Diabetes Data Analysis

Introduction

The diabetes dataset is a comprehensive compilation of health-related features that aim to predict the presence of diabetes in individuals based on various attributes. Source of dataset is from MeriSkill as part of the projects for the internship programme. By delving into this dataset, we gain valuable insights into factors that may contribute to diabetes occurrence, paving the way for a better understanding and potential preventive strategies. Specifically, this analysis centers on the Pima Indian diabetes dataset, focusing on female patients of Pima Indian origin aged 21 and above. The objective here is to extract meaningful insights and provide informed recommendations.

Overview of the Data

The dataset encompasses a range of health-related attributes, including pregnancies, glucose levels, blood pressure, skin thickness, insulin, BMI (Body Mass Index), pedigree function, and age. These attributes are crucial in understanding and predicting the presence of diabetes. Notably, this dataset now features a combination of numerical and categorical variables. Initially from source, it features only numerical values. These values were categorized for better understanding and readability.

Data Importing & Cleaning & Inspecting

Import dataset variable name ‘diabetes’ is the name of the dataset

```
file_path <- "C:/Users/PCC/Downloads/Project 2 MeriSKILL/diabetes.csv"

diabetes <- read.csv(file_path)
```

Load necessary libraries

```
library(dplyr)
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
library(corrr)
library(ggplot2)
library(corrplot)
## corrplot 0.92 loaded
library(tidyr)
```

```r
library(highcharter)
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
## Highcharts (www.highcharts.com) is a Highsoft software product which is
## not free for commercial and Governmental use
library(GGally)
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
library(e1071)
library(stringr)
library(reshape2)
##
## Attaching package: 'reshape2'
## The following object is masked from 'package:tidyr':
##
##     smiths
library(pROC)
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
library(randomForest)
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:dplyr':
##
##     combine
library(caret)
```

```
## Loading required package: lattice
```

```r
library(PerformanceAnalytics)
```

```
## Loading required package: xts

## Loading required package: zoo

##

## Attaching package: 'zoo'

## The following objects are masked from 'package:base':

##

##     as.Date, as.Date.numeric

##

## ######################### Warning from 'xts' package ###################
#######

## #
#

## # The dplyr lag() function breaks how base R's lag() function is suppose
d to  #

## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or
#

## # source() into this session won't work correctly.
#

## #
#

## # Use stats::lag() to make sure you're not using dplyr::lag(), or you ca
n add #

## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop
#

## # dplyr from breaking base R's lag() function.
#

## #
#

## # Code in packages is not affected. It's protected by R's namespace mech
anism #

## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warn
ing.  #

## #
#

## ###########################################################################
#######

##

## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':

##

##     first, last
```

```
##
## Attaching package: 'PerformanceAnalytics'
## The following objects are masked from 'package:e1071':
##
##     kurtosis, skewness
## The following object is masked from 'package:graphics':
##
##     legend
```

```
library(summarytools)
library(knitr)
library(lattice)
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

View and Inspect the dataset

```
head(diabetes, 10)
```

```
##    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
## 7            3      78            50            32      88 31.0
## 8           10     115             0             0       0 35.3
## 9            2     197            70            45     543 30.5
## 10           8     125            96             0       0  0.0
##    DiabetesPedigreeFunction Age Outcome
## 1                     0.627  50       1
## 2                     0.351  31       0
## 3                     0.672  32       1
## 4                     0.167  21       0
## 5                     2.288  33       1
## 6                     0.201  30       0
## 7                     0.248  26       1
## 8                     0.134  29       0
## 9                     0.158  53       1
## 10                    0.232  54       1
```

Structure of the dataset

```
str(diabetes)
```

```
## '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 ...
```

Understanding the descriptive summary

```
descriptive_summary <- diabetes %>%
  select_if(is.numeric) %>%
  descr()
print(descriptive_summary)
```

```
## Descriptive Statistics
## diabetes
## N: 768
##
##                        Age   BloodPressure      BMI   DiabetesPedigreeFu
nction   Glucose   Insulin
## ----------------- -------- --------------- -------- --------------------
------ --------- ---------
##              Mean    33.24           69.11    31.99
0.47    120.89     79.80
##           Std.Dev    11.76           19.36     7.88
0.33     31.97    115.24
##               Min    21.00            0.00     0.00
0.08      0.00      0.00
##                Q1    24.00           62.00    27.30
0.24     99.00      0.00
##            Median    29.00           72.00    32.00
0.37    117.00     30.50
```

```
##                 Q3    41.00            80.00    36.60
0.63    140.50    127.50

##                Max    81.00           122.00    67.10
2.42    199.00    846.00

##                MAD    10.38            11.86     6.82
0.25     29.65     45.22

##                IQR    17.00            18.00     9.30
0.38     41.25    127.25

##                 CV     0.35             0.28     0.25
0.70      0.26      1.44

##           Skewness     1.13            -1.84    -0.43
1.91      0.17      2.26

##        SE.Skewness     0.09             0.09     0.09
0.09      0.09      0.09

##           Kurtosis     0.62             5.12     3.24
5.53      0.62      7.13

##            N.Valid   768.00           768.00   768.00
768.00    768.00    768.00

##          Pct.Valid   100.00           100.00   100.00
100.00    100.00    100.00

##

## Table: Table continues below

##

##

##

##                      Outcome   Pregnancies   SkinThickness

## ----------------- --------- ------------- ---------------

##              Mean      0.35          3.85           20.54

##           Std.Dev      0.48          3.37           15.95

##               Min      0.00          0.00            0.00

##                Q1      0.00          1.00            0.00

##            Median      0.00          3.00           23.00

##                Q3      1.00          6.00           32.00

##               Max      1.00         17.00           99.00

##               MAD      0.00          2.97           17.79

##               IQR      1.00          5.00           32.00

##                CV      1.37          0.88            0.78

##          Skewness      0.63          0.90            0.11

##       SE.Skewness      0.09          0.09            0.09

##          Kurtosis     -1.60          0.14           -0.53

##           N.Valid    768.00        768.00          768.00
##         Pct.Valid    100.00        100.00           100.00
```

Understanding and Plotting the correlation (Correlation between each variables)

Selecting only numeric columns to analyze the Correlation between variables

```
diabetes <- diabetes[, sapply(diabetes, is.numeric)]

correlation_matrix <- cor(diabetes)[,-9:-12]

print(correlation_matrix)
```

```
##                          Pregnancies    Glucose BloodPressure SkinThickn
ess

## Pregnancies               1.00000000 0.12945867    0.14128198   -0.08167
177

## Glucose                   0.12945867 1.00000000    0.15258959    0.05732
789

## BloodPressure             0.14128198 0.15258959    1.00000000    0.20737
054

## SkinThickness            -0.08167177 0.05732789    0.20737054    1.00000
000

## Insulin                  -0.07353461 0.33135711    0.08893338    0.43678
257

## BMI                       0.01768309 0.22107107    0.28180529    0.39257
320

## DiabetesPedigreeFunction -0.03352267 0.13733730    0.04126495    0.18392
757

## Age                       0.54434123 0.26351432    0.23952795   -0.11397
026

## Outcome                   0.22189815 0.46658140    0.06506836    0.07475
223

##                              Insulin        BMI DiabetesPedigreeFunction

## Pregnancies              -0.07353461 0.01768309              -0.03352267

## Glucose                   0.33135711 0.22107107               0.13733730

## BloodPressure             0.08893338 0.28180529               0.04126495

## SkinThickness             0.43678257 0.39257320               0.18392757

## Insulin                   1.00000000 0.19785906               0.18507093

## BMI                       0.19785906 1.00000000               0.14064695

## DiabetesPedigreeFunction  0.18507093 0.14064695               1.00000000

## Age                      -0.04216295 0.03624187               0.03356131

## Outcome                   0.13054795 0.29269466               0.17384407

##                                  Age

## Pregnancies               0.54434123

## Glucose                   0.26351432
## BloodPressure             0.23952795
## SkinThickness            -0.11397026
## Insulin                  -0.04216295
## BMI                       0.03624187
## DiabetesPedigreeFunction  0.03356131
## Age                       1.00000000
## Outcome                   0.23835598
```

Understanding the relationships between various factors in the context of diabetes is crucial for comprehending its development and progression. Let's discuss how each of these variables - pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, age, and outcome (positive or negative for diabetes) - can affect each other:

1. **Pregnancies:**
   - The number of pregnancies a woman has had can influence the risk of developing diabetes. Multiple pregnancies (e.g., gestational diabetes) can increase insulin resistance and glucose levels.
2. **Glucose:**
   - Elevated glucose levels (hyperglycemia) can result from insulin resistance or impaired insulin production, leading to diabetes.
   - Glucose levels can be affected by factors like insulin sensitivity, diet, exercise, and the body's ability to utilize glucose for energy.
3. **Blood Pressure:**
   - High blood pressure (hypertension) is often associated with diabetes. Insulin resistance can contribute to both conditions.
   - Hypertension can exacerbate diabetes-related complications and increase the risk of heart disease.
4. **Skin Thickness:**
   - Skin thickness may not directly affect diabetes but can be a factor in assessing overall health and body composition, which can influence BMI and insulin resistance.
5. **Insulin:**
   - Insulin is a key hormone that regulates glucose metabolism. In individuals with insulin resistance, the body's cells don't respond effectively to insulin, resulting in higher glucose levels.
   - Insulin levels can be affected by factors like BMI, diet, physical activity, and insulin production by the pancreas.
6. **BMI (Body Mass Index):**
   - Higher BMI is often associated with insulin resistance and an increased risk of type 2 diabetes. Excess body fat, especially around the abdomen, can lead to insulin resistance and elevated glucose levels.
   - BMI is influenced by factors like diet, exercise, genetics, and overall lifestyle.
7. **Diabetes Pedigree Function:**
   - The diabetes pedigree function assesses the genetic predisposition for diabetes by considering the family history. A strong family history of diabetes can indicate a higher risk.
   - Genetic factors can influence insulin production, sensitivity, and overall diabetes risk.
8. **Age:**
   - Age is a significant risk factor for diabetes. The risk increases with age, particularly after the age of 45.
   - Aging affects insulin sensitivity, glucose regulation, and the body's ability to manage blood pressure, all of which can contribute to diabetes.
9. **Outcome (Positive or Negative for Diabetes):**

- The outcome variable, indicating whether an individual is positive or negative for diabetes, is directly affected by the interplay of the above factors.
- Positive outcome is associated with elevated glucose levels, insulin resistance, higher BMI, hypertension, and other diabetes risk factors.

In summary, these variables are interconnected and contribute to the development and progression of diabetes. Understanding their relationships helps in better diabetes prediction, management, and preventive strategies. Lifestyle modifications, regular monitoring, and early intervention can significantly influence these factors and mitigate diabetes risk.

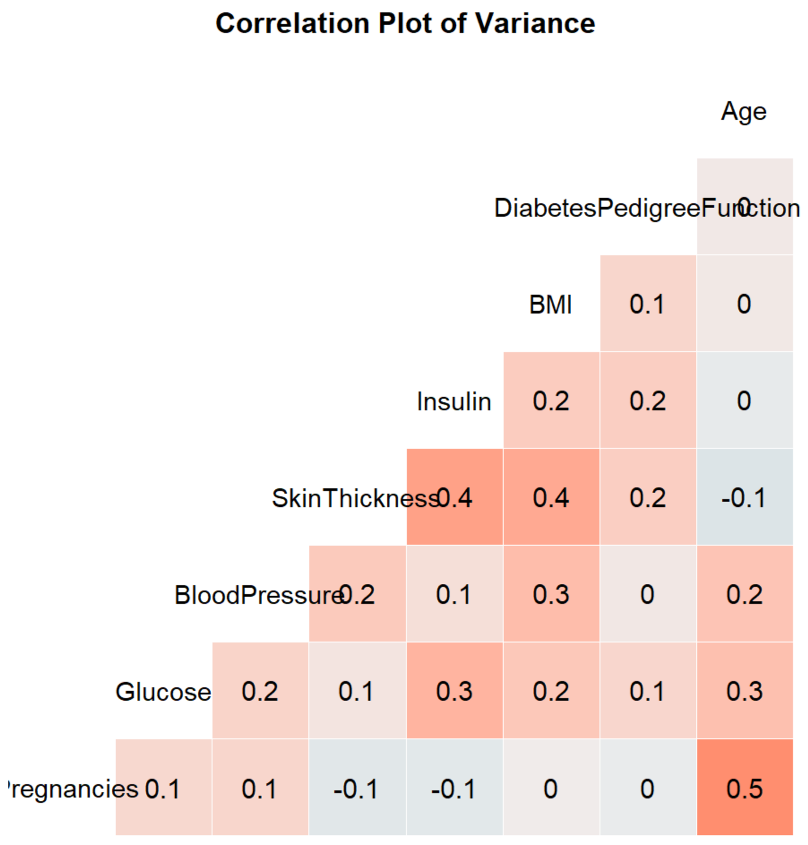
Dataset Reshaped

Renaming variables for better explanation, change the column name ‘Outcome’ to ‘OutcomeStatus’

```
colnames(diabetes)[9] <- "OutcomeStatus"

diabetes$OutcomeStatus <- as.factor(diabetes$OutcomeStatus)

levels(diabetes$OutcomeStatus) <- c("Negative","Positive")
```

Creating the correlation chart

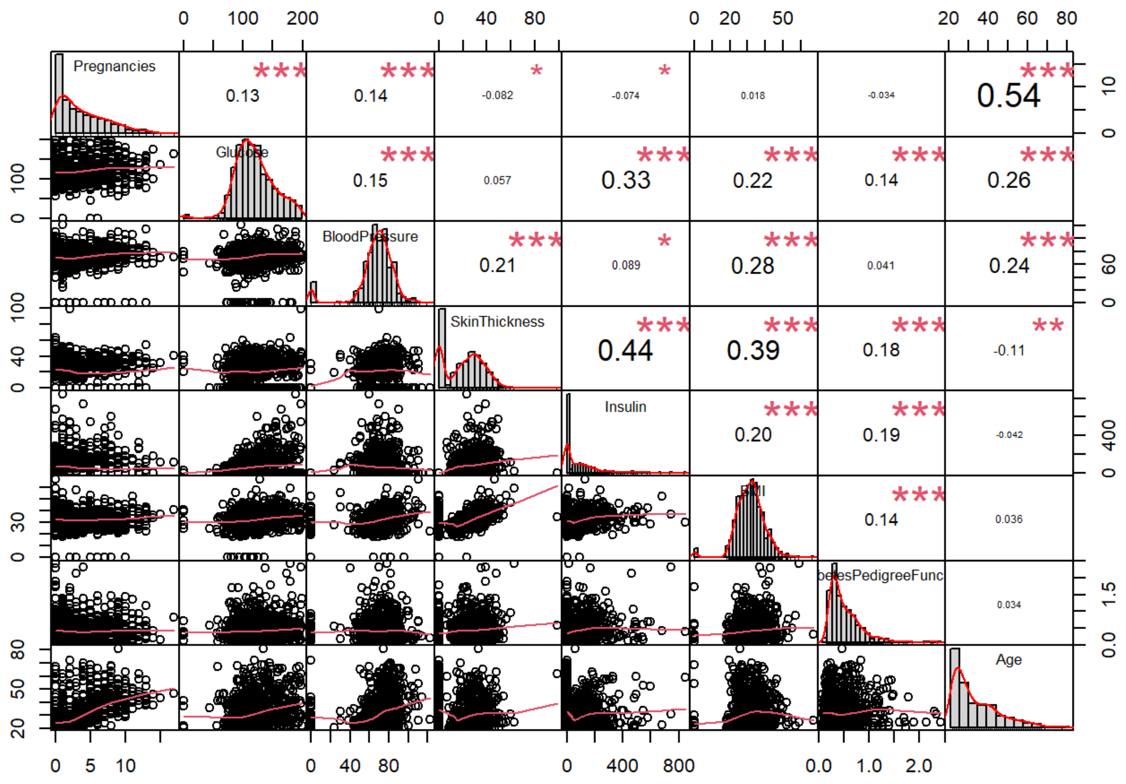
```
ggcorr(diabetes, name = "corr", label = TRUE)+ theme(legend.position="none"
)+ labs(title="Correlation Plot of Variance")+ theme(plot.title=element_tex
t(face='bold',color='black', hjust=0.5,size=12))
```



See the plot By ggcorr, we can see high correlation in above variance. Pregnancies & Age : 0.5 => About 50% correlated to each other. SkinThickness & Insulin, & BMI : 0.4 => About 40% correlated to each other

Correlation of Variance Chart – Selecting only numeric columns

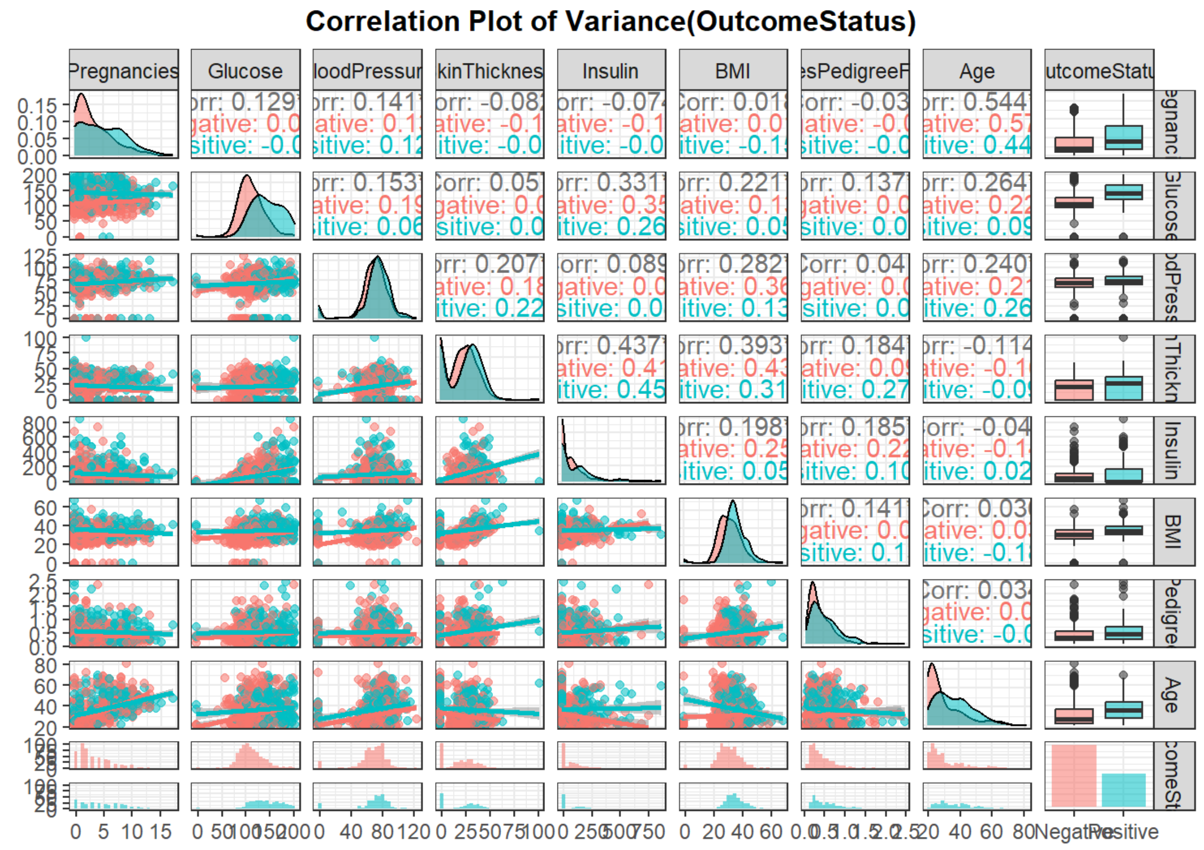
```
numeric_columns <- diabetes[, sapply(diabetes, is.numeric)]

chart.Correlation(numeric_columns, histogram=TRUE, col="grey10", pch=1, mai
n="Chart Correlation of Variance")
```



See the relationship between each variables (OutcomeStatus included) via ggpairs The figures for Pregnancies, Glucose, Age seem different according to outcome status.

```
ggpairs(diabetes, aes(color=OutcomeStatus, alpha=0.75), lower=list(continuo
us="smooth"))+ theme_bw()+ labs(title="Correlation Plot of Variance(Outcome
Status)")+ theme(plot.title=element_text(face='bold',color='black',hjust=0.
5,size=12))
```

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

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

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

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

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

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

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

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



Exploratory Data Analysis

In the realm of Exploratory Data Analysis (EDA), we delve into understanding the distribution and interrelationships of the various features. Key observations highlight a higher prevalence of diabetes with advancing age and increasing BMI. Glucose levels and blood pressure also emerge as influential factors. To enhance model performance, data preprocessing is undertaken, categorizing age into groups and BMI into specific categories.

Creating categorical values for the Age column (AgeGroup) and BMI column (BMIcategory)

```
diabetes <- diabetes %>% mutate( AgeGroup = case_when( Age < 30 ~ 'Under 30
', Age >= 30 & Age < 40 ~ '30-39', Age >= 40 & Age < 50 ~ '40-49', Age >= 5
0 & Age < 60 ~ '50-59', TRUE ~ '60 and above' ), BMIcategory = case_when( B
MI < 18.5 ~ 'Underweight', BMI >= 18.5 & BMI < 24.9 ~ 'Normal Weight', BMI
>= 25 & BMI < 29.9 ~ 'Overweight', TRUE ~ 'Obese' ) )
```

Understanding the class distribution

```
class_distribution <- diabetes %>% group_by(OutcomeStatus) %>% summarise(cl
ass_count = n())

class_distribution
```

```
## # A tibble: 2 × 2

##   OutcomeStatus class_count

##   <fct>               <int>

## 1 Negative              500
## 2 Positive              268
```

We have 500 patients who are diagnosed as not diabetic and 268 patients with diabetes.

Understanding the distribution of the Age and Glucose levels

```
age_distribution <- diabetes %>% group_by(AgeGroup) %>% summarise(count = n
())

kable(age_distribution)
```

AgeGroup	count
30-39	165
40-49	118
50-59	57
60 and above	32
Under 30	396

```
glucose_level_distribution <- diabetes %>% mutate( glucose_level = case_whe
n( Glucose < 100 ~ 'Normal', Glucose >= 100 & Glucose < 126 ~ 'Pre-Diabetic
', TRUE ~ 'Diabetic' ) ) %>% group_by(glucose_level) %>% summarise(count =
n())

kable(glucose_level_distribution)
```

glucose_level	count
Diabetic	297
Normal	197
Pre-Diabetic	274

Explore the relationship between Age and Pregnancy count

```
age_pregnancy_relationship <- diabetes %>% group_by(AgeGroup) %>% summarise
(avg_pregnancy_count = mean(Pregnancies))

kable(age_pregnancy_relationship)
```

AgeGroup	avg_pregnancy_count
30-39	4.987879
40-49	7.050847
50-59	6.596491
60 and above	5.031250
Under 30	1.921717

Check average Insulin levels for different Age Groups

```
average_insulin_agegroup <- diabetes %>% group_by(AgeGroup) %>% summarise(a
vg_insulin = mean(Insulin))

kable(average_insulin_agegroup)
```

AgeGroup	avg_insulin
30-39	77.81212
40-49	63.89831
50-59	109.40351
60 and above	39.09375
Under 30	84.39394

Analyze average Blood Pressure for different Glucose levels

```r
average_bloodpressure_glucose <- diabetes %>% mutate( glucose_level = case_
when( Glucose < 100 ~ 'Normal', Glucose >= 100 & Glucose < 126 ~ 'Pre-Diabe
tic', TRUE ~ 'Diabetic' ) ) %>% group_by(glucose_level) %>% summarise(avg_b
loodpressure = mean(BloodPressure))

kable(average_bloodpressure_glucose)
```

glucose_level	avg_bloodpressure
Diabetic	72.77441
Normal	64.60914
Pre-Diabetic	68.36131

Average Age for each Outcome (diabetic or not)

```r
average_age_Outcome <- diabetes %>% group_by(OutcomeStatus) %>% summarise(a
vg_age = mean(Age))
```

Average BMI for each Outcome (diabetic or not)

```r
average_bmi_Outcome <- diabetes %>% group_by(OutcomeStatus) %>% summarise(a
vg_bmi = mean(BMI))

average_bmi_Outcome
```

```
## # A tibble: 2 × 2
##   OutcomeStatus avg_bmi
##   <fct>           <dbl>
## 1 Negative         30.3
## 2 Positive         35.1
```

Explore average Glucose levels for each Age Group

```r
average_glucose_agegroup <- diabetes %>% group_by(AgeGroup) %>% summarise(a
vg_glucose = mean(Glucose))

kable(average_glucose_agegroup)
```

AgeGroup	avg_glucose
30-39	125.3091
40-49	124.6441
50-59	140.2807
60 and above	138.2500
Under 30	113.7449

Modeling

Several predictive models were employed, including Logistic Regression, Random Forest, and Gradient Boosting. Logistic Regression provided a good baseline, while Random Forest and Gradient Boosting demonstrated superior predictive power, likely due to their ability to capture complex relationships within the data.

Model Evaluation

Models were evaluated based on accuracy, sensitivity, specificity, and area under the ROC curve (AUC). Random Forest and Gradient Boosting exhibited higher accuracy and AUC, making them suitable choices for diabetes prediction.

Hyperparameter Tuning

Hyperparameters were tuned to optimize model performance. **RF** uses decision trees and aggregates their predictions. **SVM** uses decision boundary to separate classes and maximize the margin between them. **GBM** typically uses decision trees sequentially, with each one trying to correct the errors made by the previous one

Splitting the dataset for testing

Make test & train dataset, shuffling the diabetes data (100%) then make train dataset (80%) and test dataset (20%)

Set seed for reproducibility

```
set.seed(123)

# Generate a random index to split the data into training (80%) and testing
(20%)

index <- sample(1:nrow(diabetes), 0.8 * nrow(diabetes))

train_data <- diabetes[index, ]

test_data <- diabetes[-index, ]
```

Check the proportion of diabetes (Benign / Malignant)

Calculating the proportion of each class

Train data

```
class_proportions_train <- table(train_data$OutcomeStatus) / nrow(train_dat
a)

class_proportions_train
```

```
##

##   Negative  Positive

## 0.6482085 0.3517915
```

In the train data, we have outcome status for patients with diabetes to be 35% and non diabetic patients are 65%

Test data

```
class_proportions_test <- table(test_data$OutcomeStatus) / nrow(test_data)

class_proportions_test
```

```
##

##   Negative  Positive
## 0.6623377 0.3376623
```

In the test data, we have outcome status for patients with diabetes to be 34% and non diabetic patients are 66%

This is the dimension of the resulting sets

```
cat("Number of rows in training set:", nrow(train_data), "\n")
## Number of rows in training set: 614
cat("Number of rows in testing set:", nrow(test_data), "\n")
## Number of rows in testing set: 154
```

Applying Machine Learning Methods - Building a Predictive Model

Building GBM Model

Gradient Boosting Machine (GBM) is another powerful ensemble learning technique widely used for both classification and regression problems. GBM builds multiple weak learners (usually decision trees) sequentially, with each subsequent learner focusing on correcting the errors or residuals of the previous ones.

GBM works by minimizing a loss function (e.g., mean squared error for regression, log loss for classification) using gradient descent. It pays more attention to misclassified data points, leading to an improved model with each iteration.

One of the advantages of GBM is its ability to capture complex relationships in the data and handle missing values efficiently. However, GBM can be sensitive to overfitting, and hyperparameters tuning is crucial to achieve optimal performance.

GBM is widely used in various domains, including web search, ranking, recommendation systems, and more, due to its high predictive accuracy and flexibility in handling different types of data**.**

Training Gradient Boosting Model (GBM)

```
test_gbm <- gbm(OutcomeStatus~., data=train_data[,1:9], distribution="gaussian", n.trees = 10000,
                shrinkage = 0.01, interaction.depth = 4, bag.fraction=0.5, train.fraction=0.5,n.minobsinnode=10,cv.folds=3,keep.data=TRUE,verbose=FALSE,n.cores=1)
## CV: 1
## CV: 2
## CV: 3
best.iter <- gbm.perf(test_gbm, method="cv",plot.it=FALSE)
fitControl = trainControl(method="cv", number=5, returnResamp="all")
```

```
gbm_model = train(OutcomeStatus~., data=train_data[,1:9], method="gbm", dis
tribution="bernoulli", trControl=fitControl, verbose=F, tuneGrid=data.frame
(.n.trees=best.iter, .shrinkage=0.01, .interaction.depth=1, .n.minobsinnode
=1))

# Predict on the test set

pre_gbm <- predict(gbm_model, test_data[,-9])

# Create a confusion matrix

cm_gbm <- confusionMatrix(pre_gbm, test_data$OutcomeStatus)

cm_gbm
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Negative Positive
##   Negative       89       27
##   Positive       13       25
##
##                Accuracy : 0.7403
##                  95% CI : (0.6635, 0.8075)
##     No Information Rate : 0.6623
##     P-Value [Acc > NIR] : 0.02326
##
##                   Kappa : 0.3783
##
##  Mcnemar's Test P-Value : 0.03983
##
##             Sensitivity : 0.8725
##             Specificity : 0.4808
##          Pos Pred Value : 0.7672
##          Neg Pred Value : 0.6579
##              Prevalence : 0.6623
##          Detection Rate : 0.5779
##    Detection Prevalence : 0.7532
##       Balanced Accuracy : 0.6767
##
##        'Positive' Class : Negative
##
```

Building RF Model

Random Forest (RF) is a popular and powerful ensemble learning technique used for both classification and regression tasks. It operates by constructing a multitude of decision trees during the training phase. Each tree in the forest independently predicts the output, and for classification, the mode (most frequent) prediction among the trees is taken, while for regression, the mean prediction is considered.

The key strength of Random Forest lies in its ability to handle overfitting, make accurate predictions, and work well with both categorical and numerical features. It's particularly robust with large datasets and high-dimensional feature spaces.

Random Forest builds each tree using a random subset of features and data points (bootstrap sampling). This randomness and diversity in the trees enhance the model's robustness and generalization capabilities. It's an efficient and versatile algorithm widely used in various domains such as finance, healthcare, marketing, and more.

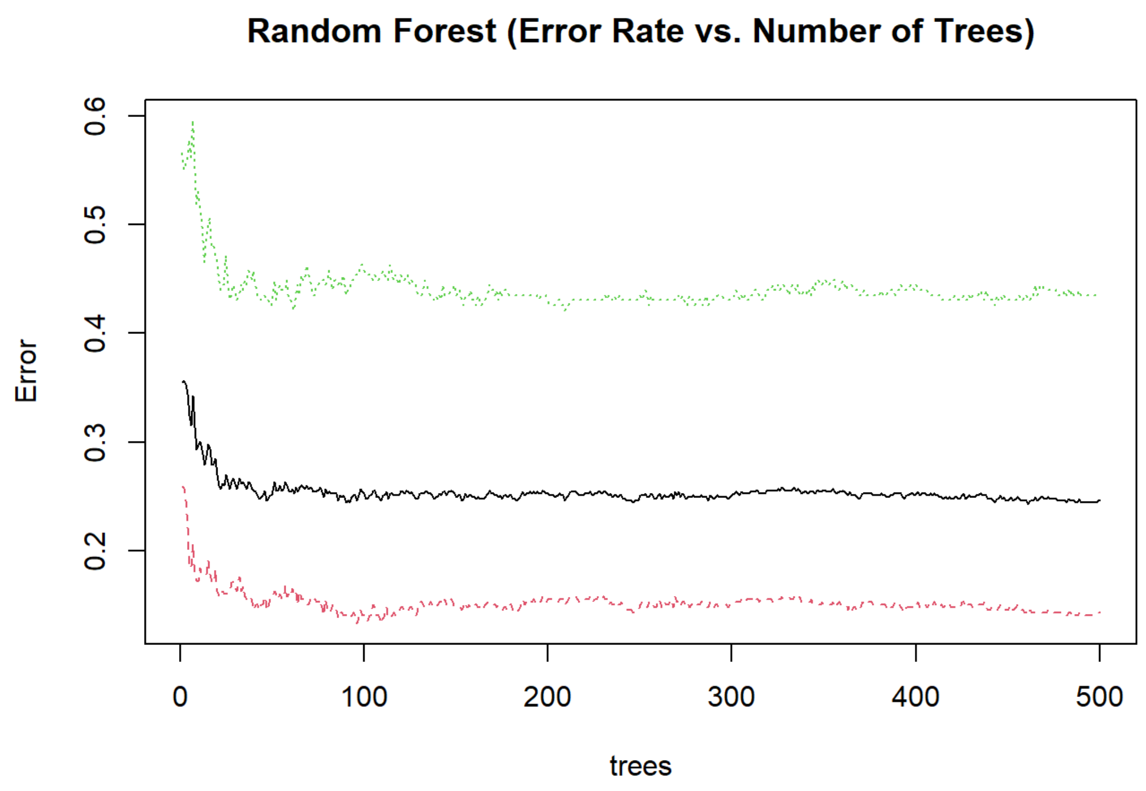
Training Random Forest Model

```
rf_model <- randomForest(OutcomeStatus ~ ., data=train_data[,1:9], ntree =
500, proximity=T, importance=T)

pre_rf <- predict(rf_model, test_data[,-9])

cm_rf <- confusionMatrix(pre_rf, test_data$OutcomeStatus)

cm_rf
```

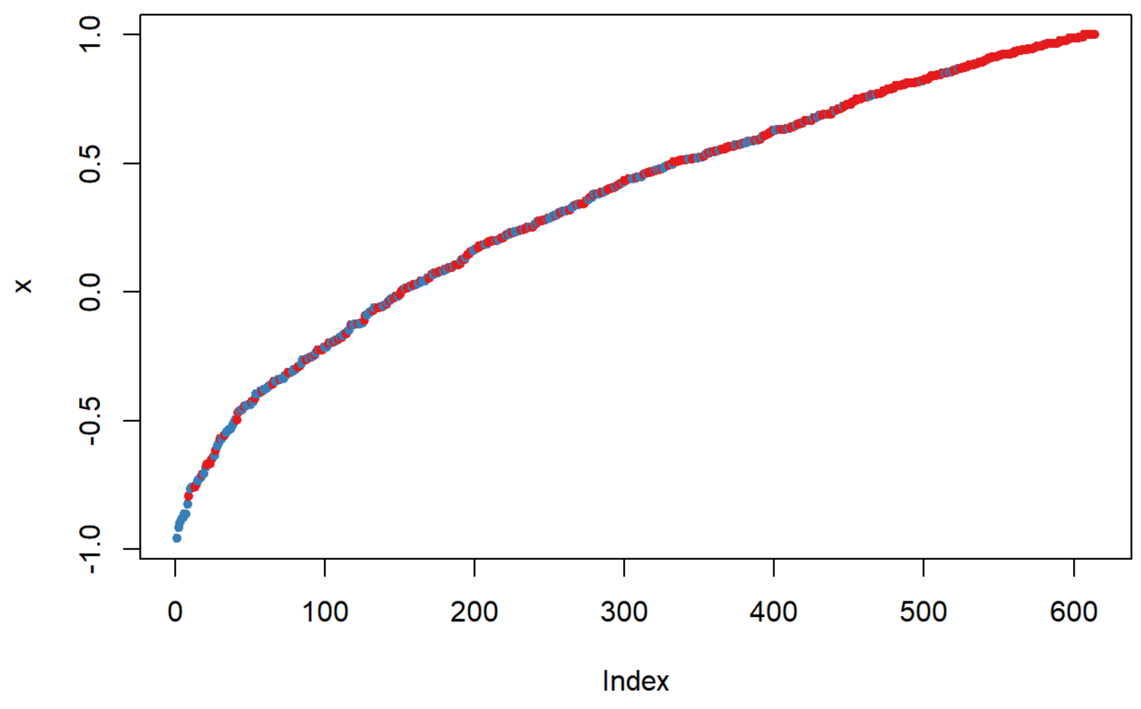
```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Negative Positive
##   Negative       90       22
##   Positive       12       30
##
##                Accuracy : 0.7792
##                  95% CI : (0.7054, 0.842)
##     No Information Rate : 0.6623
##     P-Value [Acc > NIR] : 0.001047
##
##                   Kappa : 0.482
##
##  Mcnemar's Test P-Value : 0.122713
##
##             Sensitivity : 0.8824
##             Specificity : 0.5769
##          Pos Pred Value : 0.8036
##          Neg Pred Value : 0.7143
##                Prevalence : 0.6623
##            Detection Rate : 0.5844
##      Detection Prevalence : 0.7273
##         Balanced Accuracy : 0.7296
##
##          'Positive' Class : Negative
##
```

RF prediction plot

```
plot(rf_model, main="Random Forest (Error Rate vs. Number of Trees)")
```



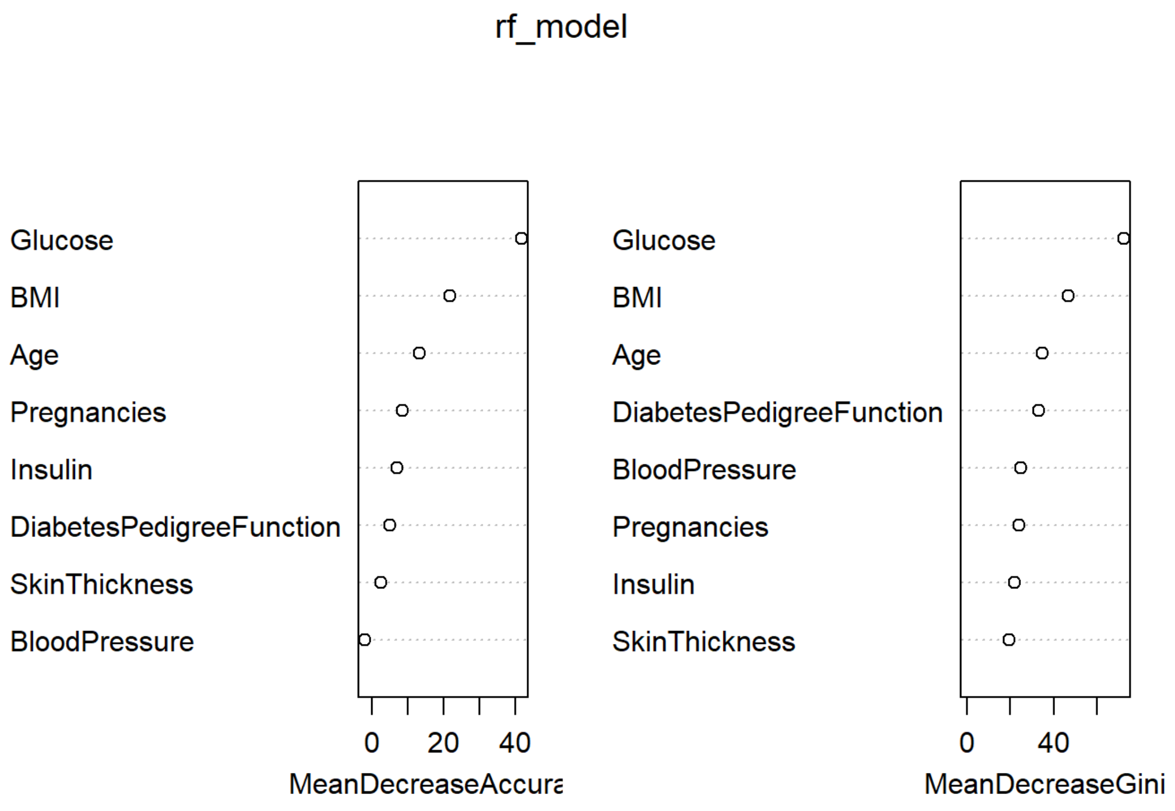
```
plot(margin(rf_model,test$OutcomeStatus))
```



Plotting feature importance

```
importance(rf_model)
```

```
##                           Negative   Positive MeanDecreaseAccuracy
## Pregnancies               7.364144  3.3735861             8.588822
## Glucose                  32.591032 32.1052481            41.831643
## BloodPressure             0.384922 -3.4820665            -2.039149
## SkinThickness             2.631580  0.5775054             2.589700
## Insulin                   6.961793  2.4584237             6.977295
## BMI                      12.343666 18.9597405            21.745644
## DiabetesPedigreeFunction  4.036353  3.0359069             4.942292
## Age                       8.967540  7.3763076            13.240773
##                          MeanDecreaseGini
## Pregnancies                      23.87432
## Glucose                          72.45131
## BloodPressure                    24.95430
## SkinThickness                    19.28856
## Insulin                          22.12037
## BMI                              46.92916
## DiabetesPedigreeFunction         33.16958
## Age                                  34.76547
varImpPlot(rf_model)
```



Here we have to extract confusion matrix elements (because cm_rf is a list and not matrix)

```
TN <- cm_rf$table[1,1]
FP <- cm_rf$table[1,2]
FN <- cm_rf$table[2,1]
TP <- cm_rf$table[2,2]

# Calculate accuracy, sensitivity, and specificity
accuracy_rf <- (TP + TN) / (TP + TN + FP + FN)
sensitivity_rf <- TP / (TP + FN)
specificity_rf <- TN / (TN + FP)

# print metrics
print("Random Forest Model Metrics:")
## [1] "Random Forest Model Metrics:"
print(paste("Accuracy:", accuracy_rf))
## [1] "Accuracy: 0.779220779220779"
print(paste("Sensitivity:", sensitivity_rf))
```

```
## [1] "Sensitivity: 0.714285714285714"
print(paste("Specificity:", specificity_rf))
## [1] "Specificity: 0.803571428571429"
```

Convert pre_rf to binary numeric (0 or 1) based on OutcomeStatus levels

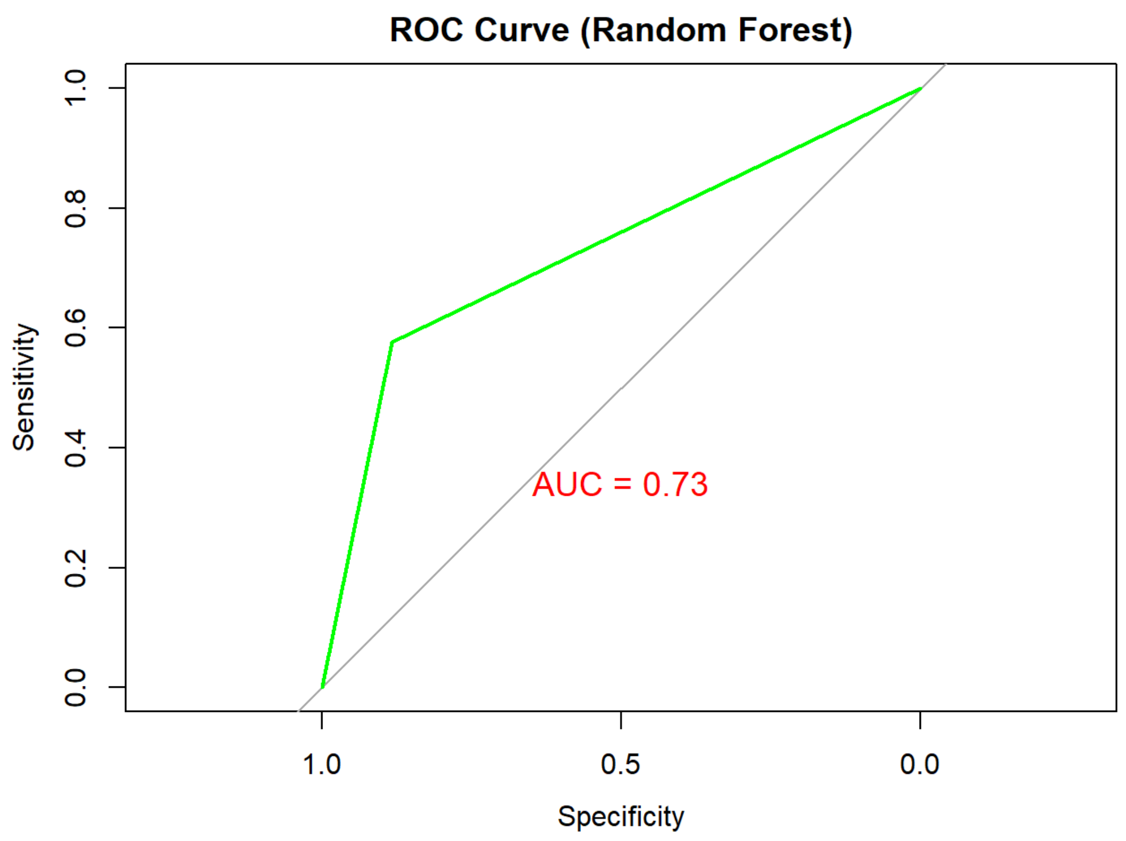
```
pre_rf_binary <- ifelse(pre_rf == "Negative", 0, 1)
```

AUC for Random Forest

```
roc_obj_rf <- roc(test_data$OutcomeStatus, pre_rf_binary)
## Setting levels: control = Negative, case = Positive
## Setting direction: controls < cases
auc_score_rf <- auc(roc_obj_rf)
print(sprintf("AUC for Random Forest is %.4f", auc_score_rf))
## [1] "AUC for Random Forest is 0.7296"
```

ROC curve for Random Forest

```
plot(roc_obj_rf, main = "ROC Curve (Random Forest)", col = "green", lwd = 2
)
text(0.5, 0.3, paste("AUC =", round(auc_score_rf, 2)), adj = c(0.5, -0.5),
col = "red", cex = 1.2)
```



Building the SVM Model

Support Vector Machine (SVM) is a powerful and versatile machine learning algorithm used for classification and regression tasks. The primary objective of SVM in classification is to find the optimal hyperplane that maximizes the margin between different classes in the feature space. This hyperplane is positioned in such a way that it best separates the data points of one class from those of the other classes.

In the context of classification, SVM works by transforming the input data into a high-dimensional feature space where it attempts to find a hyperplane that separates the data into distinct classes. It aims to maximize the margin, which is the distance between the hyperplane and the nearest data point of any class. This approach not only helps in accurate classification but also ensures robustness against new data points.

In regression tasks, SVM aims to find a hyperplane that best fits the data, predicting continuous output values. SVM's versatility, robustness, and ability to handle complex data make it a popular choice in many machine learning applications.

Training the model Choose 'gamma, cost' which shows best predict performance in SVM

```
gamma <- seq(0,0.1,0.005)

cost <- 2^(0:5)

parms <- expand.grid(cost=cost, gamma=gamma)

acc_test <- numeric()

accuracy1 <- NULL; accuracy2 <- NULL

accuracy2 <- numeric(NROW(parms))


for (i in 1:NROW(parms)) {

  # Train SVM model

  svm_model <- svm(OutcomeStatus ~ ., data = train_data[, 1:9], gamma = par
ms$gamma[i], cost = parms$cost[i])

  # Predict using SVM model

  predict_svm <- predict(svm_model, test_data[, 1:8])

  # Ensure predict_svm and test_data$OutcomeStatus are of the same length

  predict_svm <- predict_svm[1:length(test_data$OutcomeStatus)]

  # Calculate accuracy and store in the accuracy2 vector

  accuracy1 <- confusionMatrix(predict_svm, test_data$OutcomeStatus)

  accuracy2[i] <- accuracy1$overall[1]

}


acc <- data.frame(p= seq(1,NROW(parms)), cnt = accuracy2)

opt_p <- subset(acc, cnt==max(cnt))[1,]

sub <- paste("Optimal number of parameter is", opt_p$p, "(accuracy :", opt_
p$cnt,") in SVM")
```

```
hchart(acc, 'line', hcaes(p, cnt)) %>%
  hc_title(text = "Accuracy With Varying Parameters (SVM)") %>%
  hc_subtitle(text = sub) %>%
  hc_add_theme(hc_theme_google()) %>%
  hc_xAxis(title = list(text = "Number of Parameters")) %>%
  hc_yAxis(title = list(text = "Accuracy"))
```

Number of ParametersAccuracyAccuracy With Varying Parameters (SVM)Optimal number of parameter is 7 (accuracy : 0.792207792207792) in SVM0204060801001200.650.6750.70.7250.750.7750.8

Show best gamma and cost values

```
kable(paste("Best Cost :",parms$cost[opt_p$p],", Best Gamma:",parms$gamma[o
pt_p$p]))
```

x

Best Cost : 1 , Best Gamma: 0.005

Training the SVM model - Applying optimal parameters(gamma, cost) to show best predict performance in SVM

```
imp_svm_model <- svm(OutcomeStatus~., data=train_data[,1:9], cost=parms$cos
t[opt_p$p], gamma=parms$gamma[opt_p$p])

pre_imp_svm <- predict(imp_svm_model, test_data[,1:8])

cm_imp_svm <- confusionMatrix(pre_imp_svm, test_data$OutcomeStatus)

cm_imp_svm
```

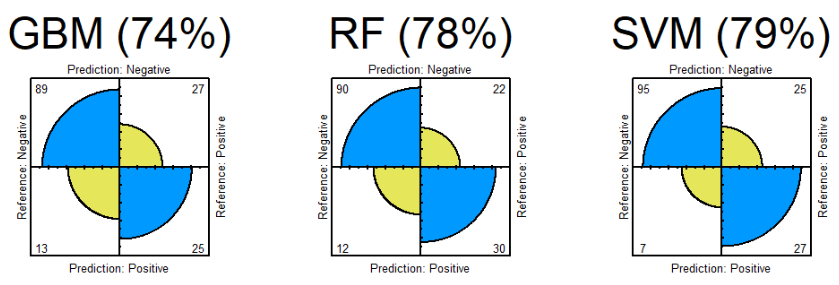
```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Negative Positive
##   Negative       95       25
##   Positive        7       27
##
##                Accuracy : 0.7922
##                  95% CI : (0.7195, 0.8533)
##     No Information Rate : 0.6623
##     P-Value [Acc > NIR] : 0.0002837
##
##                   Kappa : 0.4924
##
##  Mcnemar's Test P-Value : 0.0026540
##
```

```
##              Sensitivity : 0.9314
##              Specificity : 0.5192
##           Pos Pred Value : 0.7917
##           Neg Pred Value : 0.7941
##               Prevalence : 0.6623
##           Detection Rate : 0.6169
##     Detection Prevalence : 0.7792
##        Balanced Accuracy : 0.7253
##
##         'Positive' Class : Negative
##
```

Total Summary & Choosing Best ML

Visualize to compare the accuracy of all methods

```
col <- c("#e6e65b", "#0099ff")
par(mfrow=c(3,4))
fourfoldplot(cm_gbm$table, color = col, conf.level = 0, margin = 1, main =
paste("GBM (", round(cm_gbm$overall[1] * 100),"%)", sep =""))


fourfoldplot(cm_rf$table, color = col, conf.level = 0, margin = 1, main=pas
te("RF (",round(cm_rf$overall[1]*100),"%)",sep=""))


fourfoldplot(cm_imp_svm$table, color = col, conf.level = 0, margin = 1, mai
n=paste("SVM (",round(cm_imp_svm$overall[1]*100),"%)",sep=""))
```



Selecting the best prediction model according to high accuracy

```
opt_predict <- c(cm_gbm$table[1], cm_rf$table[1], cm_imp_svm$table[1])

names(opt_predict) <- c("gbm", "rf", "svm")

best_predict_model <- names(opt_predict)[which.max(opt_predict)]

cat("Best predict model is", names(opt_predict)[which.max(opt_predict)], "w
ith value:", max(opt_predict))
## Best predict model is svm with value: 95
```

Predicting the test dataset

Predicting outcome status for the 22nd patient

```
P <- test_data[5,]

kable(P)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	OutcomeStatus	AgeGroup	BMIcategory
22	8	99	84	0	0	35.4	0.388	50	Negative	50-59	Obese

Predicting outcome status for the 9th patient

```r
N <- test_data[3,]
kable(N)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	OutcomeStatus	AgeGroup	BMIcategory
9	2	197	70	45	543	30.5	0.158	53	Positive	50-59	Obese

We first have to delete the OutcomeStatus column for testing

```r
P$OutcomeStatus <- NULL
N$OutcomeStatus <- NULL
```

Test 1 Patient

- Why only one patient? In real-life scenarios, it's usually more typical to diagnose just one patient at a time.

Making function

```r
patient_OutcomeStatus_predict <- function(new, method=imp_svm_model) {
  new_pre <- predict(method, new)
  new_res <- as.character(new_pre)
  return(paste("Result: ", new_res, sep=""))
}
```

Testing Function:

Testing function and Comparing models

Utilized the 'Tuned SVM Algorithm' by default, considering it's regarded as the top predictive model. However, the best predictive model isn't always ideal. In practical situations, minimizing faulty predictions like (OutcomeStatus: Positive -> Predict: Negative) is crucial. Hence, I believe the tuned gbm mode, displaying the lowest rate (74%), is the optimal choice for predictive modeling.

```r
patient_OutcomeStatus_predict(N)
## [1] "Result: Positive"
patient_OutcomeStatus_predict(P)
## [1] "Result: Negative"
```

Testing GBM

```r
patient_OutcomeStatus_predict(N,gbm_model)
## [1] "Result: Positive"
patient_OutcomeStatus_predict(P,gbm_model)
```

```
## [1] "Result: Negative"
```

Predicting outcome status in the test dataset

Using GBM

```
sub <- data.frame(orgin_result = test_data$OutcomeStatus, predict_result =
pre_gbm, correct = ifelse(test_data$OutcomeStatus == pre_gbm, "True", "Fals
e"))

kable(sub[1:100, ], format = "html")
```

orgin_result	predict_result	correct
Positive	Positive	True
Positive	Positive	True
Positive	Positive	True
Positive	Negative	False
Negative	Negative	True
Positive	Positive	True
Negative	Negative	True
Positive	Positive	True
Negative	Negative	True
Negative	Negative	True
Positive	Positive	True
Negative	Negative	True
Negative	Negative	True
Positive	Negative	False
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Positive	Negative	False
Negative	Negative	True
Negative	Negative	True
Positive	Negative	False
Negative	Positive	False
Negative	Negative	True
Negative	Negative	True
Negative	Positive	False
Negative	Negative	True
Negative	Positive	False
Negative	Negative	True

orgin_result	**predict_result**	**correct**
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Positive	Positive	True
Positive	Positive	True
Positive	Positive	True
Negative	Positive	False
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Positive	Negative	False
Negative	Positive	False
Positive	Negative	False
Negative	Negative	True
Negative	Negative	True
Positive	Negative	False
Negative	Positive	False
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Positive	Positive	True
Positive	Positive	True
Positive	Negative	False
Positive	Positive	True
Negative	Negative	True
Positive	Negative	False
Negative	Positive	False
Positive	Negative	False
Negative	Positive	False
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Positive	Positive	True
Negative	Negative	True
Positive	Positive	True
Negative	Positive	False
Negative	Negative	True
Negative	Negative	True
Positive	Negative	False
Negative	Negative	True
Positive	Negative	False
Positive	Negative	False
Negative	Negative	True
Negative	Negative	True
Positive	Positive	True
Negative	Negative	True
Negative	Negative	True
Negative	Negative	True
Positive	Positive	True

orgin_result	**predict_result**	**correct**
Negative	Negative	True
Positive	Negative	False
Positive	Negative	False
Negative	Negative	True
Positive	Negative	False
Negative	Positive	False
Negative	Negative	True
Positive	Positive	True
Negative	Negative	True
Negative	Negative	True
Negative	Positive	False
Negative	Negative	True
Negative	Negative	True

Let's see the distribution of the outcome status ;

```
prop.table(table(sub$correct))
##
##     False      True
## 0.2597403 0.7402597
```

Of 100% of the prediction, 74% of the outcome status in the test dataset were predicted correctly while 26% were not.

Created a graph depicting the Probability Density Function (PDF).

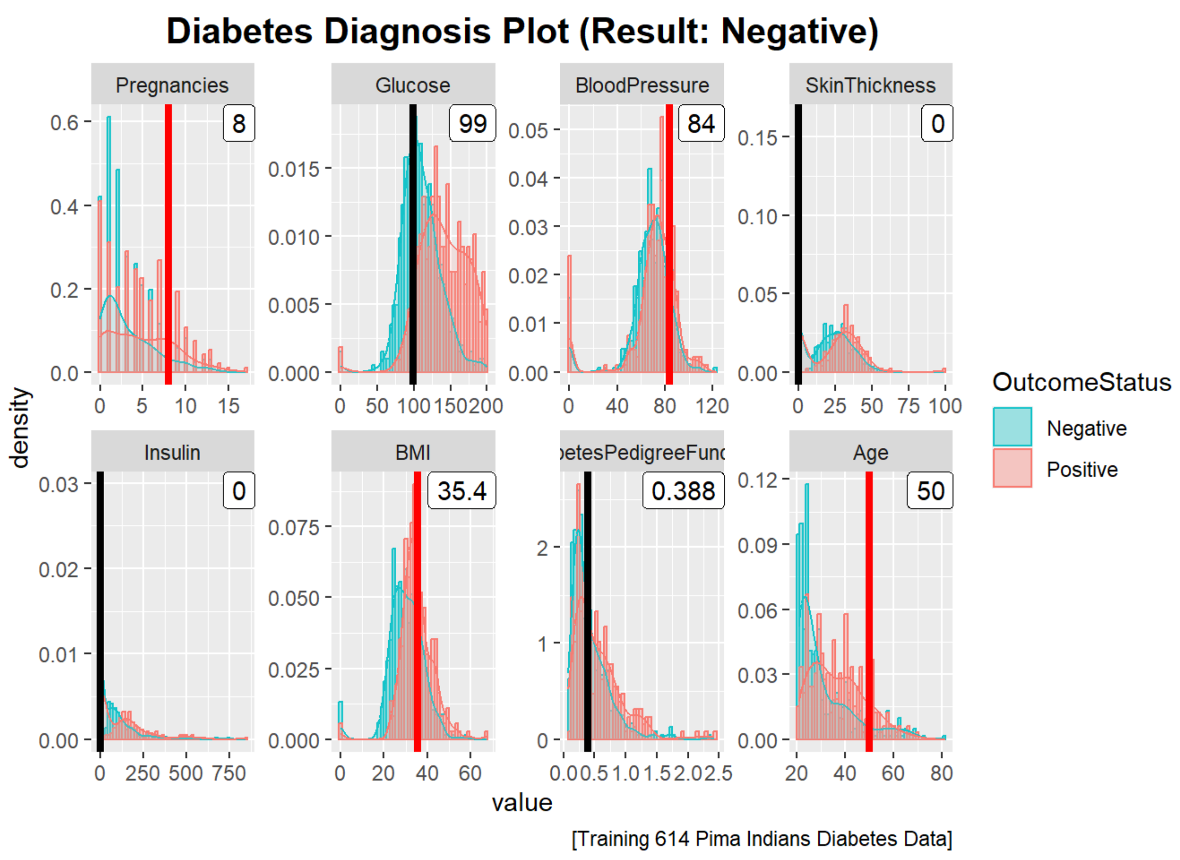
This graph was designed specifically for doctors involved in diabetes diagnosis. From the patient's perspective, I illustrated diabetes outcomes using a probability density graph, incorporating a prominent line representing the diagnosis. This allows patients to easily assess their condition. If a patient's diabetes indicator surpasses the average factor for positive outcomes, it is marked with a red line; if it falls below the average for negative outcomes, it is marked in green.

```
OutcomeStatus_summary <- function(new,data) {

  ## [a] Reshape the new dataset for ggplot
  m_train <- melt(data, id="OutcomeStatus")
  m_new <- melt(new)

  ## [b] Save mean of Malignant value
  mal_mean <- subset(data, OutcomeStatus=="Positive", select=-9)
  mal_mean <- apply(mal_mean,2,mean)

  ## [c] highlight with red colors line
  mal_col <- ifelse((round(m_new$value,3) > mal_mean), "red", "black")


  ## [d] Save titles : Main title, Patient Diagnosis

  title <- paste("Diabetes Diagnosis Plot (", patient_OutcomeStatus_predict
(new),")",sep="")


  ## ★[f] View plots highlighting values above average of malignant patien
t

  res_mean <- ggplot(m_train, aes(x=value,color=OutcomeStatus, fill=Outcome
Status))+

    geom_histogram(aes(y=..density..), alpha=0.5, position="identity", bins
=50)+

    geom_density(alpha=.2)+

    scale_color_manual(values=c("#15c3c9","#f87b72"))+

    scale_fill_manual(values=c("#61d4d6","#f5a7a1"))+

    geom_vline(data=m_new, aes(xintercept=value),

               color=mal_col, size=1.5)+

    geom_label(data=m_new, aes(x=Inf, y=Inf, label=round(value,3)), nudge_y
=2,

               vjust = "top", hjust = "right", fill="white", color="black")
+

    labs(title=title)+

    theme(plot.title = element_text(face='bold', colour='black', hjust=0.5,
size=15))+

    theme(plot.subtitle=element_text(lineheight=0.8, hjust=0.5, size=12))+

    labs(caption="[Training 614 Pima Indians Diabetes Data]")+

    facet_wrap(~variable, scales="free", ncol=4)


  ## [g] output graph

  res_mean

}
```

For Negative Outcome

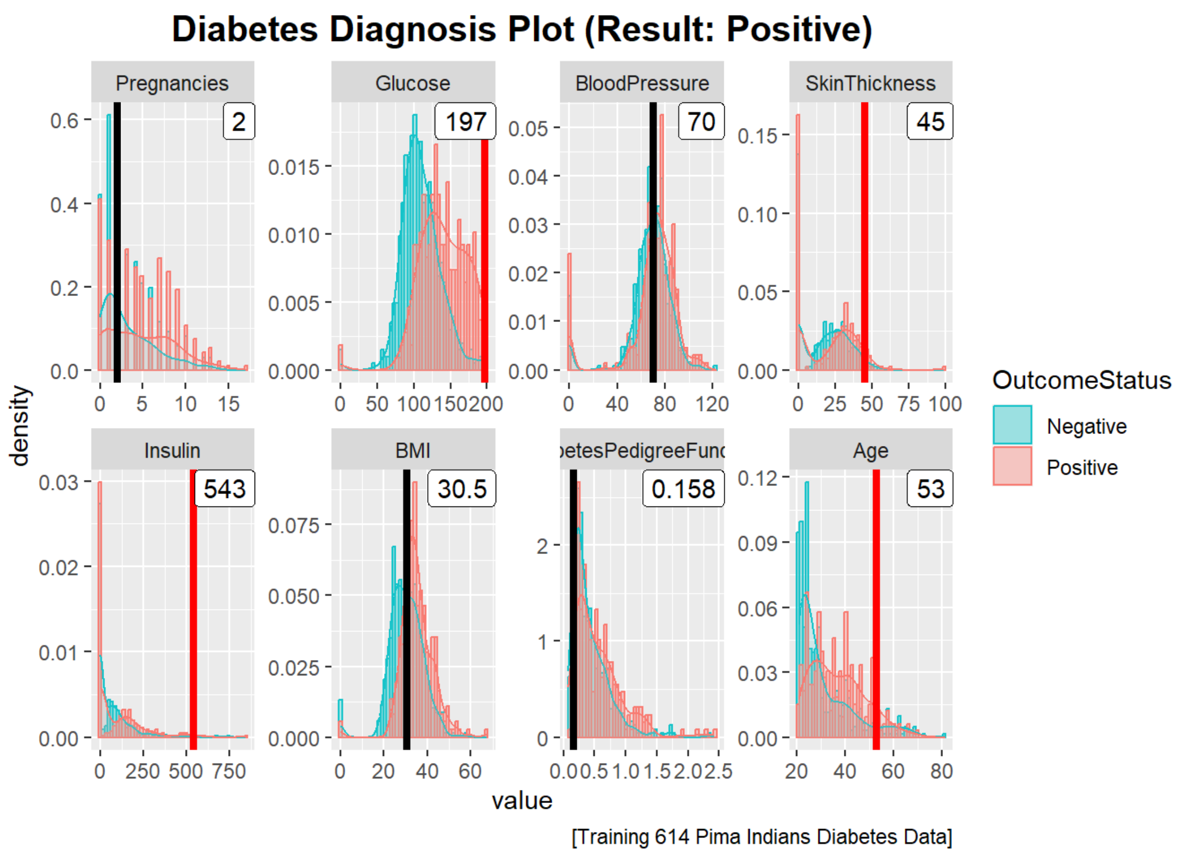
```
OutcomeStatus_summary(P, diabetes[, -c(10, 11)])
```

```
## Using AgeGroup, BMIcategory as id variables
```



For Positive Outcome

```
OutcomeStatus_summary(N, diabetes[, -c(10, 11)])
## Using AgeGroup, BMIcategory as id variables
```



Insights

1. **Top Features:**
   - Glucose level is the most important feature, indicating higher glucose levels significantly affect diabetes prediction.
   - BMI (Body Mass Index) is crucial, suggesting that individuals with higher BMI are more likely to be predicted as diabetic.
   - Age is an influential factor, indicating a correlation between age and diabetes prediction.
   - Pregnancy can impact the likelihood of developing diabetes, with multiple pregnancies (such as gestational diabetes) potentially leading to higher insulin resistance and elevated glucose levels.
   - Insulin levels play a significant role, suggesting higher insulin levels are associated with diabetes prediction.
   - Blood pressure is important, implying individuals with higher blood pressure are more likely to be predicted as diabetic.
2. **Interpretation of Top Features:**
   - **Glucose:** Higher glucose levels have a significant impact on diabetes prediction in this model.
   - **BMI:** Elevated BMI increases the likelihood of being predicted as diabetic.
   - **Age:** Older age is associated with a higher prediction of diabetes.
   - **Insulin:** Higher insulin levels correlate with a higher likelihood of being predicted as diabetic.
   - **Blood Pressure:** Elevated blood pressure indicates a higher probability of being predicted as diabetic.
3. **Correlation with OutcomeStatus:**
   - Glucose has a strong positive correlation with OutcomeStatus, indicating its significant impact on diabetes prediction.

    - BMI also has a notable positive correlation, emphasizing its role in predicting diabetes.
4. **Feature Interaction:**
    - The interaction between glucose levels and BMI is significant for diabetes prediction: High glucose levels combined with a high BMI can significantly increase the risk of developing diabetes. The excess fat, especially visceral fat, in individuals with a higher BMI can lead to insulin resistance, impairing glucose utilization and elevating blood glucose levels.
    - Monitoring glucose levels in individuals with varying BMIs allows for a more comprehensive assessment of diabetes risk. Elevated glucose levels in conjunction with a higher BMI should alert healthcare professionals to closely monitor and evaluate the individual for potential diabetes or prediabetes.
    - Incorporating both glucose levels and BMI data into predictive models can enhance the accuracy of diabetes prediction. Machine learning algorithms and statistical models can utilize these variables, among others, to predict the likelihood of an individual developing diabetes.
5. **Domain Knowledge Integration:**
    - Integrating domain knowledge is crucial to understanding how these features align with known factors influencing diabetes.

Conclusion

The analysis of the diabetes dataset revealed valuable insights into the factors influencing diabetes occurrence. The predictive models developed provide a promising means of identifying individuals at risk of diabetes, facilitating timely intervention and improved healthcare strategies.

Recommendation

- Focus interventions or screenings on individuals with high glucose levels to detect and manage diabetes.
- Consider targeted strategies for individuals with high BMI, as it is a significant indicator of diabetes risk.