Diabetes Prediction Using Machine Learning Algorithms.

#Load the dataset

```
diabetes <- read.csv ("C:\\Users\\Christianah.O_BROOKS\\Downloads\\diabetes.csv", header=T, stringsAsFactors = F)
```

```
head(diabetes)
```

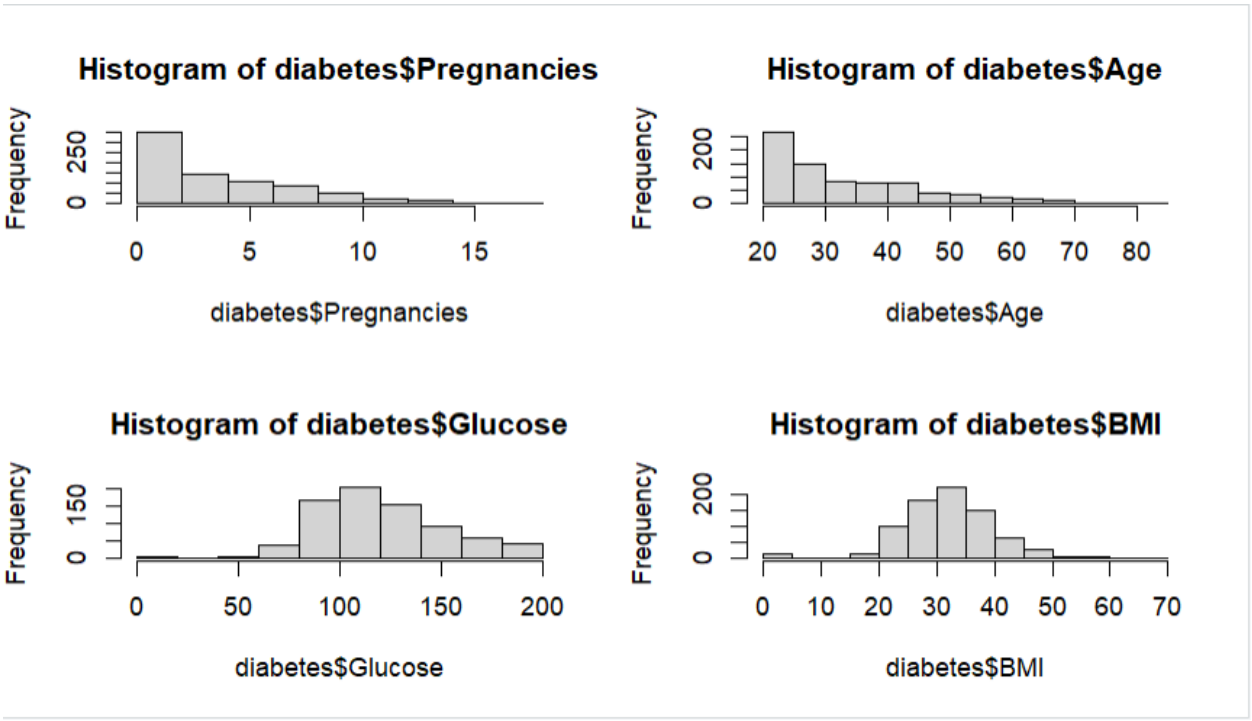
```
> head(diabetes)
  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI DiabetesPedigreeFunction
1           6     148            72            35       0 33.6                    0.627
2           1      85            66            29       0 26.6                    0.351
3           8     183            64             0       0 23.3                    0.672
4           1      89            66            23      94 28.1                    0.167
5           0     137            40            35     168 43.1                    2.288
6           5     116            74             0       0 25.6                    0.201
  Age Outcome
1  50       1
2  31       0
3  32       1
4  21       0
5  33       1
6  30       0
```

```
summary(diabetes)
```

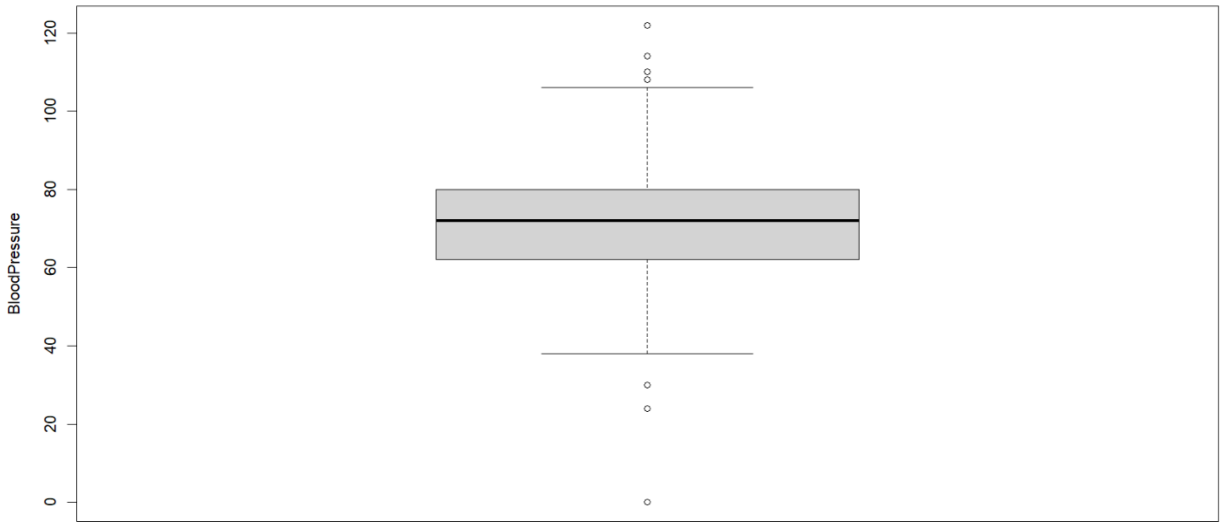
```
> summary(diabetes)
  Pregnancies        Glucose      BloodPressure    SkinThickness      Insulin
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00   1st Qu.:  0.0
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.2
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.0
      BMI        DiabetesPedigreeFunction      Age            Outcome
 Min.   : 0.00   Min.   :0.0780           Min.   :21.00   Min.   :0.000
 1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00   1st Qu.:0.000
 Median :32.00   Median :0.3725           Median :29.00   Median :0.000
 Mean   :31.99   Mean   :0.4719           Mean   :33.24   Mean   :0.349
 3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00   3rd Qu.:1.000
 Max.   :67.10   Max.   :2.4200           Max.   :81.00   Max.   :1.000
```

#Basic Visualizations

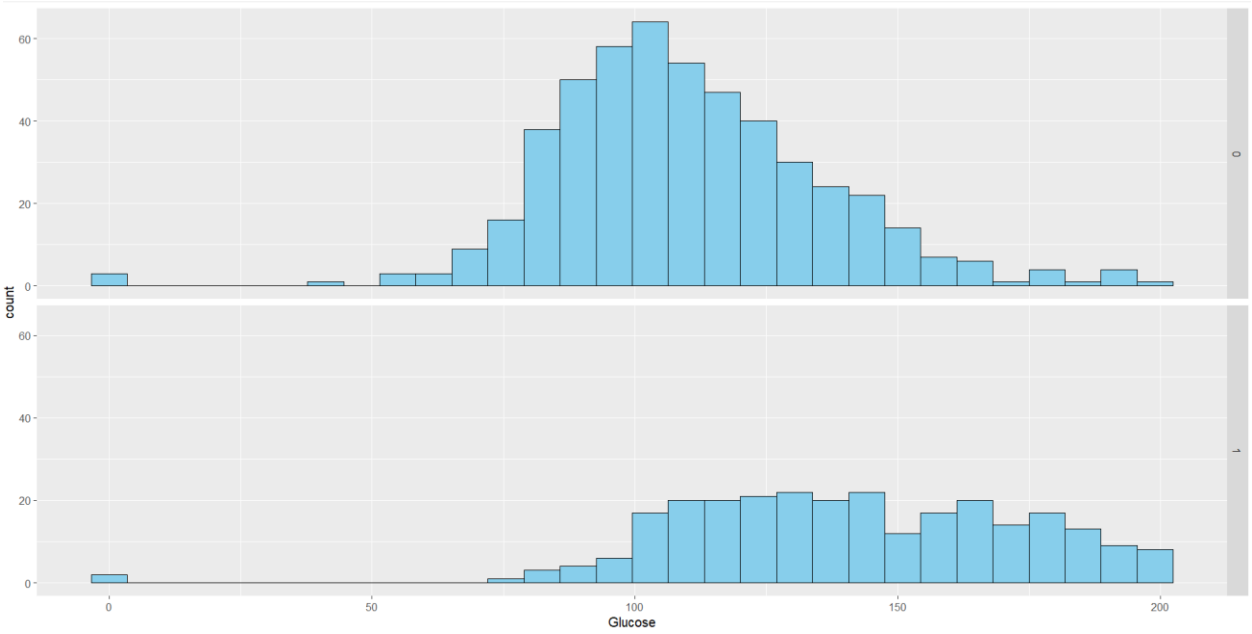
```
par(mfrow =c(2,2))                    # divide plotting area into 2x2 grid
hist(diabetes$Pregnancies)
hist(diabetes$Age)
hist(diabetes$Glucose)
hist(diabetes$BMI)
```



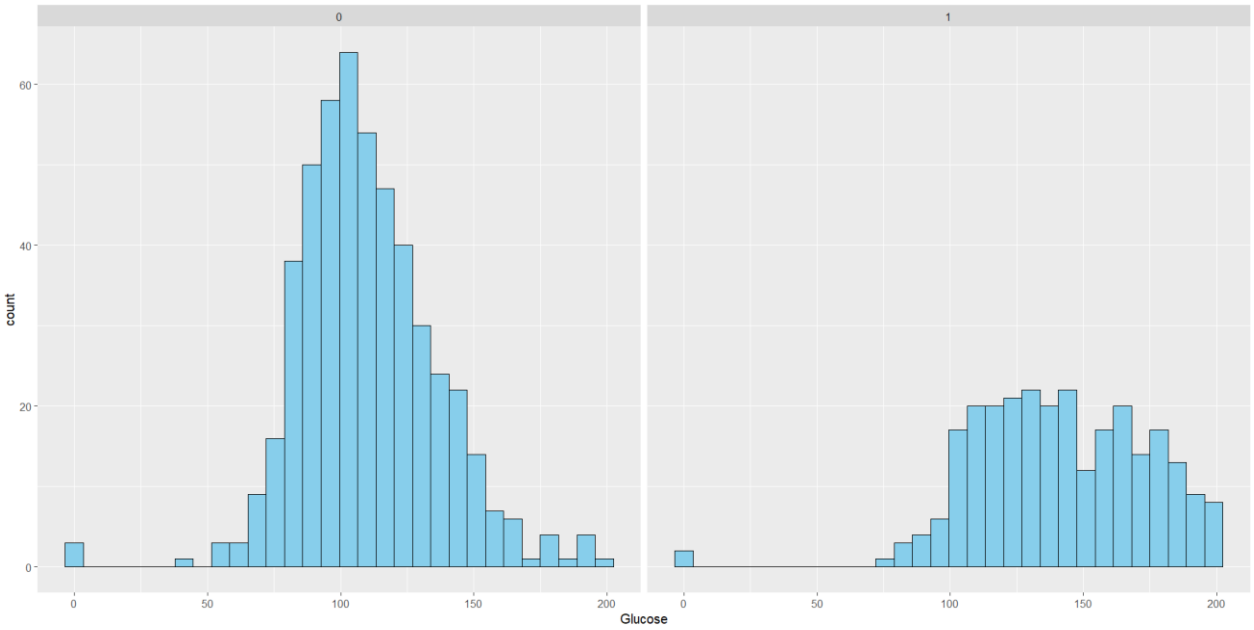

```
par(mfrow =c(1,1))                    # reset layout
boxplot(diabetes$BloodPressure, ylab = "BloodPressure")
```




```
install.packages("ggplot2")
library(ggplot2)
ggplot(diabetes,aes(x=Glucose))+geom_histogram(fill="skyblue",colour="black")+facet_grid(O
utcome~.)
```



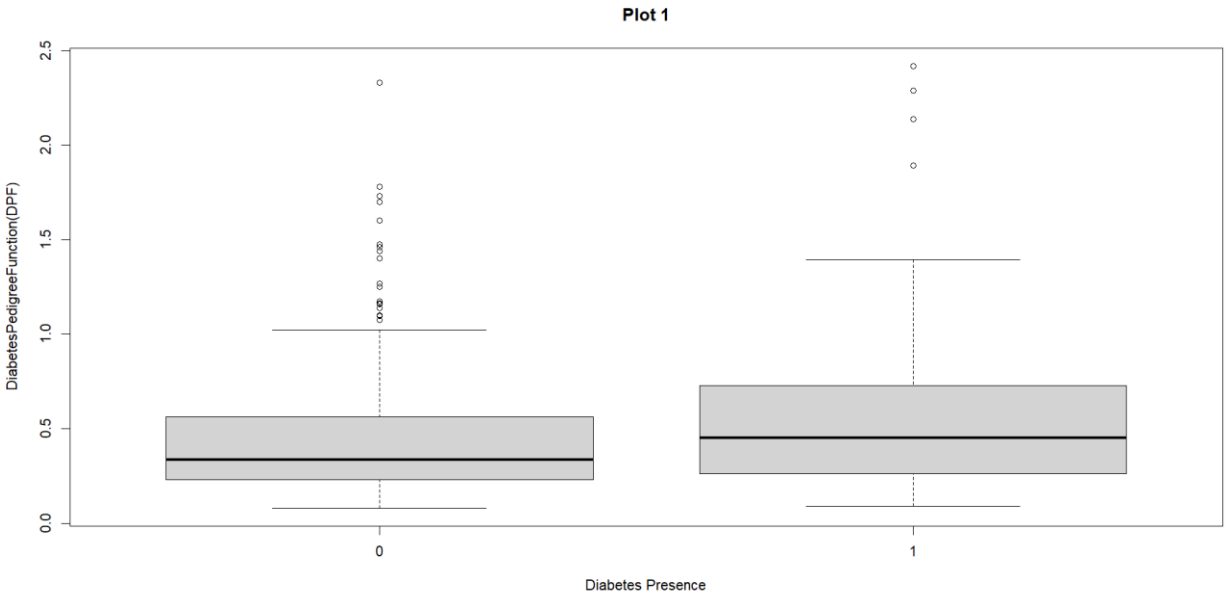
ggplot(diabetes,aes(x=Glucose))+geom_histogram(fill="skyblue",colour="black")+facet_grid(.~ Outcome)



#Statistical test
t.test(Glucose ~ Outcome, diabetes)

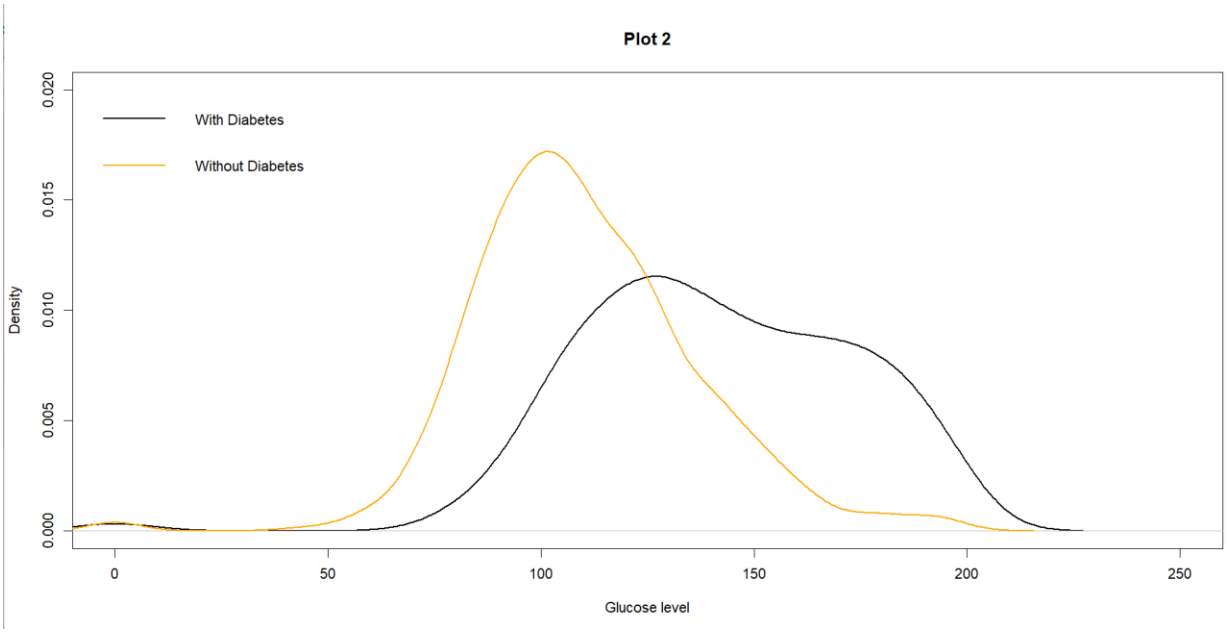
```
> t.test(Glucose ~ Outcome, diabetes)

        Welch Two Sample t-test

data:  Glucose by Outcome
t = -13.752, df = 461.33, p-value < 2.2e-16
alternative hypothesis: true difference in means between group 0 and group 1 is not equal t
o 0
95 percent confidence interval:
 -35.74707 -26.80786
sample estimates:
mean in group 0 mean in group 1
       109.9800        141.2575

>
```

```
#boxplots
with(diabetes, boxplot(DiabetesPedigreeFunction ~ Outcome,
                       ylab = "DiabetesPedigreeFunction(DPF)",
                       xlab = "Diabetes Presence",
                       main = "Plot 1",
                       outline = TRUE))
```




```
#densityplot

with_d <- diabetes [diabetes$Outcome == 1,]
without <- diabetes [diabetes$Outcome == 0,]


plot(density(with_d$Glucose),
     xlim = c(0, 250),
     ylim = c(0.00, 0.02),
```

```
    xlab = "Glucose level",
    main = "Plot 2",
    lwd = 2)
lines(density(without$Glucose),
     col = "orange",
     lwd = 2)
legend("topleft",
      col = c("black", "orange"),
     legend = c("With Diabetes", "Without Diabetes"),
     lwd = 2,
     bty = "n")
```



Welch Two Sample T-test

#two sample ttest with unequal variance
t.test(with_d$DiabetesPedigreeFunction, without$DiabetesPedigreeFunction)

```
> #two sample ttest with unequal variance
> t.test(with_d$DiabetesPedigreeFunction, without$DiabetesPedigreeFunction)

        Welch Two Sample t-test

data:  with_d$DiabetesPedigreeFunction and without$DiabetesPedigreeFunction
t = 4.5768, df = 454.51, p-value = 6.1e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.06891135 0.17262065
sample estimates:
mean of x mean of y
 0.550500  0.429734
```

#Correlation matrix

```
install.packages("GGally")
library(GGally)

ggcorr(diabetes[,-9], name = "corr", label = TRUE)+
  theme(legend.position = "none")+
  labs(title = "Correlation of Variance")+
  theme(plot.title=element_text(face='bold',color='black', hjust =0.5, size=12))
```

Logistic Regression

#Fitting a logistic regression to assess importance of predictors

```
method <- paste0(paste(names(diabetes)[length(diabetes)], collapse ="+"))
logistic <- glm(Outcome ~ ., family = binomial, data = diabetes)
```

logistic

```
> method <- paste0(paste(names(diabetes)[length(diabetes)], collapse ="+"))
> logistic <- glm(Outcome ~ ., family = binomial, data = diabetes)
> logistic

Call:  glm(formula = Outcome ~ ., family = binomial, data = diabetes)

Coefficients:
              (Intercept)               Pregnancies                   Glucose
                -8.404696                  0.123182                  0.035164
           BloodPressure             SkinThickness                   Insulin
                -0.013296                  0.000619                 -0.001192
                      BMI  DiabetesPedigreeFunction                       Age
                 0.089701                  0.945180                  0.014869

Degrees of Freedom: 767 Total (i.e. Null);  759 Residual
Null Deviance:      993.5
Residual Deviance: 723.4        AIC: 741.4
```

summary(logistic)

```
> summary(logistic)

Call:
glm(formula = Outcome ~ ., family = binomial, data = diabetes)

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              -8.4046964  0.7166359 -11.728  < 2e-16 ***
Pregnancies               0.1231823  0.0320776   3.840 0.000123 ***
Glucose                   0.0351637  0.0037087   9.481  < 2e-16 ***
BloodPressure            -0.0132955  0.0052336  -2.540 0.011072 *
SkinThickness             0.0006190  0.0068994   0.090 0.928515
Insulin                  -0.0011917  0.0009012  -1.322 0.186065
BMI                       0.0897010  0.0150876   5.945 2.76e-09 ***
DiabetesPedigreeFunction  0.9451797  0.2991475   3.160 0.001580 **
Age                       0.0148690  0.0093348   1.593 0.111192
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 993.48  on 767  degrees of freedom
Residual deviance: 723.45  on 759  degrees of freedom
AIC: 741.45

Number of Fisher Scoring iterations: 5
```

#Features Selection

```
Model_coeff <- exp(coef(logistic))[2:ncol(diabetes)]
Model_coeff <- Model_coeff[c(order(Model_coeff,decreasing=TRUE)[1:(ncol(diabetes)-1)])]
predictors_names <- c(names(Model_coeff),names(diabetes),names(diabetes)[length(diabetes)])
predictors_names <- unique(predictors_names)
```

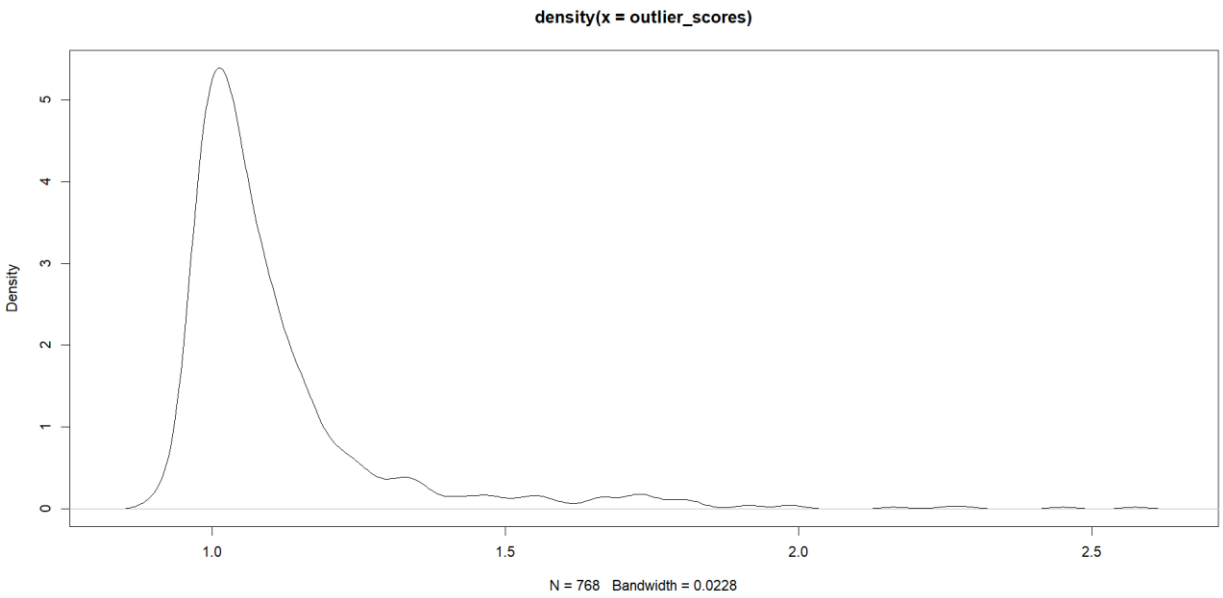
predictors_names

```
> #Features Selection
> Model_coeff <- exp(coef(logistic))[2:ncol(diabetes)]
> Model_coeff <- Model_coeff[c(order(Model_coeff,decreasing=TRUE)[1:(ncol(diabetes)-1)])]
> predictors_names <- c(names(Model_coeff),names(diabetes),names(diabetes)[length(diabete
s)])
> predictors_names <- unique(predictors_names)
> predictors_names
[1] "DiabetesPedigreeFunction" "Pregnancies"              "BMI"
[4] "Glucose"                  "Age"                      "SkinThickness"
[7] "Insulin"                  "BloodPressure"            "Outcome"
```

#filter df with n most important predictors
diabetes_df <- diabetes[, c(predictors_names)]
head(diabetes_df)

```
> #filter df with n most important predictors
> diabetes_df <- diabetes[, c(predictors_names)]
> head(diabetes_df)
  DiabetesPedigreeFunction Pregnancies  BMI Glucose Age SkinThickness Insulin
1                    0.627           6 33.6     148  50            35       0
2                    0.351           1 26.6      85  31            29       0
3                    0.672           8 23.3     183  32             0       0
4                    0.167           1 28.1      89  21            23      94
5                    2.288           0 43.1     137  33            35     168
6                    0.201           5 25.6     116  30             0       0
  BloodPressure Outcome
1            72       1
2            66       0
3            64       1
4            66       0
5            40       1
6            74       0
>
```

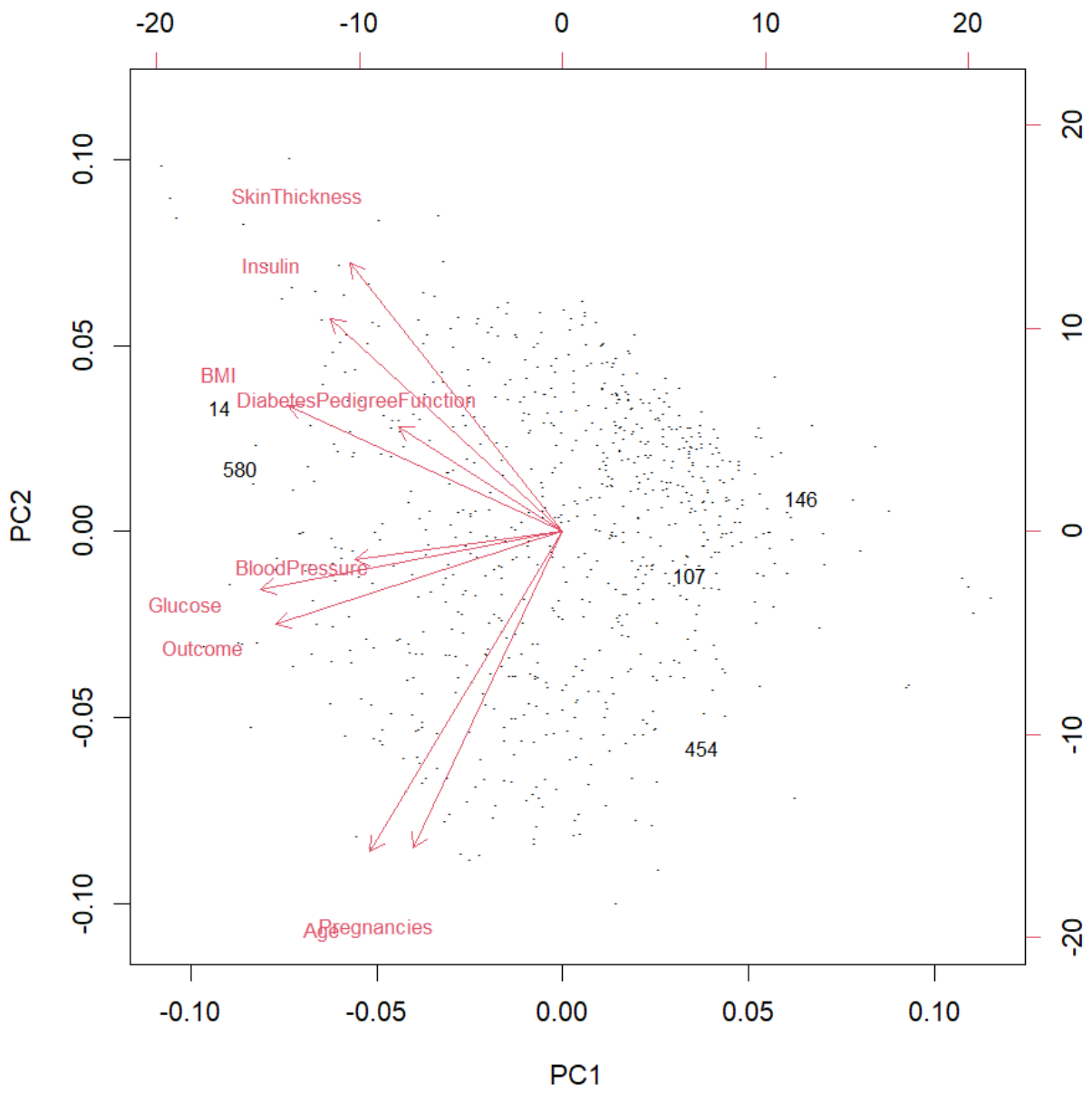
#outlier detection
install.packages("dbscan")
library(dbscan)
outlier_scores <- lof(diabetes_df, minPts=5)
plot(density(outlier_scores))




```
outliers <- order(outlier_scores, decreasing=T) [1:5]
print(outliers)
```

```
> print(outliers)
[1] 580 146  14 107 454
```

```
n <- nrow(diabetes_df)
labels <- 1:n
labels [-outliers] <- "."
pc <- prcomp(diabetes_df, scale. = TRUE)
biplot(pc, cex = 0.8, xlabs = labels)
```




```
install.packages("Rlof")
library(Rlof)
outlier_scores <- lof(diabetes_df, k=5)
outlier_scores <- lof(diabetes_df, k=c(5:10))
outlier_scores
```

```
> outlier_scores <- lof(diabetes_df, k=5)
> outlier_scores <- lof(diabetes_df, k=c(5:10))
> outlier_scores
               5         6         7         8         9        10
  [1,] 1.0614219 1.0297078 1.0468618 1.0419784 1.0388870 1.0543511
  [2,] 1.0714608 1.0562307 1.0344953 1.0391182 0.9969211 1.0019419
  [3,] 1.0788344 1.1152774 1.1539840 1.1709167 1.1495992 1.1507079
  [4,] 1.0310752 1.0242772 1.0312570 0.9994929 0.9938685 0.9917836
  [5,] 1.1173258 1.1697577 1.1500623 1.1633539 1.1486340 1.1607696
  [6,] 0.9906959 1.0008675 0.9880241 0.9811730 0.9676290 0.9504779
  [7,] 1.2264187 1.2124247 1.1996469 1.2052351 1.1890029 1.1781549
  [8,] 1.0369737 0.9901984 1.0031780 0.9801561 0.9579331 0.9599527
  [9,] 1.0690159 1.1614295 1.2047452 1.1905089 1.1939513 1.0870336
 [10,] 2.0218110 1.9454300 1.9234275 1.8711827 1.8712623 1.8085780
```

Data Modelling

1. **Basic GLM with all Variables**

```
# Create index for 70% training data
index <- sample(seq_len(nrow(diabetes)), size = 0.7 * nrow(diabetes))

# Split into training and test sets
train <- diabetes[index, ]
test  <- diabetes[-index, ]

#1st Model
First_Model <- glm(formula=Outcome~., family = binomial, data=train)
summary(First_Model)
```

```
Call:
glm(formula = Outcome ~ ., family = binomial, data = train)

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -8.1382357  0.8391733  -9.698  < 2e-16 ***
Pregnancies               0.1283833  0.0385589   3.330  0.00087 ***
Glucose                   0.0369026  0.0044071   8.373  < 2e-16 ***
BloodPressure            -0.0126189  0.0067937  -1.857  0.06325 .
SkinThickness             0.0009716  0.0083139   0.117  0.90697
Insulin                  -0.0014233  0.0010473  -1.359  0.17413
BMI                       0.0771013  0.0181149   4.256 2.08e-05 ***
DiabetesPedigreeFunction  0.8066029  0.3377739   2.388  0.01694 *
Age                       0.0094962  0.0113654   0.836  0.40342
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 680.81  on 536  degrees of freedom
Residual deviance: 494.61  on 528  degrees of freedom
AIC: 512.61

Number of Fisher Scoring iterations: 5
```

The result shows that the variables SkinThickness, BloodPressure, Insulin and Age are not statistically significant. p_values is > 0.01 so we can experiment by removing it.

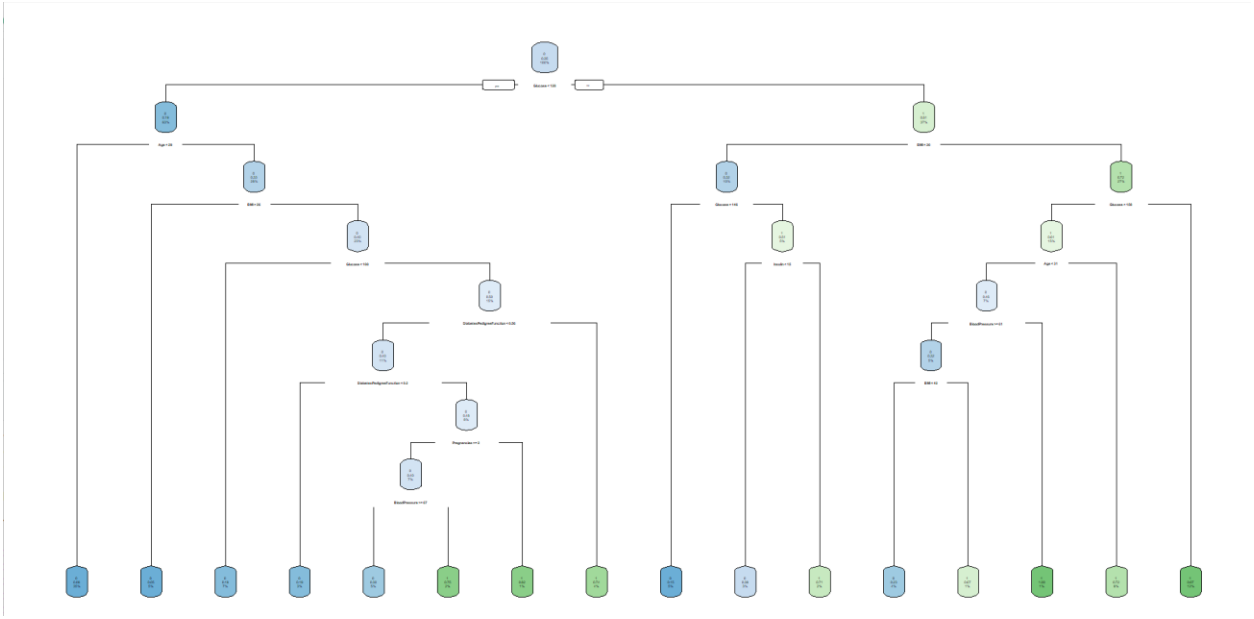
Stepwise logistic regression

model <- glm(Outcome ~ ., data = diabetes, family = binomial)
smodel <- step(model) #stepwise logistic regression

```
> model <- glm(Outcome ~ ., data = diabetes, family = binomial)
> smodel <- step(model) #stepwise logistic regression
Start:  AIC=741.45
Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
    Insulin + BMI + DiabetesPedigreeFunction + Age

                          Df Deviance    AIC
- SkinThickness            1   723.45 739.45
- Insulin                  1   725.19 741.19
<none>                         723.45 741.45
- Age                      1   725.97 741.97
- BloodPressure            1   729.99 745.99
- DiabetesPedigreeFunction 1   733.78 749.78
- Pregnancies              1   738.68 754.68
- BMI                      1   764.22 780.22
- Glucose                  1   838.37 854.37

Step:  AIC=739.45
Outcome ~ Pregnancies + Glucose + BloodPressure + Insulin + BMI +
    DiabetesPedigreeFunction + Age

                          Df Deviance    AIC
<none>                         723.45 739.45
- Insulin                  1   725.46 739.46
- Age                      1   725.97 739.97
- BloodPressure            1   730.13 744.13
- DiabetesPedigreeFunction 1   733.92 747.92
- Pregnancies              1   738.69 752.69
- BMI                      1   768.77 782.77
- Glucose                  1   840.87 854.87
>
```

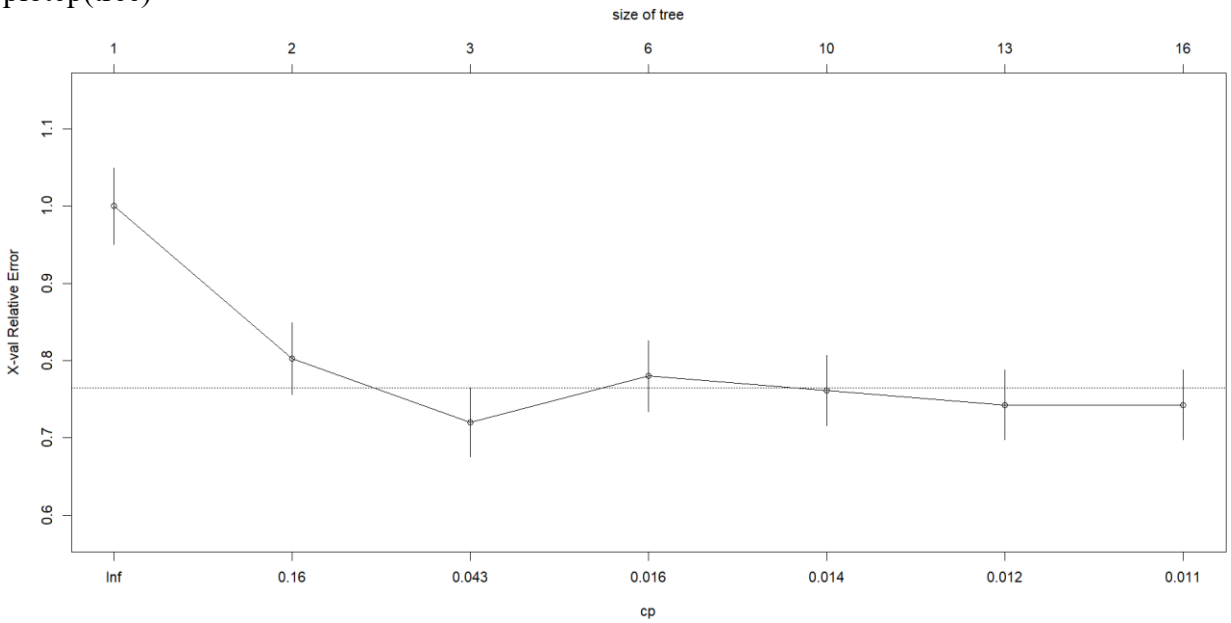
Decision Trees

```
install.packages("rpart")
library(rpart)
tree <- rpart(Outcome~., data=diabetes, method= "class")
install.packages("rpart.plot")
library(rpart.plot)
rpart.plot(tree)
```

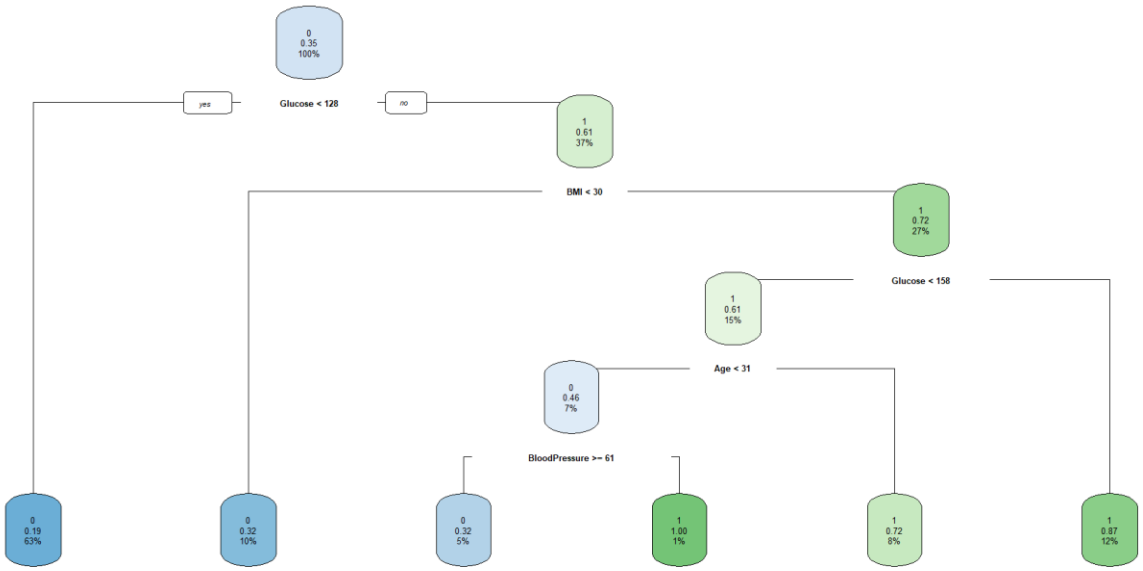


Complexity parameter

plotcp(tree)



tree1 <- rpart(Outcome~., data=diabetes, method ="class", cp=0.016)
rpart.plot(tree1)

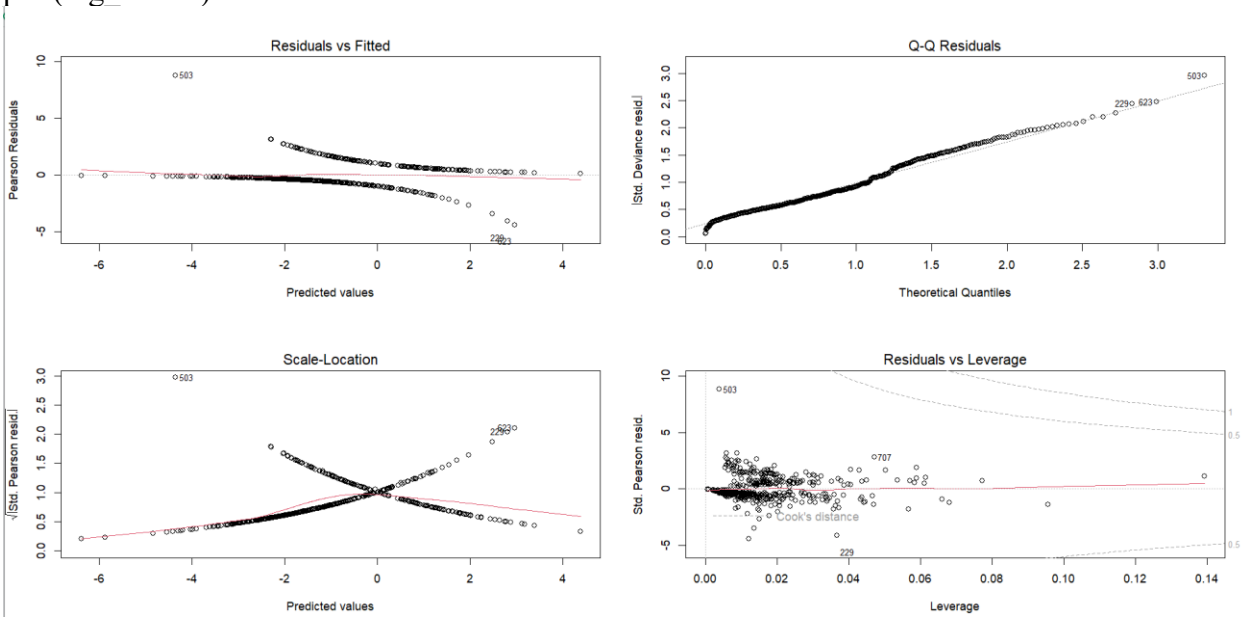


#2nd Model

```
set.seed(123)  #for reproducibility

log_model <- glm(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
                   Insulin + BMI + DiabetesPedigreeFunction,
                 family = binomial, data = train)
summary(log_model)
```

```
Call:
glm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
    SkinThickness + Insulin + BMI + DiabetesPedigreeFunction,
    family = binomial, data = train)

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)             -8.0017626  0.8185326  -9.776  < 2e-16 ***
Pregnancies              0.1441289  0.0338310   4.260 2.04e-05 ***
Glucose                  0.0376900  0.0043264   8.712  < 2e-16 ***
BloodPressure           -0.0115562  0.0066543  -1.737   0.0825 .
SkinThickness            0.0004376  0.0082976   0.053   0.9579
Insulin                 -0.0014652  0.0010477  -1.399   0.1619
BMI                      0.0759153  0.0180492   4.206 2.60e-05 ***
DiabetesPedigreeFunction 0.8181612  0.3372055   2.426   0.0153 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 680.81  on 536  degrees of freedom
Residual deviance: 495.31  on 529  degrees of freedom
AIC: 511.31

Number of Fisher Scoring iterations: 5
```

par(mfrow = c(2,2))
plot(log_model)



#3rd Model Predict Diabetes Risk on new patients using Decision Tree

```
install.packages("partykit")

# Load the library
library(partykit)


ct <- ctree(Outcome ~ ., data = train)

prediction_probability <- predict(ct, test, type = c("prob"))
prediction_class <- predict(ct, test, type = c("response"))

table(prediction_class, test$Outcome)
```

prediction_class	0	1
0.0833333333333333	68	6
0.25	16	9
0.261538461538462	37	25
0.404255319148936	9	13
0.733333333333333	9	20
0.833333333333333	1	18

```
library(caret)

# Make sure both are factors with same levels
prediction_class <- factor(prediction_class, levels = c(0, 1))
test$Outcome <- factor(test$Outcome, levels = c(0, 1))

# Confusion matrix
con_m <- confusionMatrix(prediction_class, test$Outcome,
                         positive = NULL,   # set positive class
                         dnn = c("Prediction", "Reference"))

con_m
```

```
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 0 0
         1 0 0

               Accuracy : NaN
                 95% CI : (NA, NA)
    No Information Rate : NA
    P-Value [Acc > NIR] : NA

                  Kappa : NaN

 Mcnemar's Test P-Value : NA

            Sensitivity :  NA
            Specificity :  NA
         Pos Pred Value :  NA
         Neg Pred Value :  NA
             Prevalence : NaN
         Detection Rate : NaN
   Detection Prevalence : NaN
      Balanced Accuracy :  NA

       'Positive' Class : 0
```

#4th Model
Naïve Bayes

```
Accuracy_p <- numeric(10)

for (l in 1:10)
  sample_size <- floor(0.90 * nrow(diabetes))

  # Convert Outcome to factor
  train$Outcome <- as.factor(train$Outcome)
  test$Outcome  <- as.factor(test$Outcome)

  install.packages("e1071")   #naivebayes
  library(e1071)
```

```
# Train Naive Bayes
  nb <- naiveBayes(Outcome ~ ., data = train)

  # Predict on test set
  z  <- predict(nb, test)

  # Confusion matrix & accuracy
  Acc <- table(test$Outcome, z)
  Accuracy_p[l] <- sum(diag(Acc)) / sum(Acc) * 100


# Store experiment results
Experiments <- c(1:10)
NAIVE_Bayes <- data.frame(Experiments, Accuracy_p)
NAIVE_Bayes
```

	Experiments	Accuracy_p
1	1	77.92208
2	2	77.92208
3	3	74.02597
4	4	75.32468
5	5	77.92208
6	6	68.83117
7	7	74.02597
8	8	76.62338
9	9	77.92208
10	10	70.12987

```
# Average accuracy across runs
Average <- mean(Accuracy_p)
Average
 > Average
 [1] 75.06494
```