Dataset Name: Pima Indians Diabetes Dataset
Dataset Link: Pima Indians Diabetes Database | Kaggle

Brief about Diabetes

Diabetes mellitus is one of the major noncommunicable diseases which have a great impact on human life today. Many nations are now facing a swiftly rising growth of diabetes among their residents. According to a study by the World Health Organization (WHO), this number will have risen to 552 million by 2030, denoting that one in 10 grownups will have diabetes by 2030 if no serious act is taken. Amidst this pandemic situation, the intake of sugary and junk food has increased amongst youth which further led to a steep 9% increase in diabetes cases among young adults. Also COVID-19 patients are at a high risk of developing diabetes as this virus targets and impairs the body's insulin-producing cells. Total deaths from diabetes are projected to rise by more than 50 % in the next 10 years.

About Dataset

The data set used for the purpose of this analysis is Pima Indians Diabetes Database of National Institute of Diabetes and Digestive and Kidney Diseases. This diabetes database, donated by Vincent Sigillito, is a collection of medical diagnostic reports of 768 examples from a population living near Phoenix, Arizona, USA. You can find more information about the dataset at Pima Indians Diabetes Database | Kaggle.

The samples consist of examples with 8 attribute values and one of the two possible outcomes, namely whether the patient is tested positive for diabetes (indicated by output one) or not (indicated by zero).

Attributes of Dataset :

Data Description for the 9 variables are as follows.

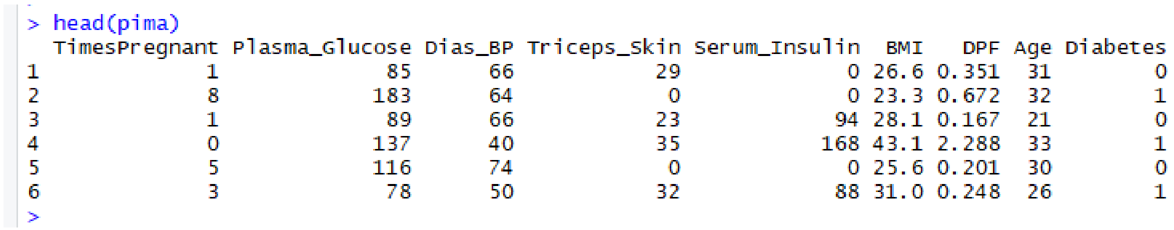
# 1. Number of times pregnant
# 2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
# 3. Diastolic blood pressure (mm Hg)
# 4. Triceps skin fold thickness (mm)
# 5. 2-Hour serum insulin (mu U/ml)
# 6. Body mass index (weight in kg/(height in m)^2)
# 7. Diabetes pedigree function
# 8. Age (years)
# 9. Class variable (0 or 1)

1. Loading Libraries

```
library(ggcorrplot)
library(ggplot2)
library(caret)
library(corrplot)
library(tidyverse)
library(e1071)
library(gridExtra)
library(graphics)
library(tree)
library(tune)
```

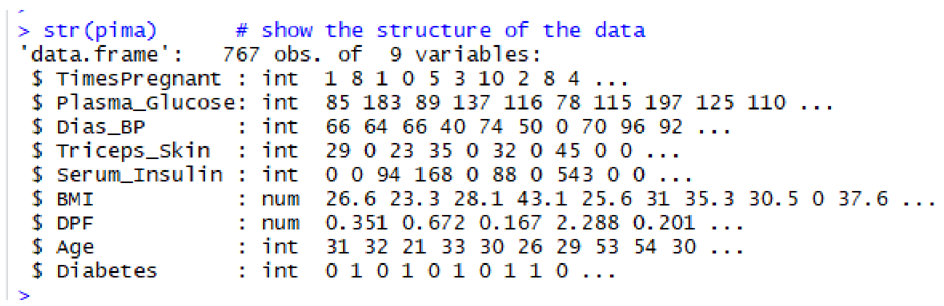
2. Loading the Dataset

```
 pima <- read.csv("C:/Users/dell/Downloads/archive/pima-indians-diabetes.csv ",
 col.names=c("Pregnant","Plasma_Glucose","Dias_BP","Triceps_Skin","Serum_Insulin",
"BMI","DPF","Age","Diabetes"))
 head(pima)
```



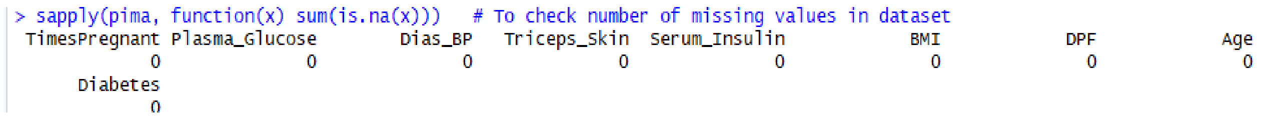
```
> head(pima)
  TimesPregnant Plasma_Glucose Dias_BP Triceps_Skin Serum_Insulin  BMI   DPF Age Diabetes
1             1             85      66           29             0 26.6 0.351  31        0
2             8            183      64            0             0 23.3 0.672  32        1
3             1             89      66           23            94 28.1 0.167  21        0
4             0            137      40           35           168 43.1 2.288  33        1
5             5            116      74            0             0 25.6 0.201  30        0
6             3             78      50           32            88 31.0 0.248  26        1
>
```

str(pima)



```
> str(pima)      # show the structure of the data
'data.frame':   767 obs. of  9 variables:
 $ TimesPregnant : int  1 8 1 0 5 3 10 2 8 4 ...
 $ Plasma_Glucose: int  85 183 89 137 116 78 115 197 125 110 ...
 $ Dias_BP       : int  66 64 66 40 74 50 0 70 96 92 ...
 $ Triceps_Skin  : int  29 0 23 35 0 32 0 45 0 0 ...
 $ Serum_Insulin : int  0 0 94 168 0 88 0 543 0 0 ...
 $ BMI           : num  26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 37.6 ...
 $ DPF           : num  0.351 0.672 0.167 2.288 0.201 ...
 $ Age           : int  31 32 21 33 30 26 29 53 54 30 ...
 $ Diabetes      : int  0 1 0 1 0 1 0 1 1 0 ...
>
```

sapply(pima, function(x) sum(is.na(x))



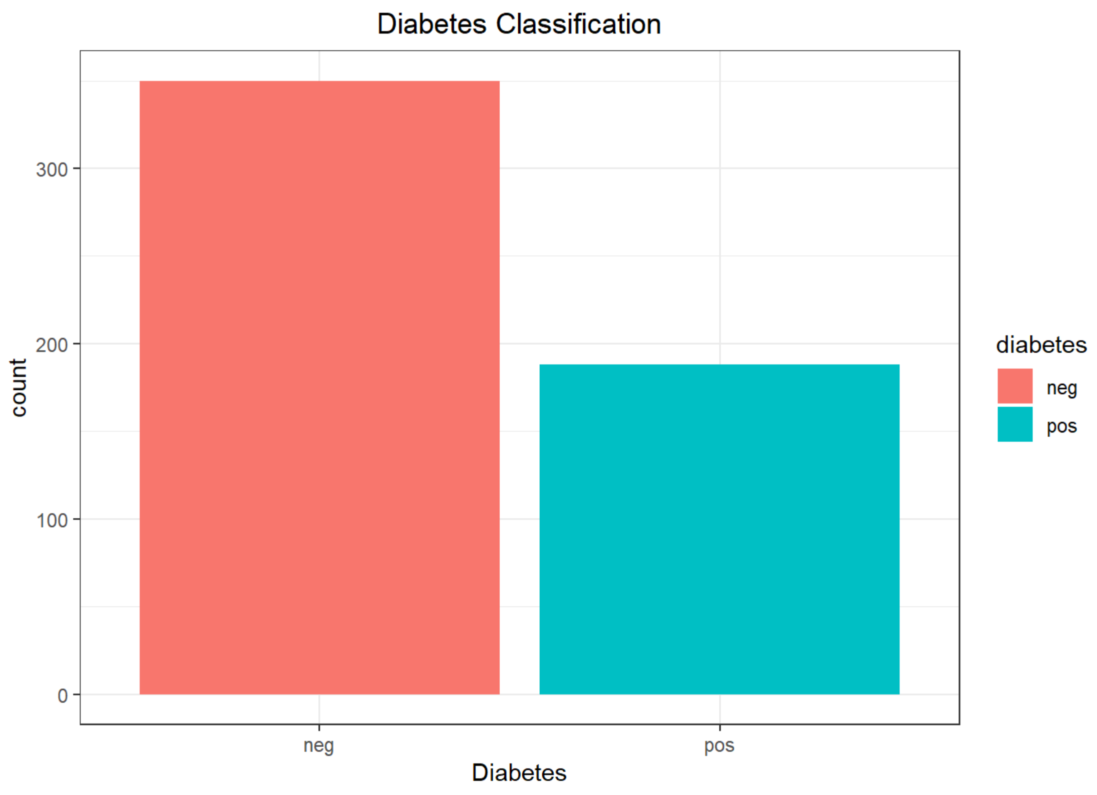
```
> sapply(pima, function(x) sum(is.na(x)))   # To check number of missing values in dataset
 TimesPregnant Plasma_Glucose        Dias_BP    Triceps_Skin  Serum_Insulin            BMI            DPF            Age
             0              0              0              0              0              0              0              0
      Diabetes
             0
```

There are no missing values, so we can start with data exploration.

3. Exploratory data analysis

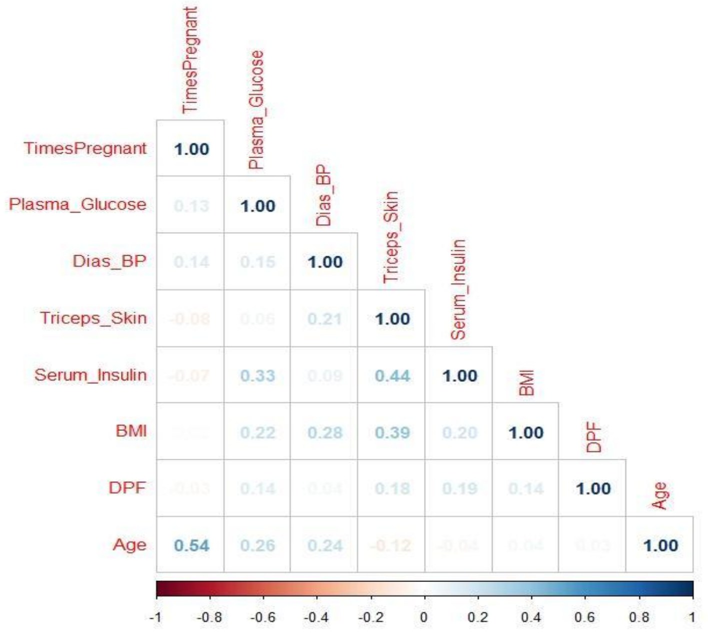
- Diabetes Classification

```
ggplot(pima, aes(pima$diabetes)) +
geom_bar(fill = c("pink","cyan")) +
theme_bw() +
labs(title = "Diabetes Classification", x = "Diabetes") +
theme(plot.title = element_text(hjust = 0.5))
```

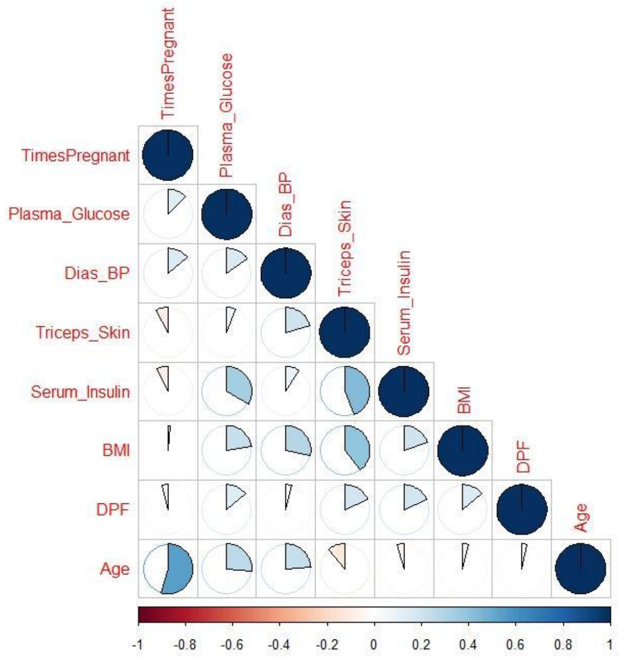


- Correlation plot

```
corrplot(cor(pima[, -9]), type = "lower", method = "number")
```



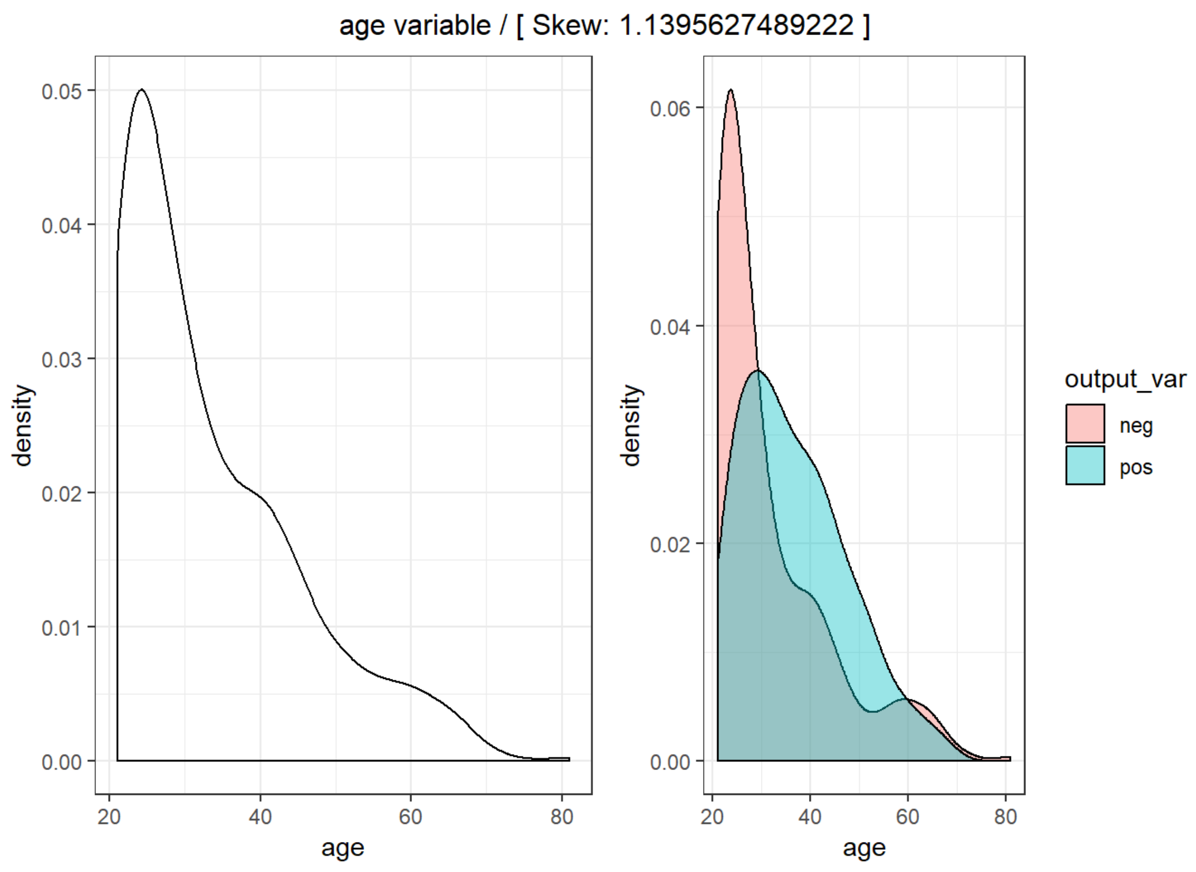
```
corrplot(cor(pima[, -9]), type = "lower", method = "pie")
```



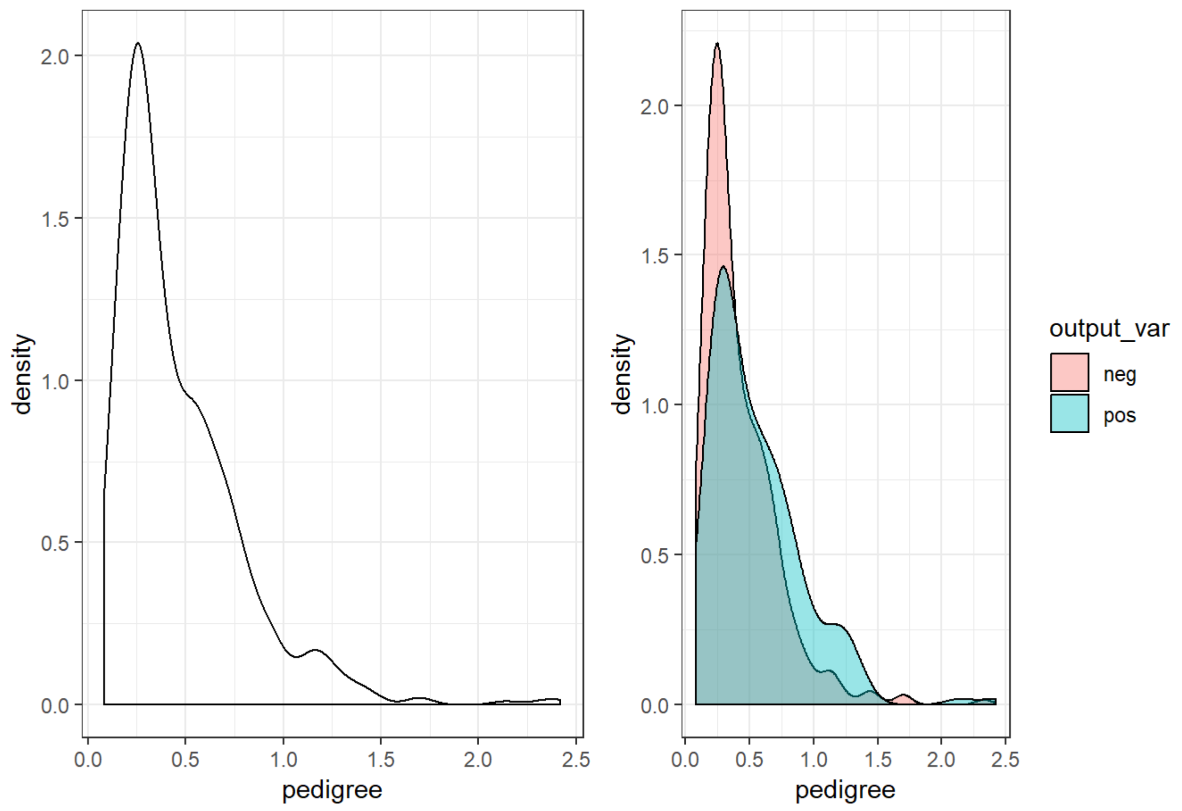
As we can see there is a moderately positive high correlation between age and pregnancy count.

- Univariate analysis:-

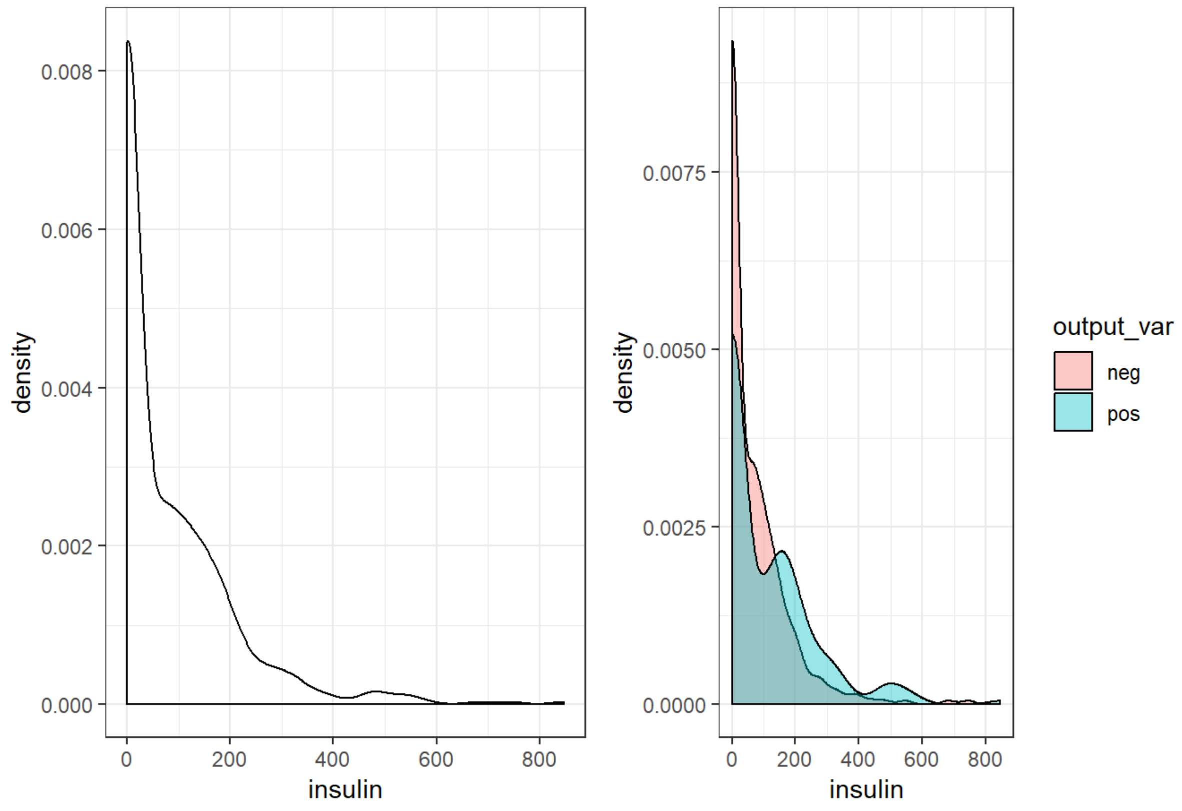
```
univar_graph <- function(univar_name, univar, data, output_var) {
g_1 <- ggplot(data, aes(x=univar)) +
geom_density() + #for color and font
xlab(univar_name) + #title name
theme_bw()
g_2 <- ggplot(data, aes(x=univar, fill=output_var)) +
geom_density(alpha=0.4) +
xlab(univar_name) +
theme_bw()
gridExtra::grid.arrange(g_1, g_2, ncol=2, top = paste(univar_name,"variable", "/ [
Skew:",timeDate::skewness(univar),"]")) #for labelling of the graph
}
for (x in 1: (ncol(pima)-1) {
univar_graph(univar_name = names(pima)[x], univar = pima[,x], data = pima,
output_var = pima[,'diabetes'])
}
```

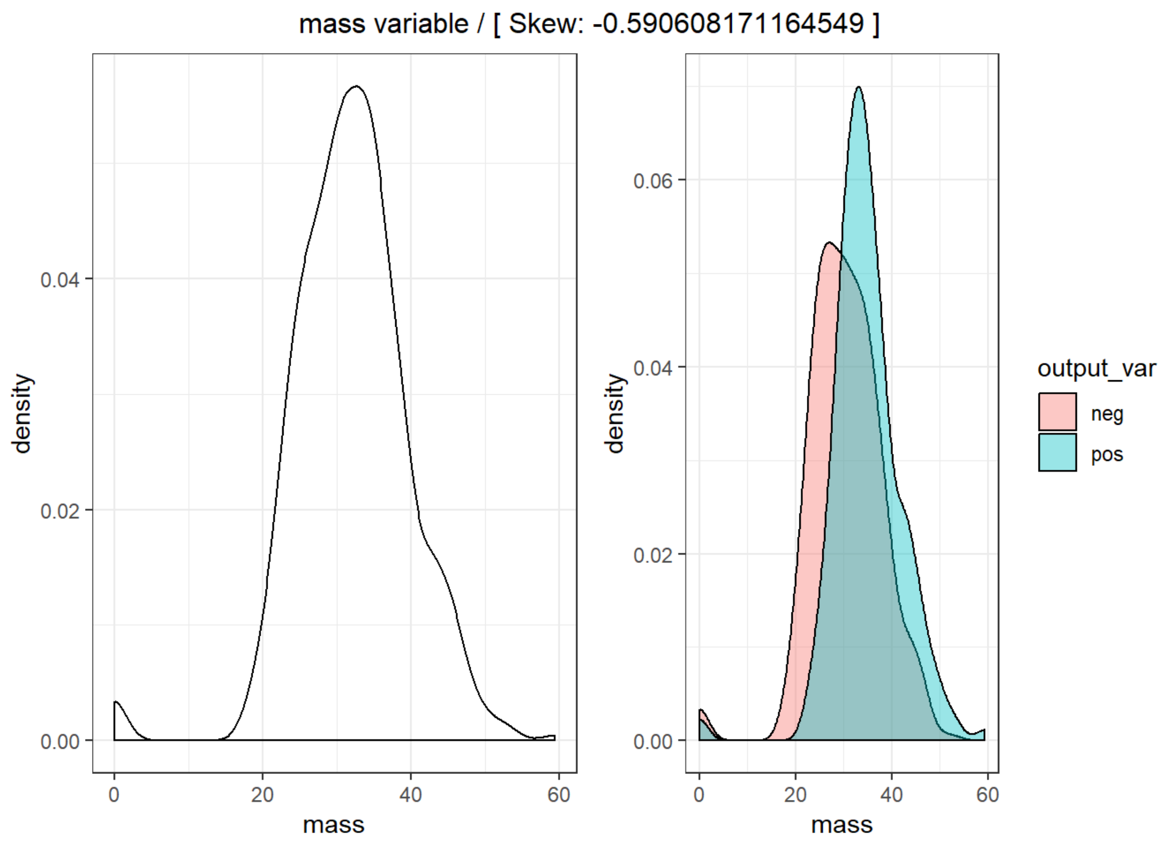


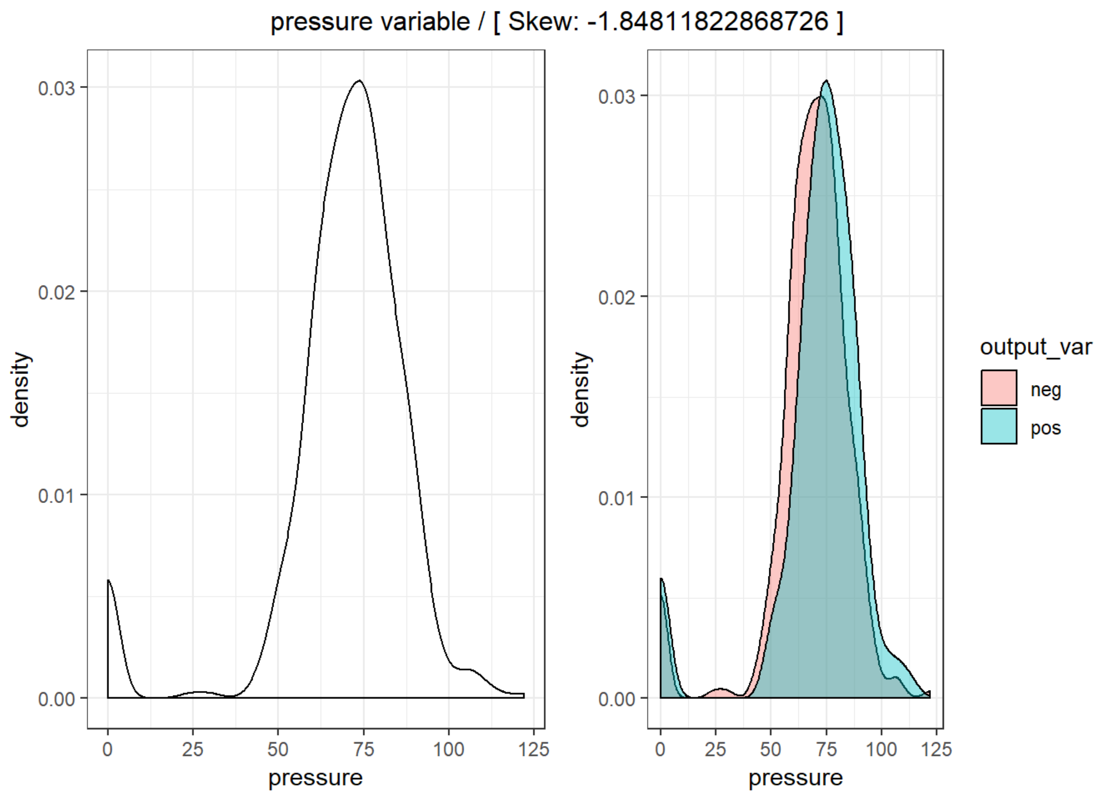
pedigree variable / [Skew: 1.91743170102542]



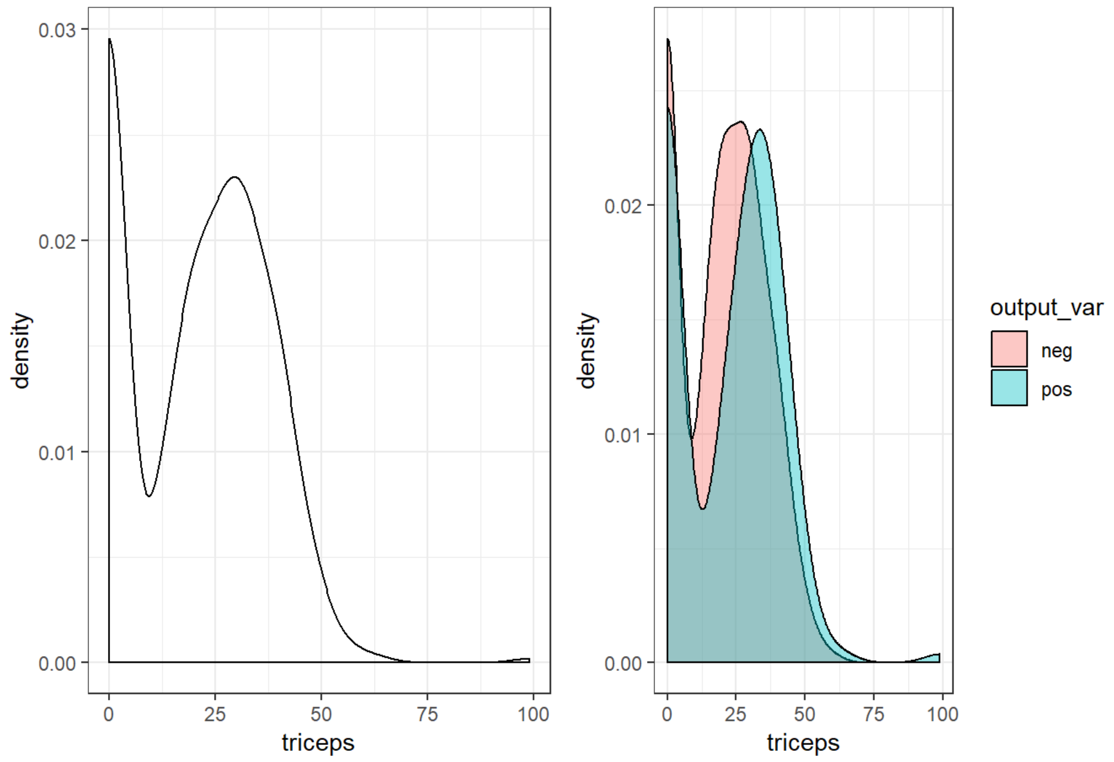
insulin variable / [Skew: 2.34310944456496]



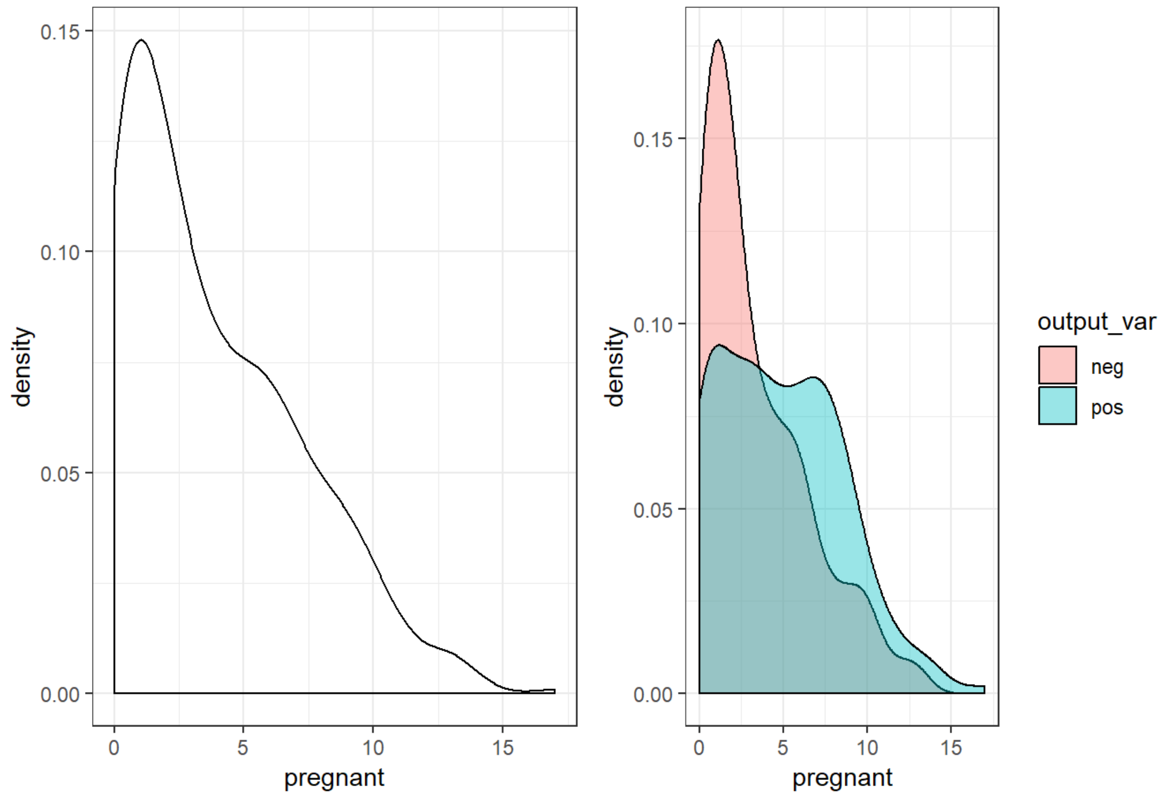

mass variable / [Skew: -0.590608171164549]
density
mass
0.00
0.02
0.04
0.06
0
20
40
60
output_var
neg
pos

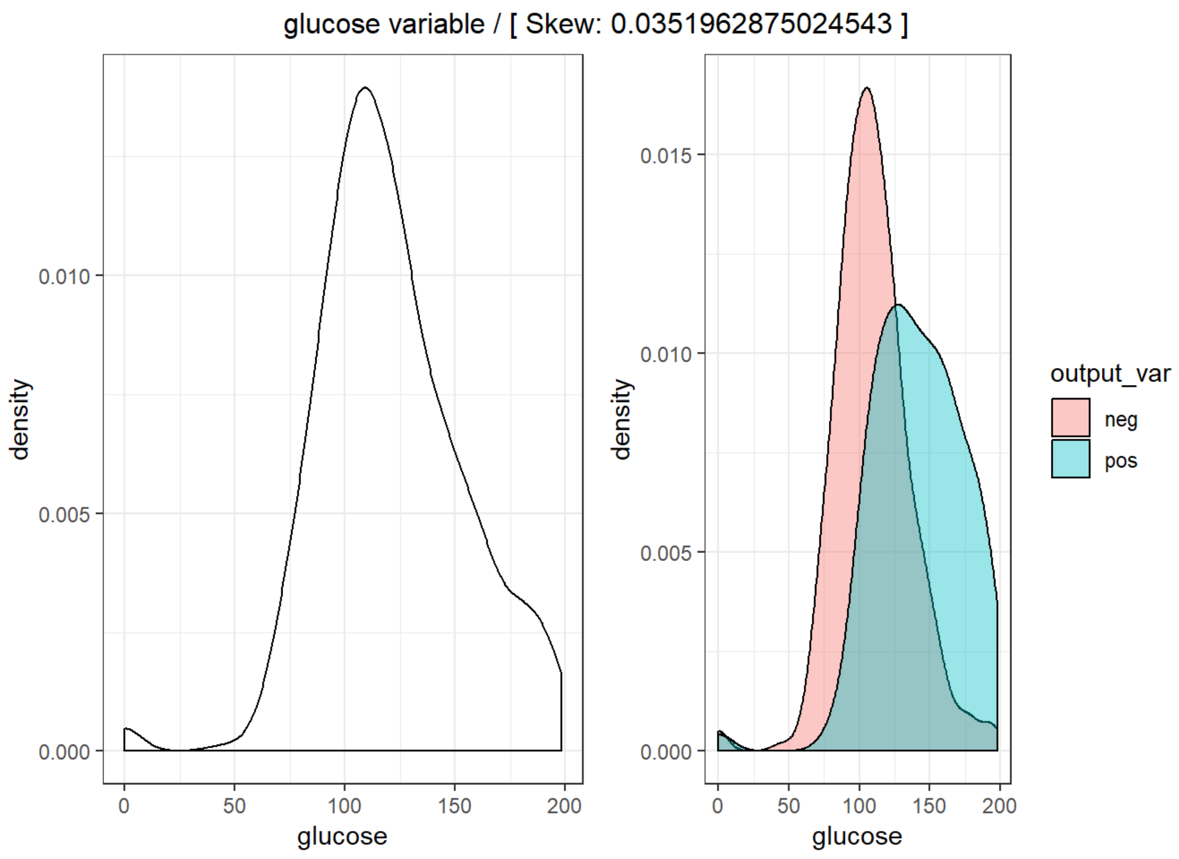

pressure variable / [Skew: -1.84811822868726]
density
pressure
0.00
0.01
0.02
0.03
0
25
50
75
100
125
output_var
neg
pos

triceps variable / [Skew: 0.211746768027423]



pregnant variable / [Skew: 0.849116909607062]



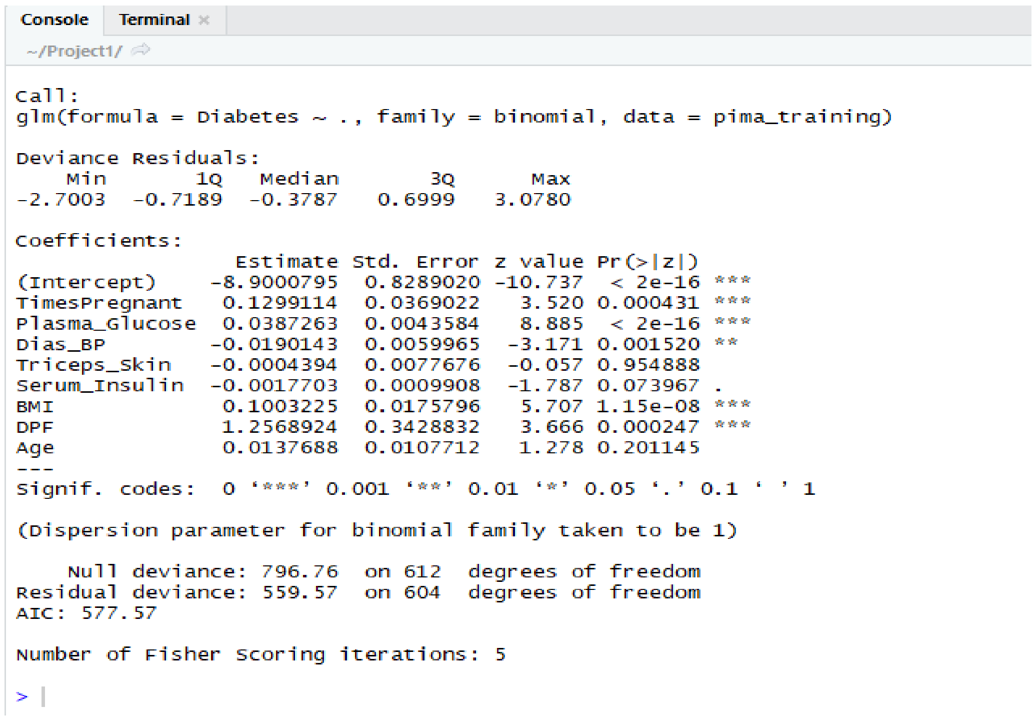


Variables such as Insulin, Pedigree and Age have high right skewness.
Pressure and Mass have negative skewness.
Pregnant, Glucose, and Triceps have moderate to low right skewness.

4. ML Model Building

4.1 Logistic Regression model

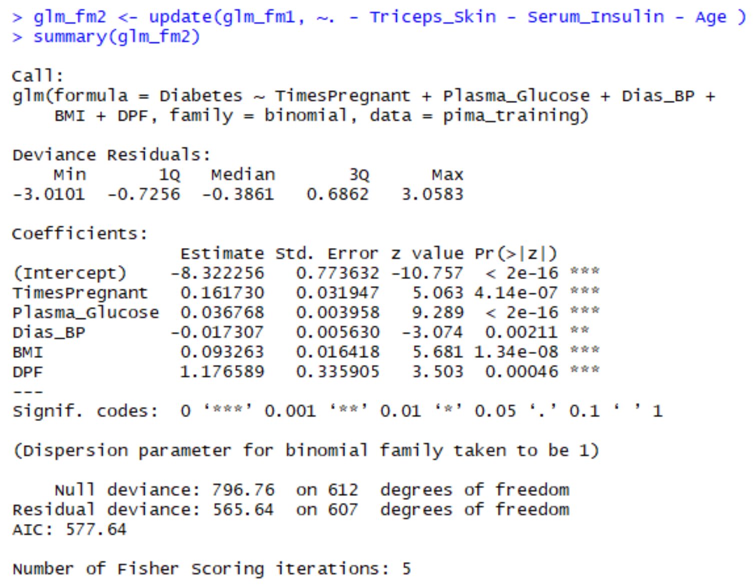
```
set.seed(123)
n <- nrow(pima)
train <- sample(n, trunc(0.80*n))
pima_training <- pima[train, ]
pima_testing <- pima[-train, ]

# Training The logistic Regression Model

glm_fm1 <- glm(Diabetes ~., data = pima_training, family = binomial)
summary(glm_fm1)
```



```
Call:
glm(formula = Diabetes ~ ., family = binomial, data = pima_training)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7003  -0.7189  -0.3787   0.6999   3.0780

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)    -8.9000795  0.8289020 -10.737  < 2e-16 ***
TimesPregnant   0.1299114  0.0369022   3.520 0.000431 ***
Plasma_Glucose  0.0387263  0.0043584   8.885  < 2e-16 ***
Dias_BP        -0.0190143  0.0059965  -3.171 0.001520 **
Triceps_Skin   -0.0004394  0.0077676  -0.057 0.954888
Serum_Insulin  -0.0017703  0.0009908  -1.787 0.073967 .
BMI             0.1003225  0.0175796   5.707 1.15e-08 ***
DPF             1.2568924  0.3428832   3.666 0.000247 ***
Age             0.0137688  0.0107712   1.278 0.201145
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 796.76  on 612  degrees of freedom
Residual deviance: 559.57  on 604  degrees of freedom
AIC: 577.57

Number of Fisher Scoring iterations: 5

> |
```

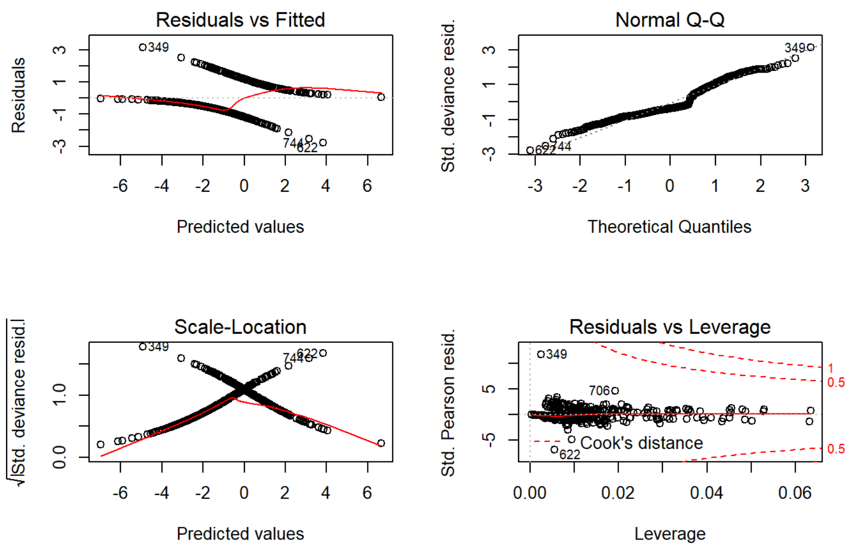
The result shows that the variables Triceps_Skin, Serum_Insulin and Age are not statistically significant. In other words, the p_values is greater than 0.01. Therefore they will be removed.

glm_fm2 <- update(glm_fm1, ~. - Triceps_Skin - Serum_Insulin - Age)
summary(glm_fm2)



```
> glm_fm2 <- update(glm_fm1, ~. - Triceps_Skin - Serum_Insulin - Age )
> summary(glm_fm2)

Call:
glm(formula = Diabetes ~ TimesPregnant + Plasma_Glucose + Dias_BP +
    BMI + DPF, family = binomial, data = pima_training)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0101  -0.7256  -0.3861   0.6862   3.0583

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    -8.322256   0.773632 -10.757  < 2e-16 ***
TimesPregnant   0.161730   0.031947   5.063 4.14e-07 ***
Plasma_Glucose  0.036768   0.003958   9.289  < 2e-16 ***
Dias_BP        -0.017307   0.005630  -3.074  0.00211 **
BMI             0.093263   0.016418   5.681 1.34e-08 ***
DPF             1.176589   0.335905   3.503  0.00046 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 796.76  on 612  degrees of freedom
Residual deviance: 565.64  on 607  degrees of freedom
AIC: 577.64

Number of Fisher Scoring iterations: 5
```

```
par(mfrow = c(2,2))      # Graphically shows the statistical difference
plot(glm_fm2)
```



Residuals vs fitted values; . The dotted line at y=0 indicates our fit line; . Any point on fit line obviously has zero residual. Points above have positive residuals and points below have negative residuals. The red line is the smoothed high order polynomial curve to give us an idea of the pattern of residual movement. In our case we can see that our residuals have a logarithmic pattern that means we got a better model.

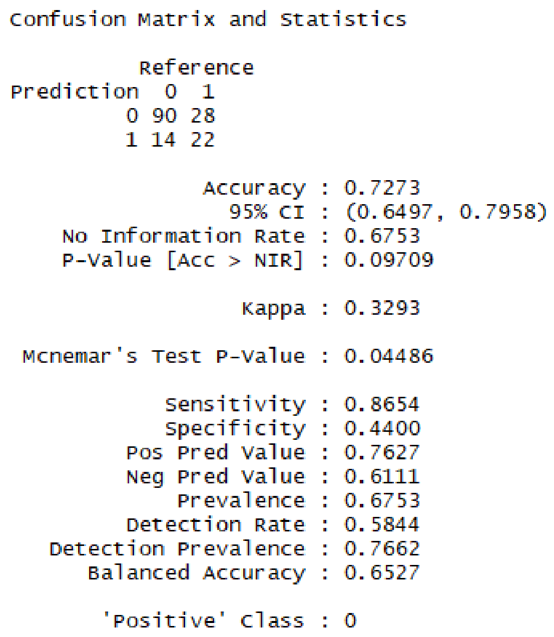
Normal Q-Q Plot: The Normal Q-Q plot is used to check if our residuals follow Normal distribution or not; The residuals are normally distributed if the points follow the dotted line closely; In this case residual points follow the dotted line closely.

Scale - Location Plot: . Scale location plot indicates spread of points across predicted values range; One of the assumptions for Regression is Homoscedasticity i.e variance should be reasonably equal across the predictor range; A horizontal red line is ideal and would indicate that residuals have uniform variance across the range.

Now let's analyse our leverage plot draw inferences. In this plot the dotted red lines are Cook's distance and the areas of interest for us are the ones outside the dotted line on the top right corner or bottom right corner. If any point falls in that region, we say that the observation has high leverage or potential for influencing our model is higher if we exclude that point. It's not always the case though that all outliers will have high leverage or vice versa.

In this case we do not have any points considered outlier, therefore the Logistic Regression model fits perfectly.

```
# Testing the logistic Regression  Model
glm_probs <- predict(glm_fm2, newdata = pima_testing, type = "response")
glm_pred <- ifelse(glm_probs > 0.5, 1, 0)
confusionMatrix(as.factor(glm_pred), as.factor(pima_testing$Diabetes ))
 # Confusion Matrix for logistic regression
acc_glm_fit <- confusionMatrix(as.factor(glm_pred), as.factor(pima_testing$Diabetes
))$overall['Accuracy']
```

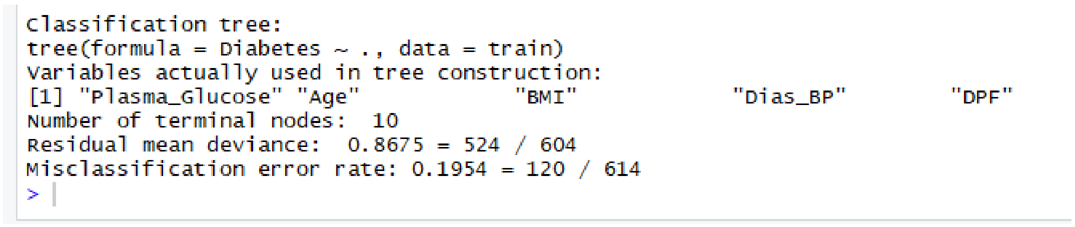


```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 90 28
         1 14 22

               Accuracy : 0.7273
                 95% CI : (0.6497, 0.7958)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : 0.09709

                  Kappa : 0.3293

 Mcnemar's Test P-Value : 0.04486

            Sensitivity : 0.8654
            Specificity : 0.4400
         Pos Pred Value : 0.7627
         Neg Pred Value : 0.6111
             Prevalence : 0.6753
         Detection Rate : 0.5844
   Detection Prevalence : 0.7662
      Balanced Accuracy : 0.6527

       'Positive' Class : 0
```

4.2 Decision Tree Model

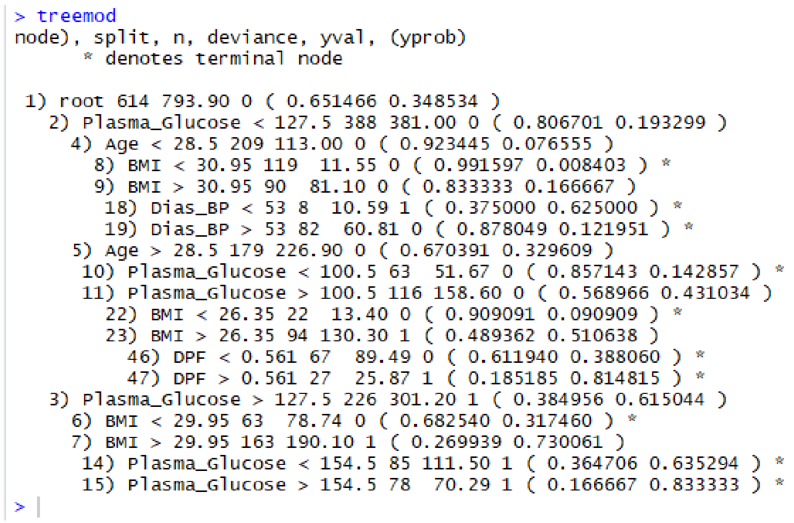
```
pima <- read.csv("C:/Users/dell/Downloads/archive/pima-indians-diabetes.csv ",
col.names=c("Pregnant","Plasma_Glucose","Dias_BP","Triceps_Skin","Serum_Insulin","
BMI","DPF","Age","Diabetes"))
pima$Diabetes <- as.factor(pima$Diabetes)

set.seed(1000)
intrain <- createDataPartition(y = pima$Diabetes, p = 0.7, list = FALSE)
train <- pima[intrain, ]
test <- pima[-intrain, ]
```

```
# Training The Decision tree Model
treemod <- tree(Diabetes ~ ., data = train)
summary(treemod)
```



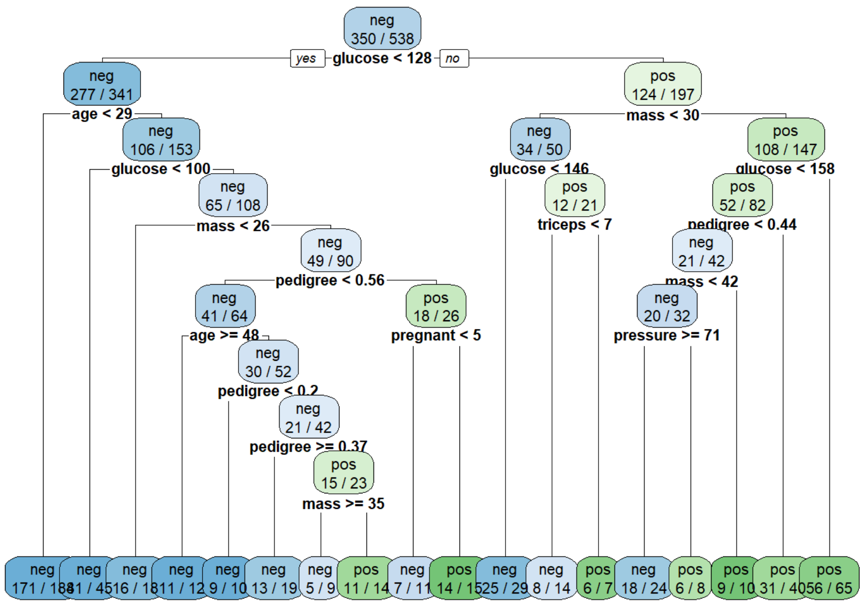
```
Classification tree:
tree(formula = Diabetes ~ ., data = train)
Variables actually used in tree construction:
[1] "Plasma_Glucose" "Age"            "BMI"            "Dias_BP"        "DPF"
Number of terminal nodes:  10
Residual mean deviance:  0.8675 = 524 / 604
Misclassification error rate: 0.1954 = 120 / 614
>
```

```
treemod
```

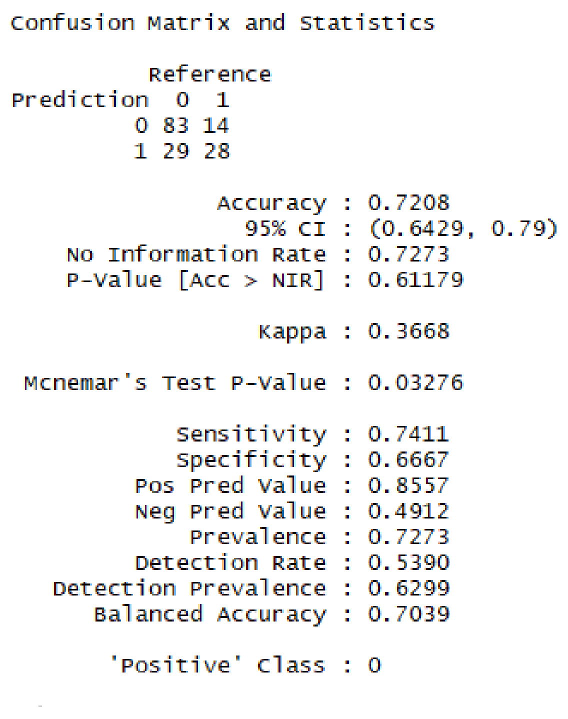


```
> treemod
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 614 793.90 0 ( 0.651466 0.348534 )
   2) Plasma_Glucose < 127.5 388 381.00 0 ( 0.806701 0.193299 )
     4) Age < 28.5 209 113.00 0 ( 0.923445 0.076555 )
       8) BMI < 30.95 119  11.55 0 ( 0.991597 0.008403 ) *
       9) BMI > 30.95 90  81.10 0 ( 0.833333 0.166667 )
        18) Dias_BP < 53 8  10.59 1 ( 0.375000 0.625000 ) *
        19) Dias_BP > 53 82  60.81 0 ( 0.878049 0.121951 ) *
     5) Age > 28.5 179 226.90 0 ( 0.670391 0.329609 )
      10) Plasma_Glucose < 100.5 63  51.67 0 ( 0.857143 0.142857 ) *
      11) Plasma_Glucose > 100.5 116 158.60 0 ( 0.568966 0.431034 )
        22) BMI < 26.35 22  13.40 0 ( 0.909091 0.090909 ) *
        23) BMI > 26.35 94 130.30 1 ( 0.489362 0.510638 )
          46) DPF < 0.561 67  89.49 0 ( 0.611940 0.388060 ) *
          47) DPF > 0.561 27  25.87 1 ( 0.185185 0.814815 ) *
   3) Plasma_Glucose > 127.5 226 301.20 1 ( 0.384956 0.615044 )
     6) BMI < 29.95 63  78.74 0 ( 0.682540 0.317460 ) *
     7) BMI > 29.95 163 190.10 1 ( 0.269939 0.730061 )
      14) Plasma_Glucose < 154.5 85 111.50 1 ( 0.364706 0.635294 ) *
      15) Plasma_Glucose > 154.5 78  70.29 1 ( 0.166667 0.833333 ) *
>
```

```
plot(treemod)
text(treemod, pretty = 0)
```



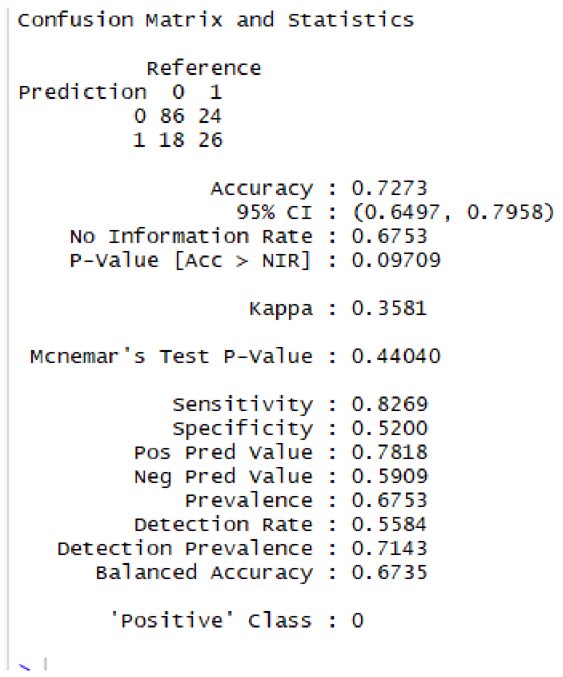
```
# Testing the Decision tree Model
tree_pred <- predict(treemod, newdata = test, type = "class" )
confusionMatrix(tree_pred, test$Diabetes)
acc_treemod <- confusionMatrix(tree_pred, test$Diabetes)$overall['Accuracy']
```



```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 83 14
         1 29 28

               Accuracy : 0.7208
                 95% CI : (0.6429, 0.79)
    No Information Rate : 0.7273
    P-Value [Acc > NIR] : 0.61179

                  Kappa : 0.3668

 Mcnemar's Test P-Value : 0.03276

            Sensitivity : 0.7411
            Specificity : 0.6667
         Pos Pred Value : 0.8557
         Neg Pred Value : 0.4912
             Prevalence : 0.7273
         Detection Rate : 0.5390
   Detection Prevalence : 0.6299
      Balanced Accuracy : 0.7039

       'Positive' Class : 0
```

4.3 Random Forest Model

```
# Training the random forest
set.seed(123)
rf_pima <- randomForest(Diabetes ~., data = pima_training, mtry = 8, ntree=50,
importance = TRUE)
# Testing the Model
rf_probs <- predict(rf_pima, newdata = pima_testing)
rf_pred <- ifelse(rf_probs > 0.5, 1, 0)
confusionMatrix(as.factor(rf_pred),as.factor(pima_testing$Diabetes))
acc_rf_pima <- confusionMatrix(as.factor(rf_pred),
```



```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 86 24
         1 18 26

               Accuracy : 0.7273
                 95% CI : (0.6497, 0.7958)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : 0.09709

                  Kappa : 0.3581

 Mcnemar's Test P-Value : 0.44040

            Sensitivity : 0.8269
            Specificity : 0.5200
         Pos Pred Value : 0.7818
         Neg Pred Value : 0.5909
             Prevalence : 0.6753
         Detection Rate : 0.5584
   Detection Prevalence : 0.7143
      Balanced Accuracy : 0.6735

       'Positive' Class : 0
```

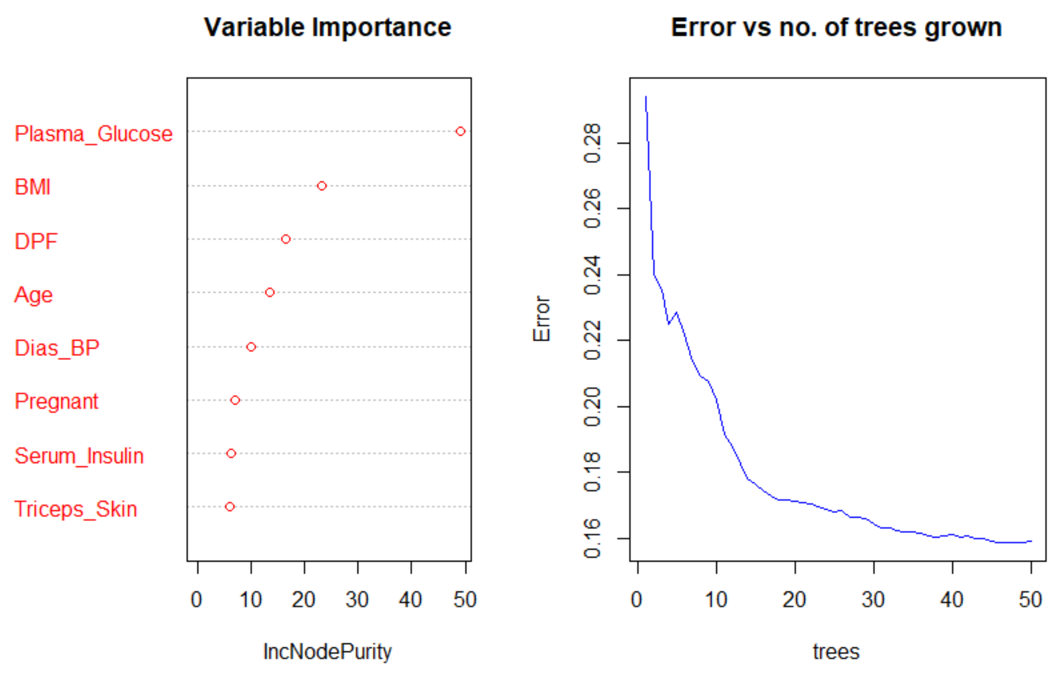
importance(rf_pima)

```
##                    %IncMSE IncNodePurity
## Pregnant        4.29534581      6.689093
## Plasma_Glucose 18.06247751     40.309147
## Dias_BP         3.73430996      9.720362
## Triceps_Skin   -0.09933701      3.813198
## Serum_Insulin  -1.80517988      4.980444
## BMI             7.40408204     20.281781
## DPF             2.90470464     14.348029
## Age             5.71252884     10.788115
```

The “Plasma_Glucose” is by far the most important variable.

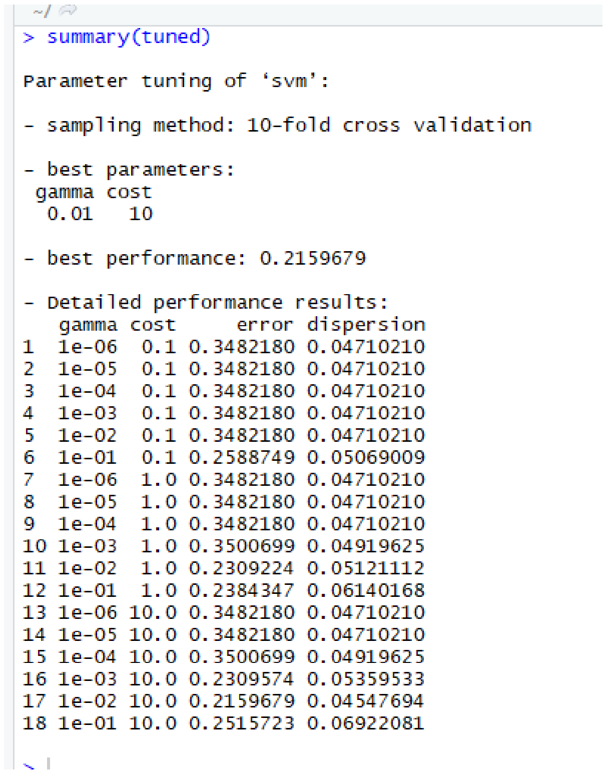
Let us plot a graph for displaying the most important variable.

```
par(mfrow = c(1, 2))
varImpPlot(rf_pima, type = 2, main = "Variable Importance",col = 'black')
plot(rf_pima, main = "Error vs no. of trees grown")
```



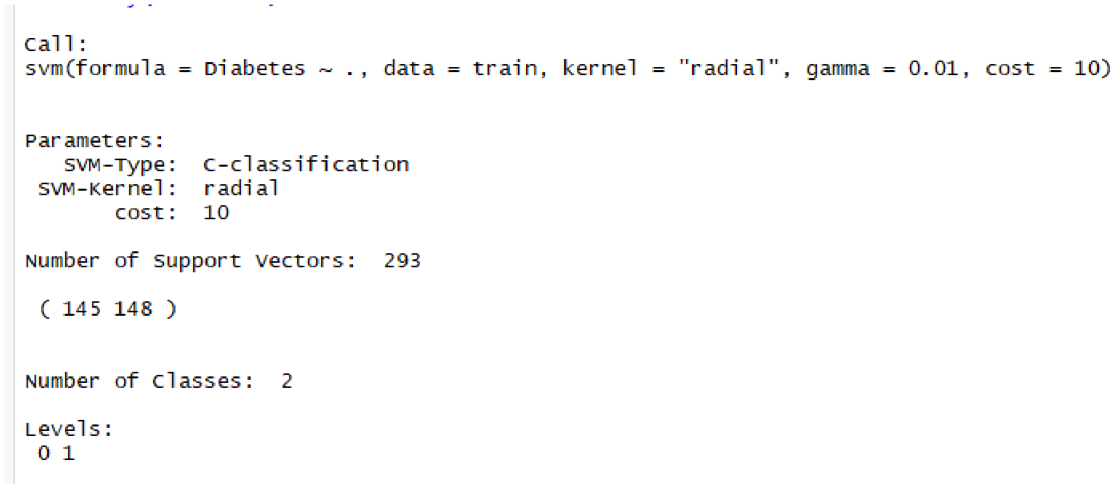
4.4 SVM Model

```
pima <- read.csv("C:/Users/dell/Downloads/archive/pima-indians-diabetes.csv ",
col.names=c("Pregnant","Plasma_Glucose","Dias_BP","Triceps_Skin","Serum_Insulin","
BMI","DPF","Age","Diabetes"))
pima$Diabetes <- as.factor(pima$Diabetes)

set.seed(1000)
intrain <- createDataPartition(y = pima$Diabetes, p = 0.7, list = FALSE)
train <- pima[intrain, ]
test <- pima[-intrain, ]

#Training the SVM model
tuned <- tune.svm(Diabetes ~., data = train, gamma = 10^(-6:-1), cost = 10^(-1:1))
summary(tuned)
```



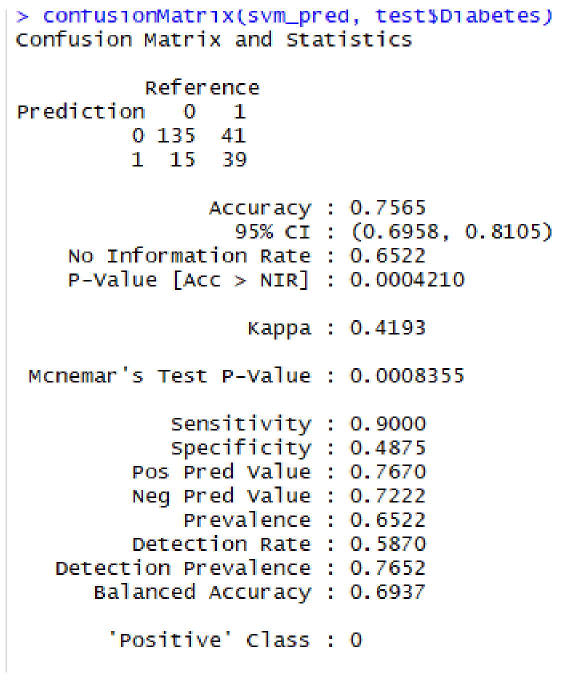
```
> summary(tuned)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
  0.01   10

- best performance: 0.2159679

- Detailed performance results:
   gamma cost     error dispersion
1  1e-06  0.1 0.3482180 0.04710210
2  1e-05  0.1 0.3482180 0.04710210
3  1e-04  0.1 0.3482180 0.04710210
4  1e-03  0.1 0.3482180 0.04710210
5  1e-02  0.1 0.3482180 0.04710210
6  1e-01  0.1 0.2588749 0.05069009
7  1e-06  1.0 0.3482180 0.04710210
8  1e-05  1.0 0.3482180 0.04710210
9  1e-04  1.0 0.3482180 0.04710210
10 1e-03  1.0 0.3500699 0.04919625
11 1e-02  1.0 0.2309224 0.05121112
12 1e-01  1.0 0.2384347 0.06140168
13 1e-06 10.0 0.3482180 0.04710210
14 1e-05 10.0 0.3482180 0.04710210
15 1e-04 10.0 0.3500699 0.04919625
16 1e-03 10.0 0.2309574 0.05359533
17 1e-02 10.0 0.2159679 0.04547694
18 1e-01 10.0 0.2515723 0.06922081
```

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



```
Call:
svm(formula = Diabetes ~ ., data = train, kernel = "radial", gamma = 0.01, cost = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10

Number of Support Vectors:  293

 ( 145 148 )


Number of Classes:  2

Levels:
 0 1
```

```
#Testing the SVM  model
svm_pred <- predict(svm_model, newdata = test)
confusionMatrix(svm_pred, test$Diabetes)
acc_svm_model <- confusionMatrix(svm_pred, test$Diabetes)$overall['Accuracy']
```

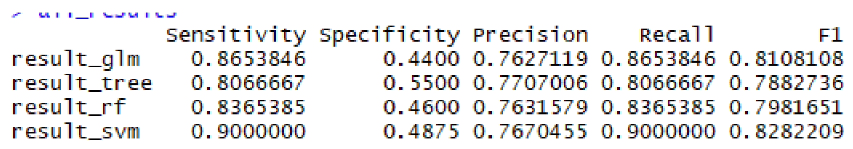


```
> confusionMatrix(svm_pred, test$Diabetes)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 135  41
         1  15  39

               Accuracy : 0.7565
                 95% CI : (0.6958, 0.8105)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 0.0004210

                  Kappa : 0.4193

 Mcnemar's Test P-Value : 0.0008355

            Sensitivity : 0.9000
            Specificity : 0.4875
         Pos Pred Value : 0.7670
         Neg Pred Value : 0.7222
             Prevalence : 0.6522
         Detection Rate : 0.5870
   Detection Prevalence : 0.7652
      Balanced Accuracy : 0.6937

       'Positive' Class : 0
```

5. Evaluation of Models

#Evaluating the performance of all models

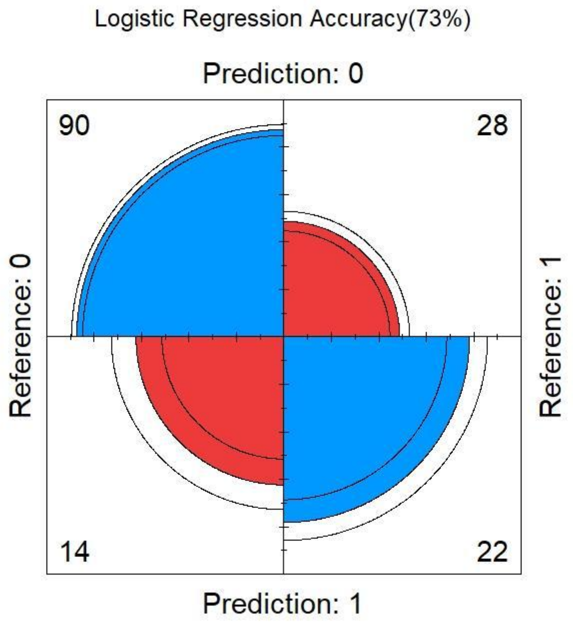
```
result_glm <- c(acc_glm_fit$byClass['Sensitivity'],acc_glm_fit$byClass['Specificity'],
                acc_glm_fit$byClass['Precision'],  acc_glm_fit$byClass['Recall'],
                acc_glm_fit$byClass['F1'])
result_tree <- c(acc_treemod$byClass['Sensitivity'],acc_treemod$byClass['Specificity'],
                acc_treemod$byClass['Precision'], acc_treemod$byClass['Recall'],
                acc_treemod$byClass['F1'])
 result_rf <-   c(acc_rf_pima$byClass['Sensitivity'],acc_rf_pima$byClass['Specificity'],
               acc_rf_pima$byClass['Precision'], acc_rf_pima$byClass['Recall'],
               acc_rf_pima$byClass['F1'])
result_svm <-c(acc_svm_model$byClass['Sensitivity'],
                acc_svm_model$byClass['Specificity'], acc_svm_model$byClass['Precision'],
                acc_svm_model$byClass['Recall'], acc_svm_model$byClass['F1'])
   all_results <- data.frame(rbind(result_glm, result_tree, result_rf, result_svm))
   names(all_results) <- c("Sensitivity", "Specificity", "Precision", "Recall", "F1")
   all_results
```



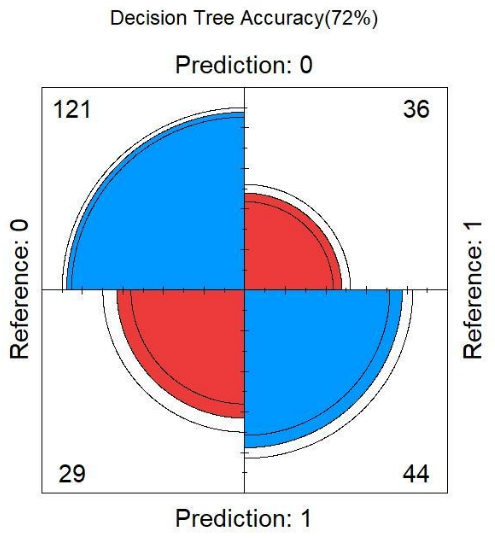
```
            Sensitivity Specificity Precision    Recall        F1
result_glm    0.8653846      0.4400 0.7627119 0.8653846 0.8108108
result_tree   0.8066667      0.5500 0.7707006 0.8066667 0.7882736
result_rf     0.8365385      0.4600 0.7631579 0.8365385 0.7981651
result_svm    0.9000000      0.4875 0.7670455 0.9000000 0.8282209
```

Accuracy of all the models

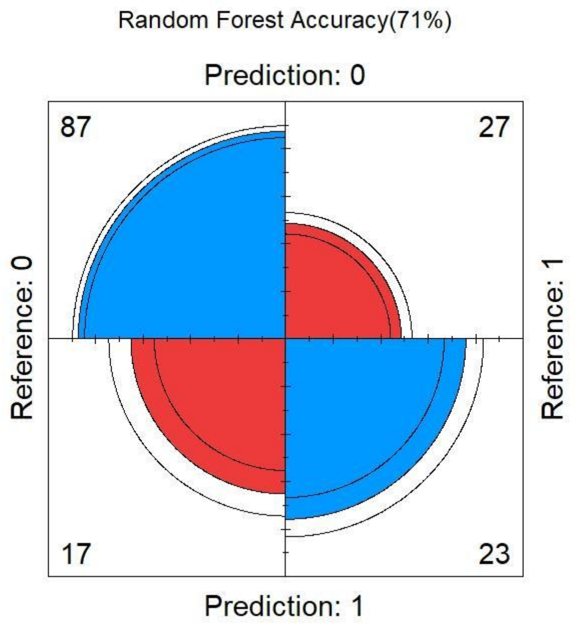
```
col <- c("#ed3b3b", "#0099ff")

graphics::fourfoldplot(acc_glm_fit$table, color = col, conf.level = 0.95, margin = 1,
        main = paste("Logistic Regression Accuracy (",round(acc_glm_fit$overall[1]*
        100),"%)", sep = ""))
```



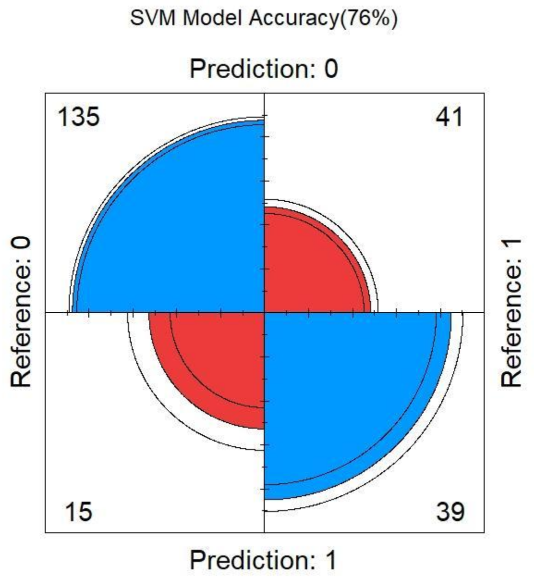
```
graphics::fourfoldplot(acc_treemod$table, color = col, conf.level = 0.95, margin = 1,
     main = paste("Decision Tree Accuracy(",round(acc_treemod$overall[1]*100),"%)",
     sep = ""))
```



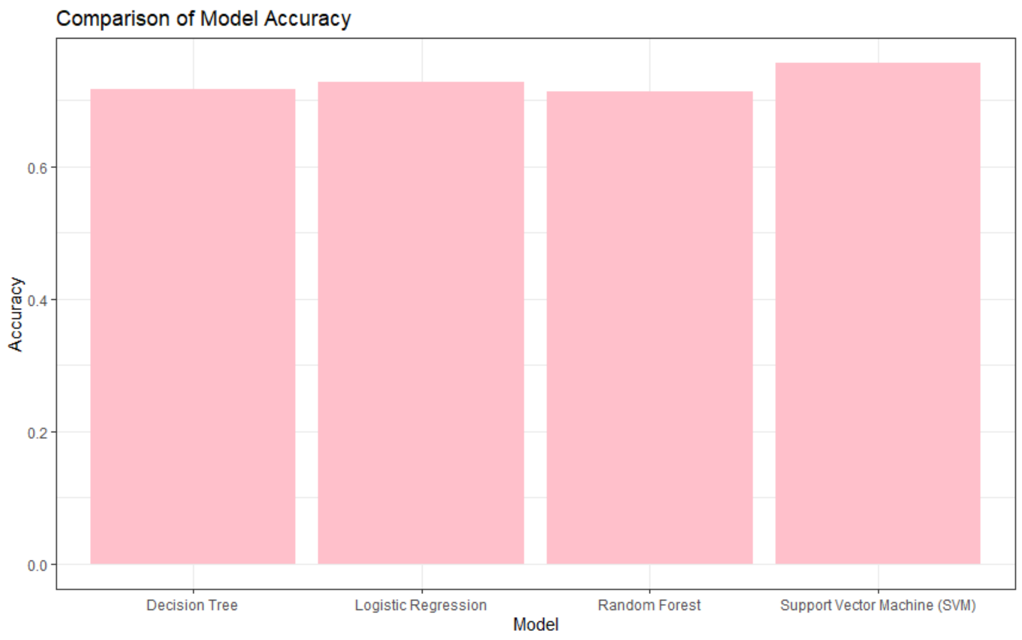
```
graphics::fourfoldplot(acc_rf_pima$table, color = col, conf.level = 0.95, margin = 1,
    main = paste("Random Forest Accuracy(",round(acc_rf_pima$overall[1]*100),"%)",
    sep = ""))
```



```
graphics::fourfoldplot(acc_svm_model$table, color = col, conf.level = 0.95, margin = 1,
    main = paste("SVM Model Accuracy(",round(acc_svm_model$overall[1]*100),"%)",
    sep = ""))
```



```
accuracy <- data.frame(Model=c("Logistic Regression","Decision Tree","Random
             Forest", "Support Vector Machine (SVM)"),
            Accuracy=c(acc_glm_fit$overall['Accuracy'], acc_treemod$overall['Accuracy'],
            acc_rf_pima$overall['Accuracy'], acc_svm_model$overall['Accuracy'] ))
            ggplot(accuracy,aes(x=Model,y=Accuracy)) + geom_bar(stat='identity') +
            theme_bw() + ggtitle('Comparison of Model Accuracy')
```



6. Conclusion:

To conclude, the graph shows that the SVM model has the best performance based on highest accuracy and F1 score achieved, followed by Logistic Regression, Decision Tree, Random Forest.