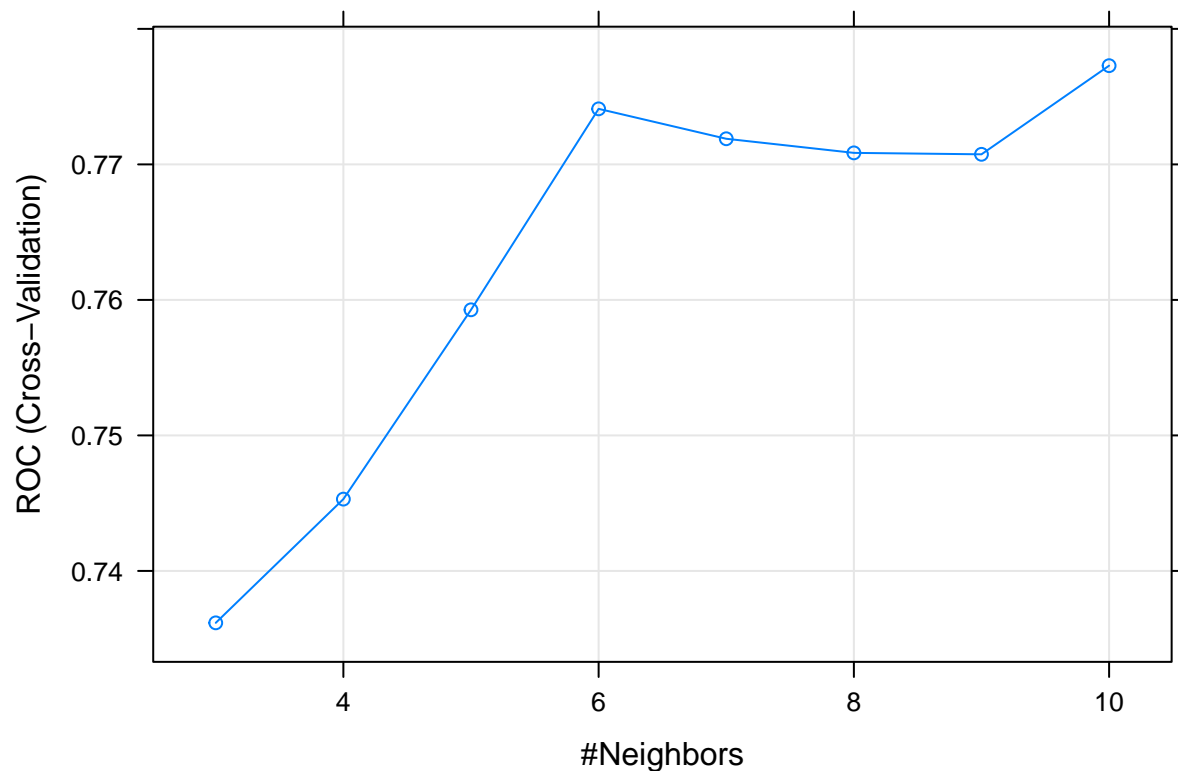
```

## 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, 554, 553, 553, 554, ...
## Resampling results across tuning parameters:
##
##   k   ROC      Sens   Spec
##   3  0.7361688  0.8325  0.5300866
##   4  0.7453003  0.8250  0.4887446
##   5  0.7592695  0.8450  0.5264069
##   6  0.7740990  0.8400  0.5266234
##   7  0.7718912  0.8475  0.4984848
##   8  0.7708496  0.8525  0.5216450
##   9  0.7707440  0.8525  0.5264069
##  10  0.7772835  0.8550  0.5259740
##
## 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] 0.770744
```


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: 0.2358012
##
## - Detailed performance results:
##   gamma cost      error dispersion
## 1  1e-06  0.1 0.3496298 0.06417097
## 2  1e-05  0.1 0.3496298 0.06417097
## 3  1e-04  0.1 0.3496298 0.06417097
## 4  1e-03  0.1 0.3496298 0.06417097
## 5  1e-02  0.1 0.3496298 0.06417097
## 6  1e-01  0.1 0.2749603 0.09230934
## 7  1e-06  1.0 0.3496298 0.06417097
## 8  1e-05  1.0 0.3496298 0.06417097
## 9  1e-04  1.0 0.3496298 0.06417097
## 10 1e-03  1.0 0.3479640 0.06483382
## 11 1e-02  1.0 0.2358012 0.06064462
## 12 1e-01  1.0 0.2700423 0.06829675
## 13 1e-06 10.0 0.3496298 0.06417097
## 14 1e-05 10.0 0.3496298 0.06417097
## 15 1e-04 10.0 0.3479640 0.06483382
## 16 1e-03 10.0 0.2390799 0.05769760
## 17 1e-02 10.0 0.2456372 0.06489226
## 18 1e-01 10.0 0.2522210 0.08911213
```

```
# 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 = TRUE)
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: 338
##
## ( 170 168 )
##
##
## Number of Classes: 2
##
## Levels:
##   neg pos
```

Prediction on the Test data set.

Model 1 - Logistic Regression

Prediction on Test data set

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

## Setting direction: controls < cases

## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##      neg  90  17
##      pos  10  36
##
##           Accuracy : 0.8235
##           95% CI : (0.7537, 0.8804)
##      No Information Rate : 0.6536
##      P-Value [Acc > NIR] : 2.528e-06
##
##           Kappa : 0.5978
##
## Mcnemar's Test P-Value : 0.2482
##
##           Sensitivity : 0.6792
##           Specificity : 0.9000
##      Pos Pred Value : 0.7826
##      Neg Pred Value : 0.8411
##           Prevalence : 0.3464
##      Detection Rate : 0.2353
##      Detection Prevalence : 0.3007
##      Balanced Accuracy : 0.7896
```

```
##
##      'Positive' Class : pos
##

##
## 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: 0.8794
```

Model 2 - Classification tree

Prediction on the test data set.

```
## Confusion Matrix and Statistics
##
##      Reference
## Prediction neg pos
##      neg  83  17
##      pos  22  31
##
##      Accuracy : 0.7451
##      95% CI : (0.6684, 0.812)
##      No Information Rate : 0.6863
##      P-Value [Acc > NIR] : 0.06736
##
##      Kappa : 0.4243
##
##      McNemar's Test P-Value : 0.52184
##
##      Sensitivity : 0.7905
##      Specificity : 0.6458
##      Pos Pred Value : 0.8300
##      Neg Pred Value : 0.5849
##      Prevalence : 0.6863
##      Detection Rate : 0.5425
##      Detection Prevalence : 0.6536
##      Balanced Accuracy : 0.7182
##
##      'Positive' Class : neg
##
```

Model 3 - XGBoost

Prediction on Test data set

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

## Setting direction: controls < cases
```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##      neg  90  30
##      pos  10  23
##
##           Accuracy : 0.7386
##           95% CI : (0.6615, 0.8062)
##      No Information Rate : 0.6536
##      P-Value [Acc > NIR] : 0.015356
##
##           Kappa : 0.3665
##
##  McNemar's Test P-Value : 0.002663
##
##      Sensitivity : 0.4340
##      Specificity : 0.9000
##      Pos Pred Value : 0.6970
##      Neg Pred Value : 0.7500
##      Prevalence : 0.3464
##      Detection Rate : 0.1503
##      Detection Prevalence : 0.2157
##      Balanced Accuracy : 0.6670
##
##      'Positive' Class : pos
##

```

Model 4 - KNN

Prediction on Test data set

```

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

## Setting direction: controls < cases

## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##      neg  85  22
##      pos  15  31
##
##           Accuracy : 0.7582
##           95% CI : (0.6824, 0.8237)
##      No Information Rate : 0.6536
##      P-Value [Acc > NIR] : 0.003479
##
##           Kappa : 0.4488
##
##  McNemar's Test P-Value : 0.323940
##
##      Sensitivity : 0.5849

```

```

##           Specificity : 0.8500
##           Pos Pred Value : 0.6739
##           Neg Pred Value : 0.7944
##           Prevalence : 0.3464
##           Detection Rate : 0.2026
##           Detection Prevalence : 0.3007
##           Balanced Accuracy : 0.7175
##
##           'Positive' Class : pos
##

```

Model 5 - SVM

Prediction on the test data

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##           neg  90  10
##           pos  18  35
##
##           Accuracy : 0.817
##           95% CI : (0.7465, 0.8748)
##           No Information Rate : 0.7059
##           P-Value [Acc > NIR] : 0.00116
##
##           Kappa : 0.581
##
## Mcnemar's Test P-Value : 0.18588
##
##           Sensitivity : 0.8333
##           Specificity : 0.7778
##           Pos Pred Value : 0.9000
##           Neg Pred Value : 0.6604
##           Prevalence : 0.7059
##           Detection Rate : 0.5882
##           Detection Prevalence : 0.6536
##           Balanced Accuracy : 0.8056
##
##           'Positive' Class : neg
##

```

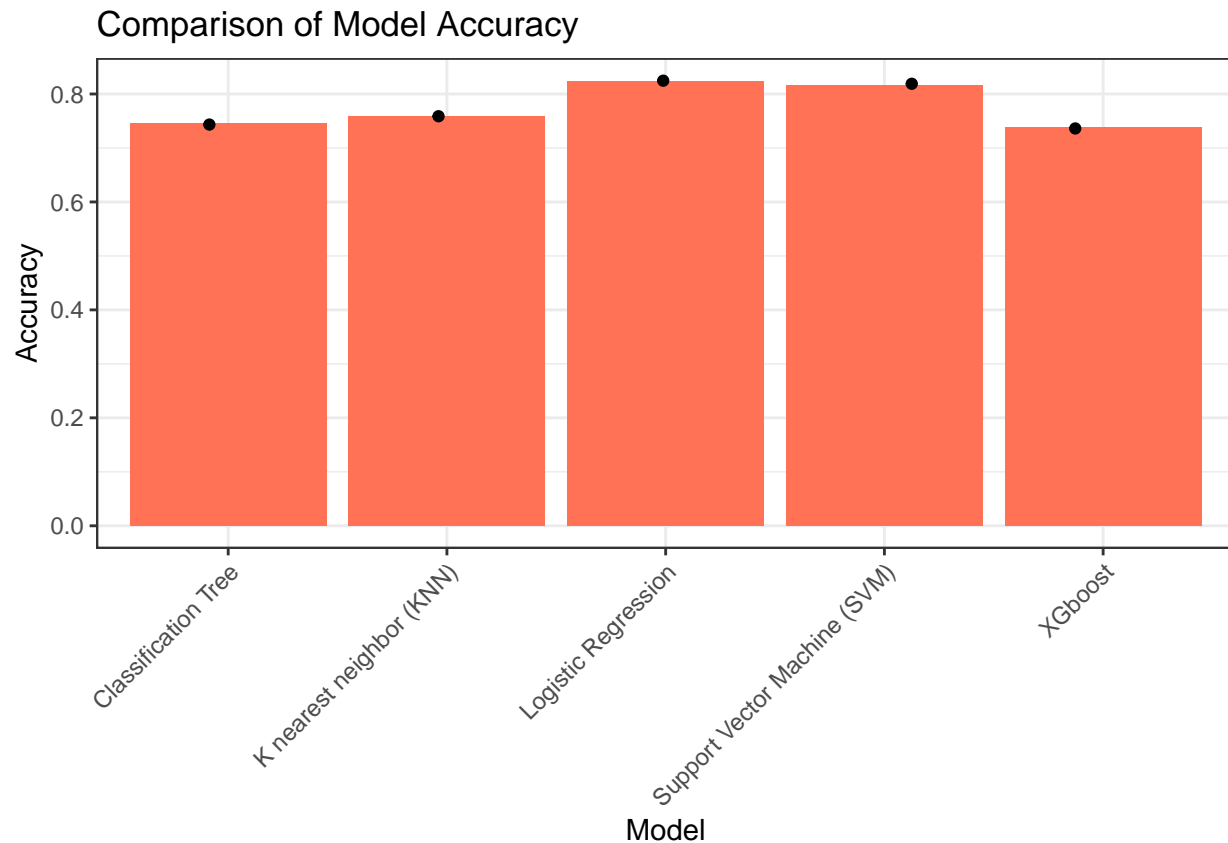
Comparing the Test results of all the models.

Results

	Sensitivity	Specificity	Precision	Recall	F1
test_glm	0.6792453	0.9000000	0.7826087	0.6792453	0.7272727
test_tree	0.7904762	0.6458333	0.8300000	0.7904762	0.8097561
test_knn	0.5849057	0.8500000	0.6739130	0.5849057	0.6262626
test_xgb	0.4339623	0.9000000	0.6969697	0.4339623	0.5348837
test_svm	0.8333333	0.7777778	0.9000000	0.8333333	0.8653846

Comparing the accuracy of all the models

Model	Accuracy
Logistic Regression	0.8235294
Classification Tree	0.7450980
XGboost	0.7385621
K nearest neighbor (KNN)	0.7581699
Support Vector Machine (SVM)	0.8169935



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      -
## arch          x86_64-w64-mingw32
## 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
```

““