DATA MINING ASSIGNMENT

Monika and Prakriti

PRN- 2134 &. 2145

2022-10-05

The objective of the analysis is to distinguish cancer versus normal patterns from mass-spectrometric data. This is a two-class classification problem with continuou s input variables. The samples include patients with cancer and healthy or control patients. The cancer group is denoted by 1.

First, we preprocess the given data by analyzing it and finding the most important variables among it using various variable selection approaches:

```
head(Data)

## # A tibble: 6 × 3,001
##       X1    X2    X3    X4    X5    X6    X7    X8    X9   X10   X11   X12
X13
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl>
## 1   167    37     0    36     0     0    34     0     0     0     0     0
73
## 2   195    78     0     0   113   110    28     8     0     0    88    12
65
## 3   135     6     0     0     0     0    24     0     0     0     0     0
76
## 4   156   133     0    18     0    57     0     0     0    21     0     0
34
## 5   398   160    20    14    13   190     0     0     0     0    21    18
36
## 6    57   552    42     0   131   453     0     0    12    29    17   206
0
## # … with 2,988 more variables: X14 <dbl>, X15 <dbl>, X16 <dbl>, X17 <dbl>,
## #   X18 <dbl>, X19 <dbl>, X20 <dbl>, X21 <dbl>, X22 <dbl>, X23 <dbl>,
## #   X24 <dbl>, X25 <dbl>, X26 <dbl>, X27 <dbl>, X28 <dbl>, X29 <dbl>,
## #   X30 <dbl>, X31 <dbl>, X32 <dbl>, X33 <dbl>, X34 <dbl>, X35 <dbl>,
## #   X36 <dbl>, X37 <dbl>, X38 <dbl>, X39 <dbl>, X40 <dbl>, X41 <dbl>,
## #   X42 <dbl>, X43 <dbl>, X44 <dbl>, X45 <dbl>, X46 <dbl>, X47 <dbl>,
## #   X48 <dbl>, X49 <dbl>, X50 <dbl>, X51 <dbl>, X52 <dbl>, X53 <dbl>, …
```

```
dim(Data)

## [1]  200 3001

Y=Data$Y   ## Response Variable
p=ncol(Data[,-Y]) # no. of predictors
p

## [1] 3000

str(Y)

##  num [1:200] 1 0 1 1 0 0 1 0 0 0 ...

table(Y)  ## 112 obs. is in one class and 88 obs. is in another class..

## Y
##   0   1
## 112  88
```

# VARIABLE SELECTION (PRE PROCESSING)

Feature selection methods are intended to reduce the number of input variables to those that are believed to be most useful to a model in order to predict the target variable.

Calculate the variance of the predictors:

```
v=c()
for (i in 1:p)
{
  v[i]=var(Data[,i])
}
length(v)

## [1] 3000

length(which(v==0))

## [1] 15
```

Remove the predictors whose variance is zero and new data set will be:

```
new_data=Data[,which(v!=0)]
D=data.frame(new_data)  ## frame the data
dim(D)

## [1]  200 2985

p1=ncol(D);p1

## [1] 2985

# check the p-value:

pval=c()
for (i in 1:(ncol(D)))
{
  model=glm(Y~D[,i],family = binomial(link = "logit"))
  pval[i]=summary(model)$coefficient[2,4]
}

P=which(pval<0.05)


#f the p-value is less than 0.05, we reject the null hypothesis that there's
no
#difference between the means and conclude that a significant difference does
exist.


#D=D[,P]  ## new data set whose p-value is less than 0.05


dim(D)

## [1]  200 2985

p11=ncol(D);p11

## [1] 2985
```

Check the normality:

```
s1=which(Y==0)
s2=which(Y==1)
D1=D[s1,]
D2=D[s2,]
dim(D1)

## [1]  112 2985

dim(D2)
```

```
## [1]   88 2985

table(Y)

## Y
##   0   1
## 112  88
```

Again check the variance of a new data set:

```
v1=c()
for (i in 1:ncol(D))
{
  v1[i]=var(D1[,i])
}
s11=which(v1==0)
length(s11)

## [1] 18

v2=c()
for (i in 1:ncol(D))
{
  v2[i]=var(D2[,i])
}
s22=which(v2==0)
length(s22)

## [1] 30

s=c(s11,s22)
D_1=D[,-(s)]
dim(D_1)

## [1]  200 2937

ncol(D_1)

## [1] 2937

pnorm=matrix(nrow = ncol(D_1),ncol = 2)
dim(pnorm)

## [1] 2937    2

for (i in 1:ncol(D_1))
{
  u=D_1[,i]
  pnorm[i,1]=shapiro.test(u[s1])$p.value
  pnorm[i,2]=shapiro.test(u[s2])$p.value
}

Norm=c()
for(i in 1:ncol(D_1))
{
  m=min(pnorm[i,])
  Norm[i]=ifelse(m<=0.05,1,0)
}
table(Norm)

## Norm
##    0    1
##    7 2930

length(which(Norm==1))

## [1] 2930

f1=which(Norm==0)  ##
Normx=D_1[,f1]
dim(Normx)

## [1] 200   7

f2=which(Norm==1)
NonNormx=cbind(D[,s],D_1[,f2])
dim(NonNormx)

## [1]  200 2978
```

Notes:

P-value ≤ α: The data do not follow a normal distribution (Reject H0).If the p-value is less than or equal to the significance level, the decision is to reject the null hypothesis and conclude that your data do not follow a normal distribution.

P-value > α: You cannot conclude that the data do not follow a normal distribution (Fail to reject H0) If the p-value is larger than the significance level, the decision is to fail to reject the null hypothesis. You do not have enough evidence to conclude that your data do not follow a normal distribution.

```
p2=ncol(Normx)   ## normal varibles
peqvar=c()
for(i in 1:p2)
{
  v=Normx[,i]
  peqvar[i]=var.test(v[s1],v[s2])$p.value
}
length(which(peqvar<0.05))

## [1] 1
```

```
VE=ifelse(peqvar<0.05,FALSE,TRUE)
table(VE)

## VE
## FALSE  TRUE
##     1     6

NormPVal=c()
for(i in 1:p2)
{
  v=Normx[,i]
  NormPVal[i]=t.test(v~Y,var.equal=VE[i])$p.value
}
S_Norm=which(NormPVal<0.05)
length(S_Norm)

## [1] 4

p3=ncol(NonNormx)
NonNormPVal=c()
for(i in 1:p3)
{
  v=NonNormx[,i]
  NonNormPVal[i]=wilcox.test(v~Y)$p.value
}

S_NonNorm=which(NonNormPVal<0.05)
length(S_NonNorm)

## [1] 1211

D3=cbind(Normx[,S_Norm],NonNormx[,S_NonNorm])
dim(D3)   ## final predictors=1215 out of 3000

## [1]  200 1215

c=cor(D3)
dim(c)

## [1] 1215 1215

library("caret")

## Loading required package: ggplot2

## Loading required package: lattice

non_sig<-findCorrelation(c,cutoff = 0.9)
non_sig<-sort(non_sig)
D3=D3[,-non_sig]
dim(D3)

## [1] 200 227
```

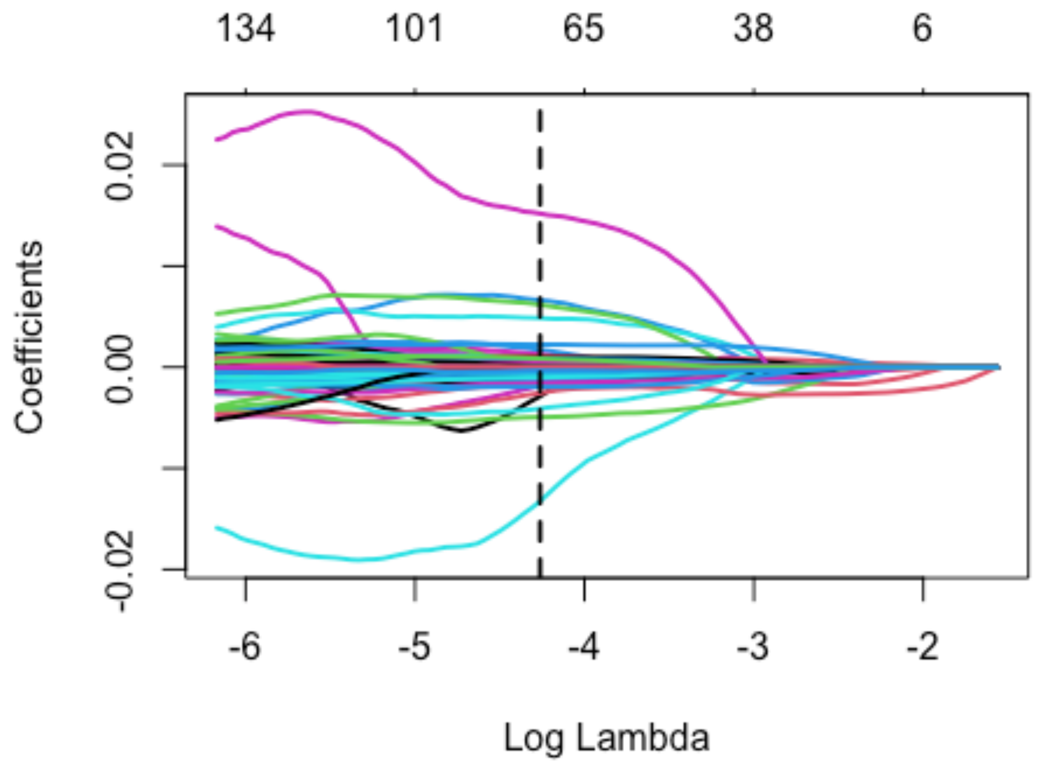
```
Final_data<-cbind(D3,Y)
dim(Final_data)

## [1] 200 228
```

Lasso regularisation:

Lasso regression is a classification algorithm that uses shrinkage in simple and sparse models(i.e model with fewer parameters). In Shrinkage, data values are shrunk towards a central point like the mean. Lasso regression is a regularized regression algorithm that performs L1 regularization which adds penalty equal to the absolute value of the magnitude of coefficients.

When lambda = 0, no parameters are eliminated.
As lambda increases, more and more coefficients are set to zero and eliminated & bias increases.
When lambda = infinity, all coefficients are eliminated.
As lambda decreases, variance increases.

```
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-4

Y=Final_data$Y
X <- model.matrix(Y~.,Final_data)[,-1]
lassocv <- cv.glmnet(X, Y, alpha=1, nfolds=10)
lassomdl <- glmnet(X,Y,alpha=1, nlambda=100)
lassocoef <- coef(lassomdl, s=lassocv$lambda.min)
plot(lassomdl, xvar="lambda", lwd=2)
abline(v=log(lassocv$lambda.min), col="black",lty=2, lwd=2)
```



```
w=lassocoef
length(which(w!=0))

## [1] 71

i=which(w!=0)

Final_data=Final_data[,i]
dim(Final_data)

## [1] 200  71

Y=Final_data$Y
```

####variable selection using Random forest(boruta algortihm)

The Boruta algorithm is a wrapper built around the random forest classification algorithm. It tries to capture all the important, interesting features you might have in your dataset with respect to an outcome variable.
1) First, it duplicates the dataset, and shuffle the values in each column. These

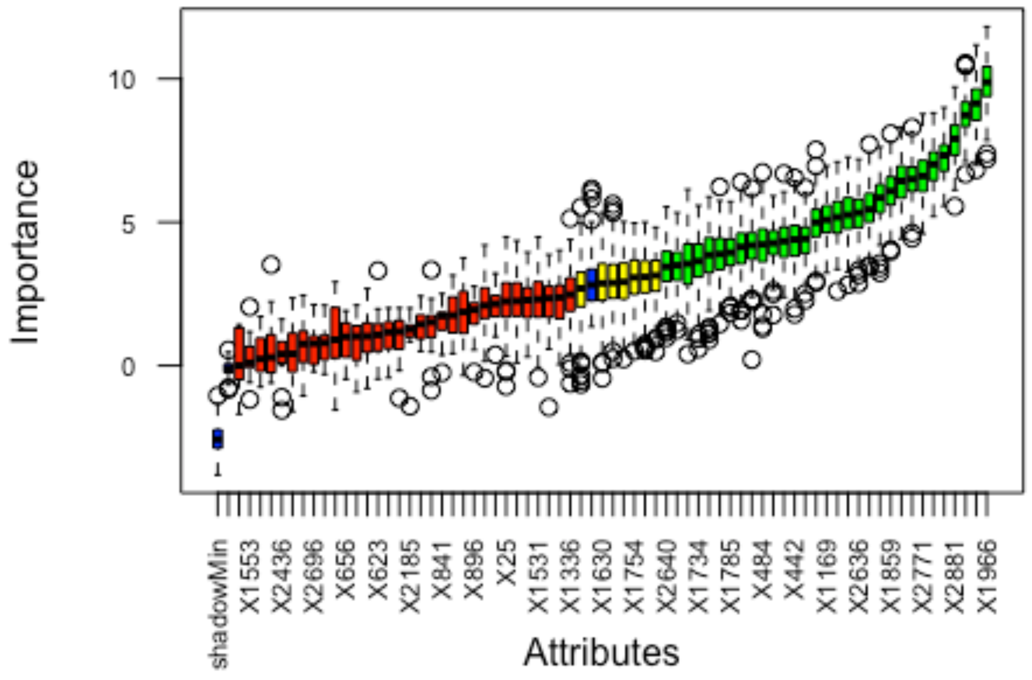
values are called shadow features. * Then, it trains a classifier, such as a Random Forest Classifier, on the dataset. By doing this, you ensure that you can an idea of the importance -via the Mean Decrease Accuracy or Mean Decrease Impurity- for each of the features of your data set. The higher the score, the better or more important.

2)Then, the algorithm checks for each of your real features if they have higher importance. That is, whether the feature has a higher Z-score than the maximum Z-score of its shadow features than the best of the shadow features. If they do, it records this in a vector. These are called a hits. Next,it will continue with another iteration. After a predefined set of iterations, you will end up with a table of these hits

.

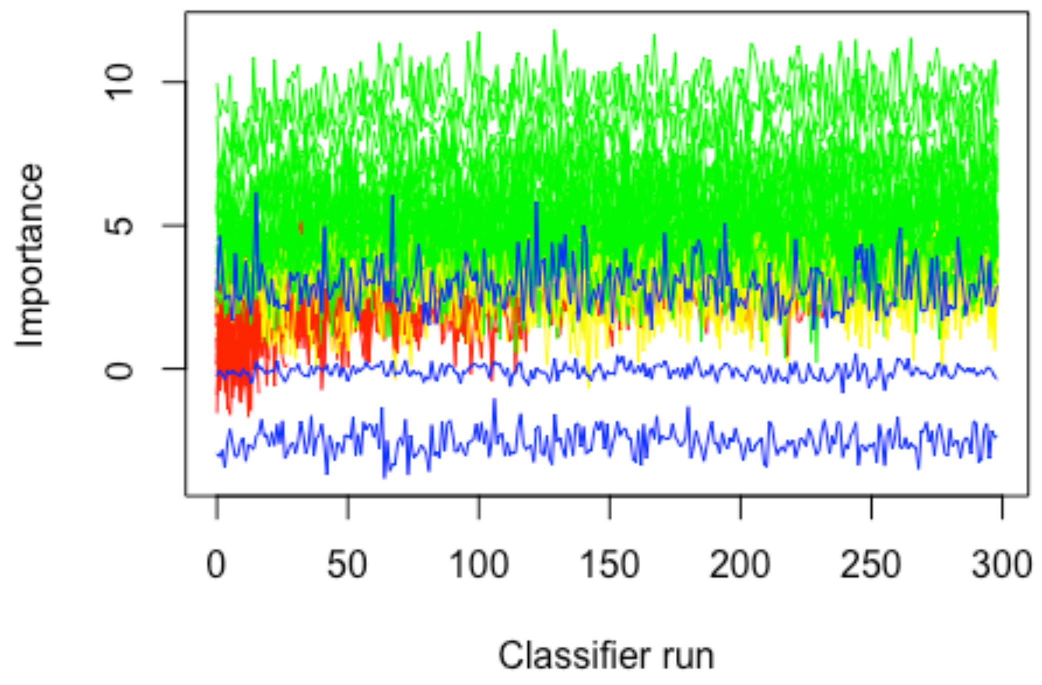
3)At every iteration, the algorithm compares the Z-scores of the shuffled copies of the features and the original features to see if the latter performed better than the former. If it does, the algorithm will mark the feature as important. In essence, the algorithm is trying to validate the importance of the feature by comparing with random shuffled copies, which increases the robustness.

```
library(Boruta)
library(mlbench)
library(caret)
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

set.seed(10)
boruta <- Boruta( Y~ ., data = Final_data, doTrace = 2, maxRuns = 300)

boruta

## Boruta performed 299 iterations in 8.018605 secs.
##  31 attributes confirmed important: X1116, X1138, X1169, X121, X1561
## and 26 more;
##  32 attributes confirmed unimportant: X1106, X1336, X1531, X1553, X2026
## and 27 more;
##  7 tentative attributes left: X1630, X1754, X186, X2631, X2691 and 2
## more;

plot(boruta, las = 2, cex.axis = 0.7)
```



Blue box corresponds to shadow attributes, green color indicates important attributes, yellow boxes are tentative attributes and red boxes are unimportant.

```
plotImpHistory(boruta)
```



```
bor <- TentativeRoughFix(boruta)
bor

## Boruta performed 299 iterations in 8.018605 secs.
## Tentatives roughfixed over the last 299 iterations.
##  33 attributes confirmed important: X1116, X1138, X1169, X121, X1561
## and 28 more;
##  37 attributes confirmed unimportant: X1106, X1336, X1531, X1553, X1630
## and 32 more;

Att=data.frame(attStats(bor))
dim(Att)

## [1] 70  6

Conf=which(Att[,6]=="Confirmed")
length(Conf)

## [1] 33

Final_data=Final_data[,Conf]
Final_data=cbind(Final_data,Y)  ### New final data1
dim(Final_data)

## [1] 200  34
```

```
library('randomForest')
```

##Using random forest for variable selection:

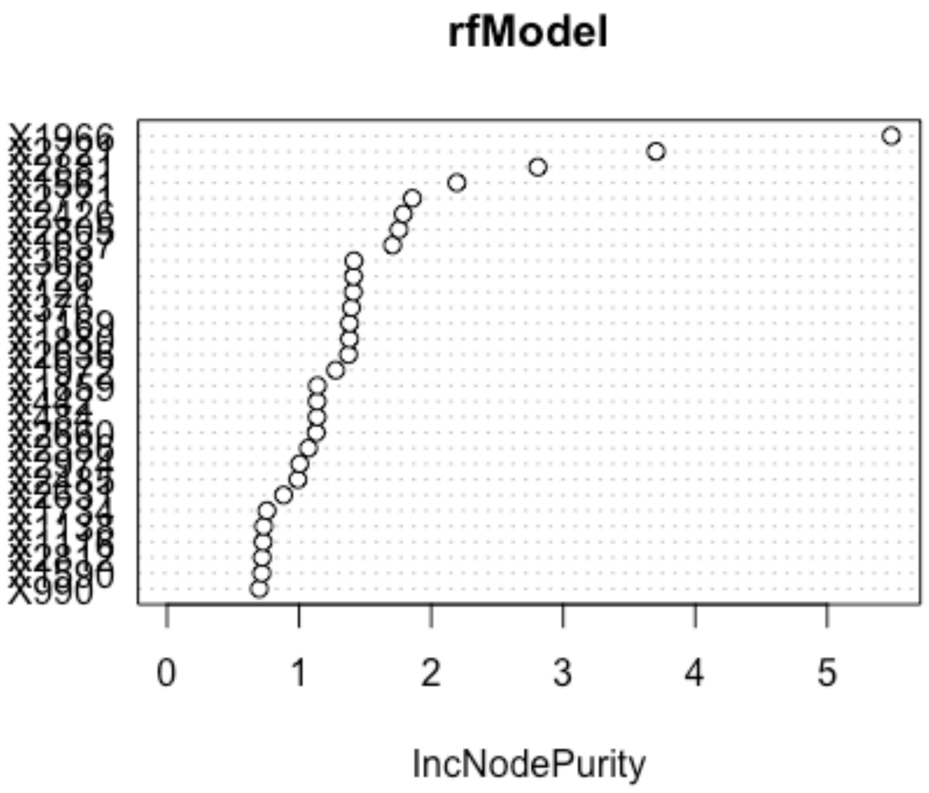
VIF random forest -

Variables with high importance are drivers of the outcome and their values have a significant impact on the outcome values. By contrast, variables with low importance might be omitted from a model, making it simpler and faster to fit and predict.

```
rfModel <-randomForest(Y ~ ., data = Final_data)

## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

Getting the list of important variables

```
S=importance(rfModel)
varImpPlot(rfModel)
```



```
C=which(S>=1.5)
length(C)

## [1] 8

Final_data=Final_data[,C]
Final_data=data.frame(Final_data,Y)   ### New final data
dim(Final_data)

## [1] 200   9




D=Final_data   ## Data we are going to use:
dim(D)
```

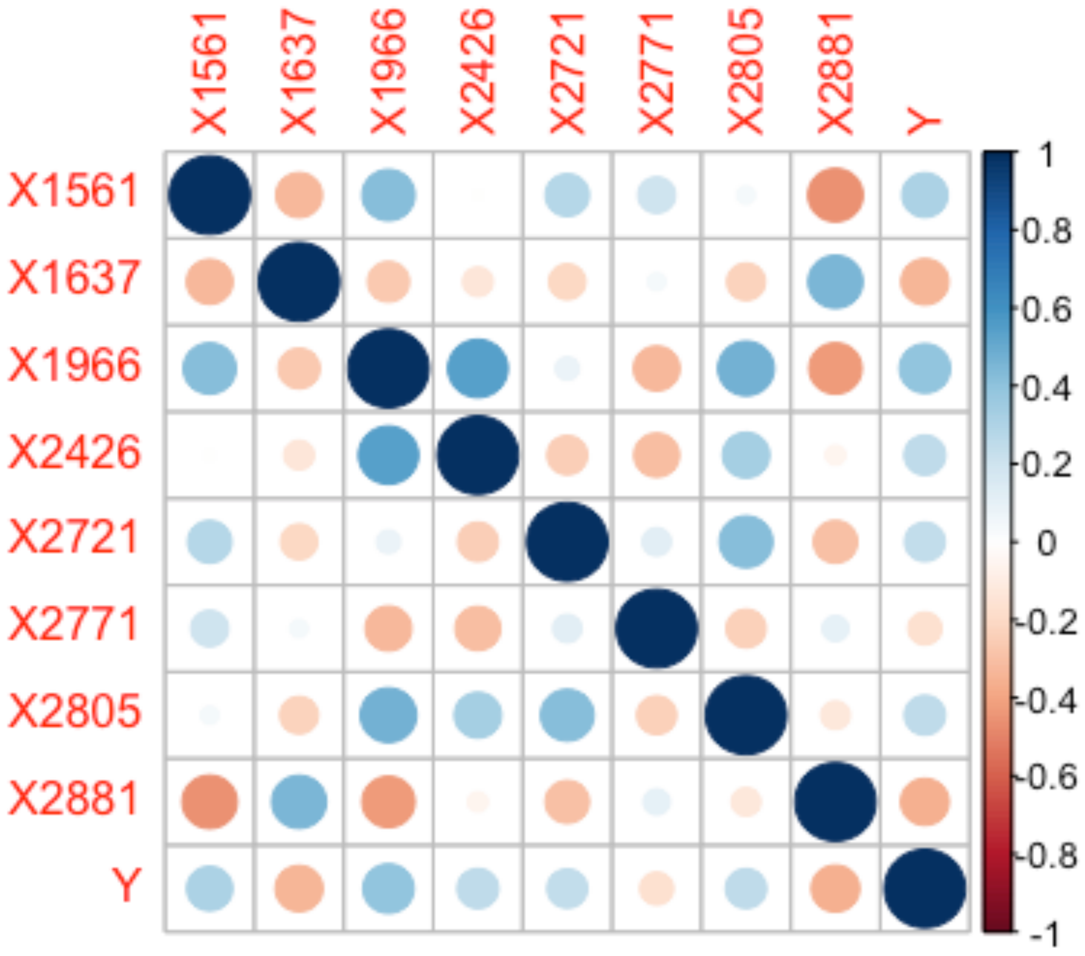
COMMENT- We found out that 8 variables are important for analysis of response variable Y by using feature selection techniques.

EXPLORATORY DATA ANALYSIS:

Loading corrplot

Creating correlation matrix for final dataset

```
library(corrplot)

## corrplot 0.92 loaded

D <- cor(Final_data[,c(1:9)])
corrplot(D, method = "circle")
```



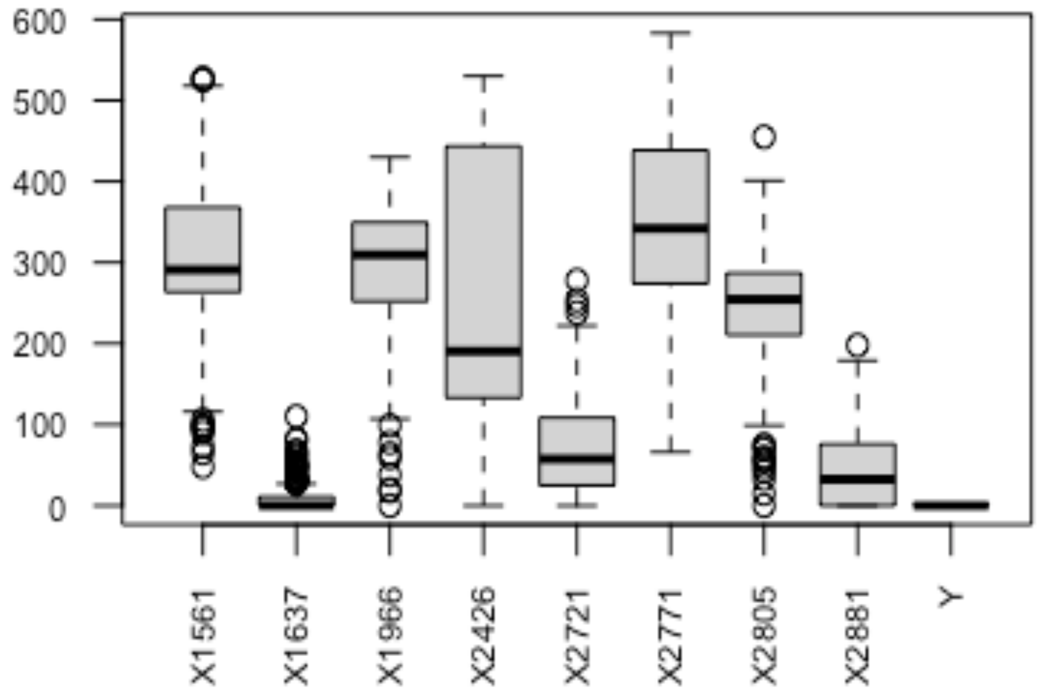
COMMENT- corrplot provides a visual exploratory tool on correlation matrix that supports automatic variable reordering to help detect hidden patterns among variables.

Boxplot viusalisation:

```
boxplot(Final_data,las = 2, cex.axis = 0.7)
```

COMMENT- A boxplot is a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile [Q1], median, third quartile [Q3] and "maximum")

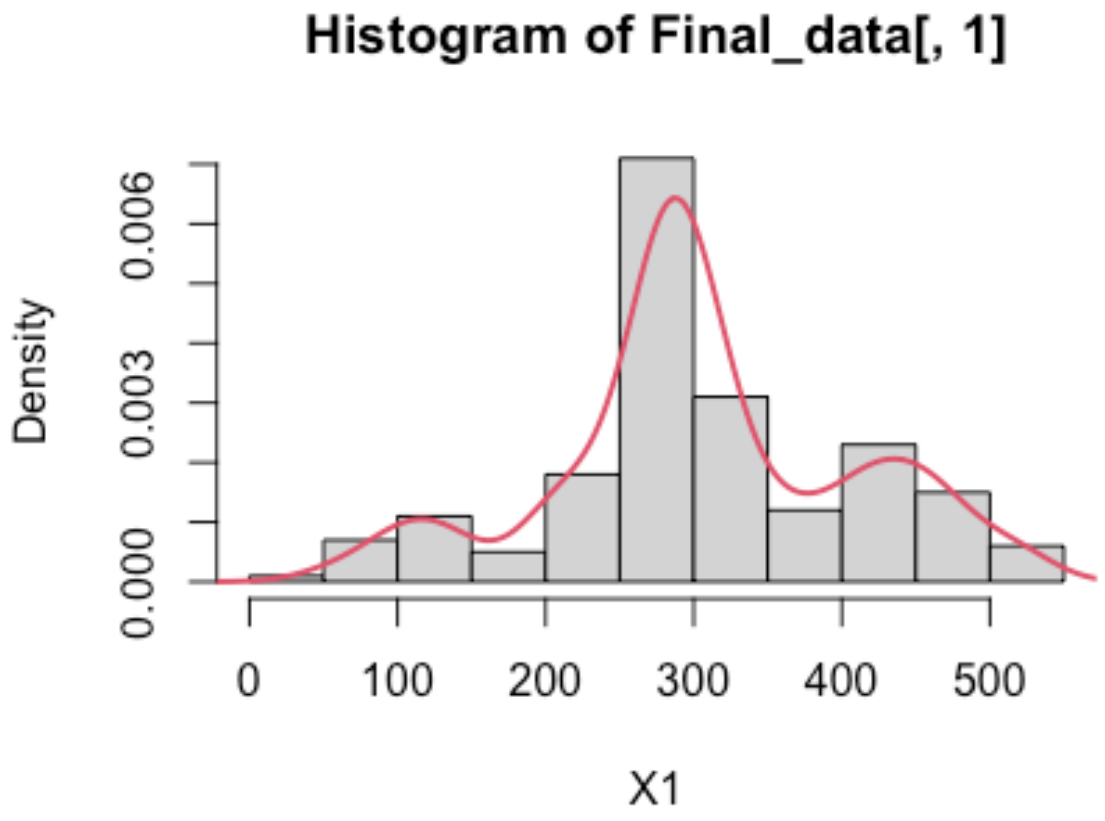
Boxplots can tell you about your outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped and if and how your data is skewed.



Histograms for EDA: Histogram -Superimposing a Density

A histogram can be used to compare the data distribution to a theoretical model, such as a normal distribution.

```
library(Hmisc)

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: Formula

##
## Attaching package: 'Hmisc'
## The following object is masked from 'package:e1071':
##
##     impute

## The following objects are masked from 'package:base':
##
##     format.pval, units

hist(Final_data[,1],probability = TRUE,xlab = "X1",ylab = "Density")
lines(density(Final_data[,1]),lwd=2,col=2)
```



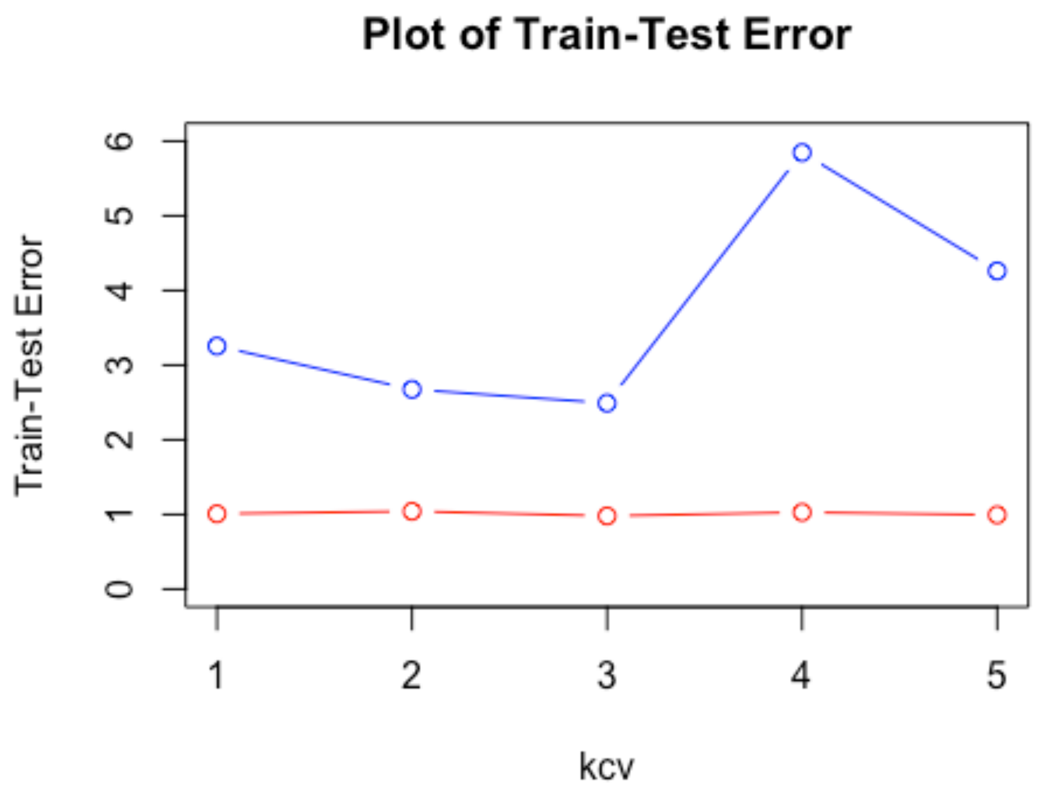
COMMENT-X1 i.e, 1st predictor is normaly distributed we can say by visualizing the histogram.

Similarly, we can visualize all other predictor variables.

PART 2} #k-fold cross validation for finding the best fitted model

1)Fitting logistic regression model:

```
D=Final_data   ## Data we are going to use:
dim(D)

## [1] 200   9

n=nrow(D);n

## [1] 200

kcv=5
Train_Err=Test_Err=c()
f=floor(n/kcv);f

## [1] 40

for (i in 1:kcv)
{
  set.seed(4)
  sam=sample(1:n,n,replace = FALSE)   ## sample
  T1=((i-1)*f+1):(i*f)
  Test_data=D[sam[T1],]
  Train_data=D[sam[-T1],]
  Model1=glm(Train_data$Y~.,data = Train_data[,-9],family = binomial(link = "
logit"))
  Train_Err[i]=mean(residuals(Model1)^2)
  Ypred=predict(Model1,newdata = Test_data)
  Test_Err[i]=mean((Test_data[,9]-Ypred)^2)
}
Train_Err

## [1] 1.0084573 1.0419455 0.9793515 1.0281133 0.9938302

Test_Err

## [1] 3.254842 2.674663 2.490670 5.846198 4.260820

Tot_TeE_log=mean(Test_Err)
Tot_TrE_log=mean(Train_Err)

plot(Train_Err,col="red",type = "b",main = "Plot of Train-Test Error",xlab =
"kcv",ylab = "Train-Test Error",ylim = c(0,6))
lines(Test_Err,col="blue",type="b")
```



Comment- test error using cv is higher then train error from logistic regression.

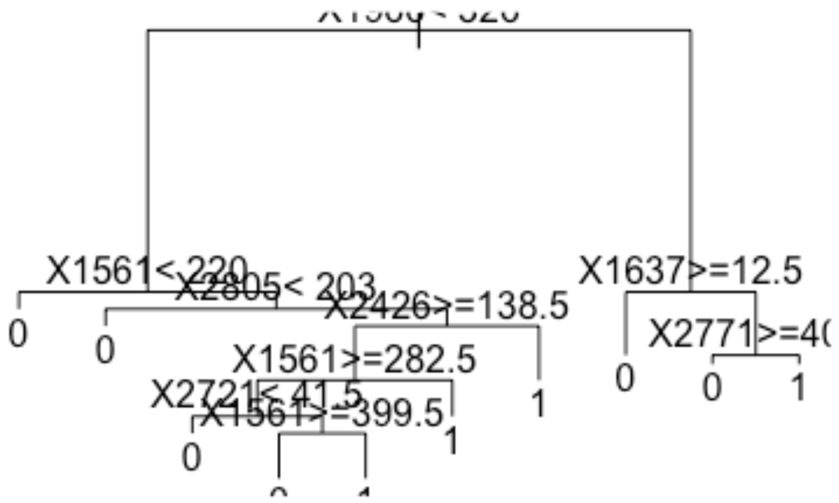
2) Fitting K-NN:

```
library(class)

KNN.PRED=knn(Train_data[,-9],Test_data[,-9],Train_data$Y,kcv)
table(KNN.PRED,Test_data$Y)

##
## KNN.PRED  0  1
##        0 16  2
##        1  8 14

miss_error=mean(KNN.PRED!=Test_data$Y);miss_error   ### misclassified error r
ate.

## [1] 0.25
```

Comment- Accuracy using KNN is 75% where as misclassification rate is 25%

3) CART:

```
library(rpart)
library(class)
D.rpart=rpart(formula = Y~.,data=Train_data,method = "class")
pred.tree=predict(D.rpart,Test_data,type = "class")
table(pred.tree,Test_data$Y)

##
## pred.tree  0  1
##         0 19  4
##         1  5 12

plot(D.rpart)
text(D.rpart,pretty = 0)
```



```
miss_err_tree=mean(pred.tree!=Test_data$Y);miss_err_tree    ## misclassified e
rror rate.

## [1] 0.225
```

Comment- Accuracy using CART is 77.5% where as misclassification rate is 22.5%

4) Naïve Bayes:

```
library(e1071)
library(caret)
library(ggplot2)
set.seed(1)
model_NV=naiveBayes(as.factor(Train_data$Y) ~ ., data = Train_data)
model_NV

##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##    0    1
## 0.55 0.45
##
## Conditional probabilities:
##    X1561
## Y       [,1]      [,2]
##   0 275.0455 114.14469
##   1 336.7222  76.42067
##
##    X1637
## Y       [,1]      [,2]
##   0 16.27273 23.193205
##   1  2.00000  5.634401
##
##    X1966
## Y       [,1]     [,2]
##   0 251.6818 94.70505
##   1 325.9861 65.63739
##
##    X2426
## Y       [,1]     [,2]
##   0 217.5114 150.6779
##   1 308.6250 155.9356
##
##    X2721
## Y       [,1]     [,2]
##   0 57.47727 52.95432
##   1 83.86111 60.07045
##
##    X2771
## Y        [,1]      [,2]
##   0 359.7386 109.59680
##   1 331.6944  88.71074
##
##    X2805
## Y        [,1]     [,2]
##   0 226.8409 87.62602
##   1 267.8056 56.98545
##
##    X2881
## Y        [,1]     [,2]
##   0 55.29545 45.84636
##   1 26.76389 37.20915
```

```
# Confusion Matrix
pred=predict(model_NV,newdata = Test_data[,-9])
table(pred,Test_data[,9])
```

```
##
## pred  0  1
##    0 13  3
##    1 11 13
```

```
miss_err_nav=mean(pred!=Test_data$Y);miss_err_nav  ## misclassified error rat
e.
```

```
## [1] 0.35
```

Comment- Accuracy using Naives Bayes is 65% where as misclassification rate is 35%

5)SVM:

```
library(e1071)
model_svm=svm(as.factor(Train_data$Y)~.,data = Train_data)
pred_svm=predict(model_svm,Test_data[,-9])
table(predict=pred_svm,truth=Test_data[,9])  ##confusion matrix
```

```
##        truth
## predict  0  1
##       0 19  4
##       1  5 12
```

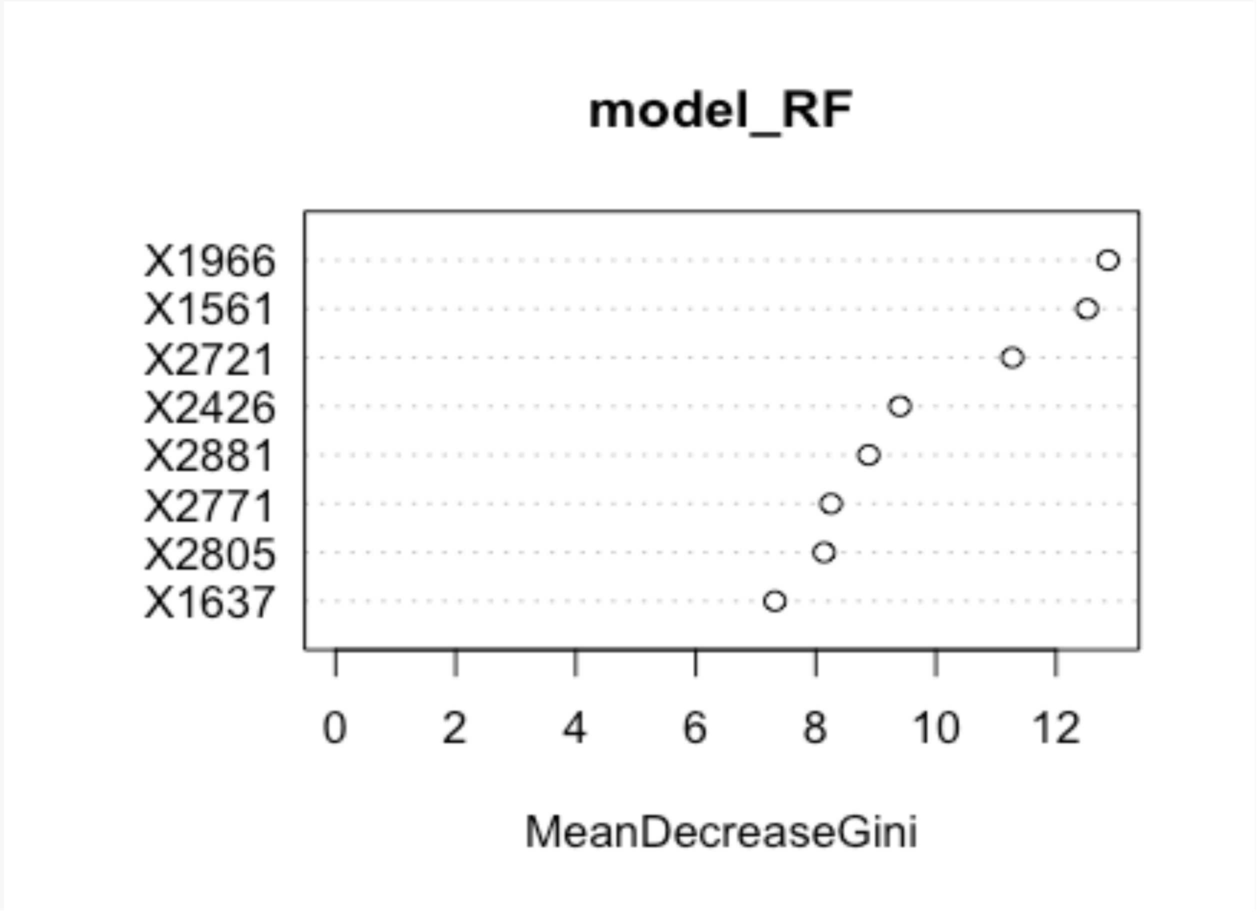
```
miss_err_svm=mean(pred_svm!=Test_data$Y);miss_err_svm ## misclassified error
rate.

## [1] 0.225
```

Comment- Accuracy using SVM is 77.5% where as misclassification rate is 22.5%

6)Random Forest:

```
library(randomforest)

model_RF=randomForest(as.factor(Train_data$Y)~.,data = Train_data,ntree=1000)

pred_RF=predict(model_RF,Test_data[,-9])

table(pred_RF,Test_data[,9])


pred_RF  0  1

       0   19  4

       1    5 12

> miss_err_RF=mean(pred_RF!=Test_data$Y);miss_err_RF  ## misclassified error rate.

[1] 0.225

> importance(model_RF)

     MeanDecreaseGini

X1561       12.518922

X1637        7.318361

X1966       12.868173

X2426        9.401428

X2721       11.273019

X2771        8.251608
```

```
X2805      8.138443

X2881      8.879709

> varImpPlot(model_RF)
```



Comment- Accuracy using Random forest is 77.5% where as misclassification rate is 22.5%. BY looking at the above plot we can say that X1966,X1561 influence the response Y most.

FINAL CONCLUSION: We observed that from all the fitted models,

SVM , Random Forest and CART works the best with misclassificatio n rate 22.5 % and accuracy score is 77.5% works the best .