***Santander Customer Transaction Prediction***

Aishwarya Kapoor

13th May 2019

**Content**

Section 1 ………………………………………………………………………………………. 3

Problem Statement

Section 2 ………………………………………………………………………………………. 6

Pre-processing of Data

Section 3 ………………………………………………………………………………………. 9

Modelling

Section 4 ………………………………………………………………………………………. 11

R-codes

Appendix ………………………………………………………………………………………. 19

**Project Name – Santander Customer Transaction Prediction**

***Section 1***

*Problem Statement*

Our client “[Santande](https://www.santanderbank.com/)​[r](https://www.santanderbank.com/)”, is on a mission to help people and businesses prosper. They are always looking for ways to help their customers to understand their financial health and identify which products and services might help them to achieve their monetary goals. Our client wants to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted.

We need to design a model which help our client to identify the customers which will continue a healthy business with our client in future.

*Data Type*

We have classification problem with 200 continuous variables with one binary variable. It is a classification problem. In this data we have around “202” variables including target variable and around “200000” observations.

**Number of attributes:**

* **ID\_codes** – it just tell us about the number of transactions
* **Var\_0 to Var\_199** – they all are continuous and independent variables.

**Dependent Variable:**

**Target**- Our main Target variable, it is our binary variable with 1 (yes) and 0 (no)

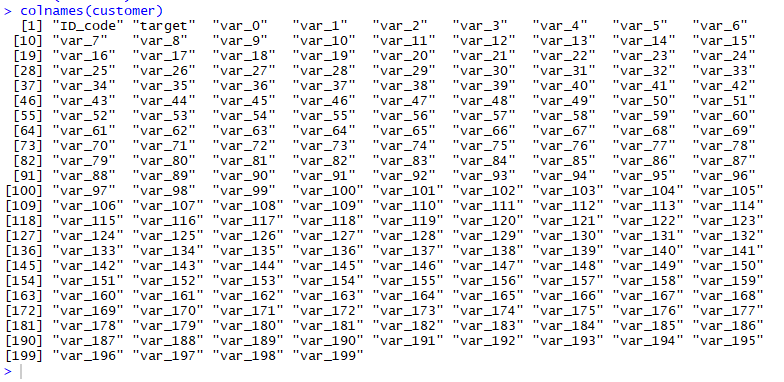


Image 1 – All the names of the variables present in our data

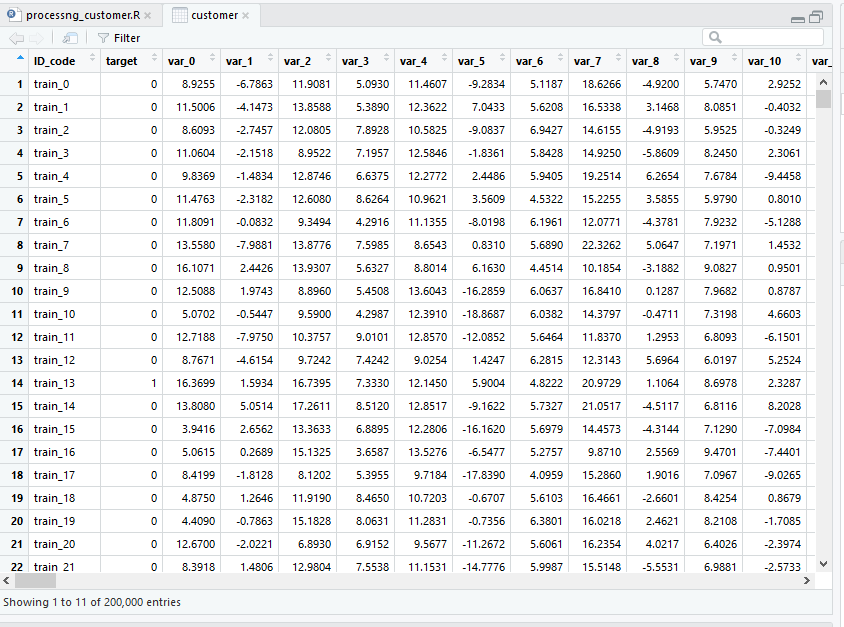


Image 2 – It the head of the data

Central tendency of the data like – mean, medium, etc. about each variable.

(In below screenshot only few of the variables has been shown rest all are present in appendix)

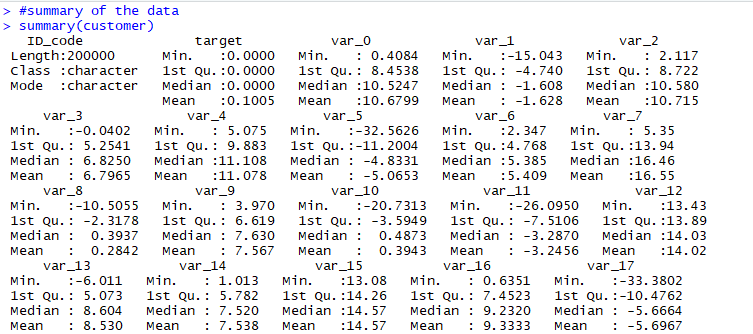


Image 3 – Central tendency of each variable

***Section 2***

*Pre-processing of the data*

We have noticed that we have a huge data. The number of observations in our data is huge. Our target variable “target” is a binary data in which 0 – no and 1 is yes which means that if it’s “1” that means that customer will do the transaction in the future irrespective of the value of the transactions and if it’s “0” customer will no longer continue business with our client.

As we have already seen that number of independent variables are too much so our approach for preprocessing will be different.

For pre-processing there are many steps which we are require to do, our very first step is “Removing NA’s” .

Step 1:

**Removing NA’s**

First we will check whether NA’s are present in our data or not. For which in R there is a very simple code(image is provided below for the reference)

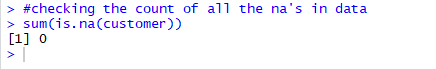


Image 4 – Missing value count

As, there is no missing value is present in our data. So, we can move forward.

*Step 2:*

Detecting the **outliers** in the data

We have around 200 independent variables for which using the boxplot method is not ideal solutions as it will consume too much time and money. As, we have multivariate data, we can use cook’s distance method for detecting the outliers.

In cook’s distance method for each observation it will measures the changes in y-hat for all observations with and without the presence of observation, so to know how much it will impact the y. It will consider all values which is greater than 4 times the mean as an outlier or influential value.

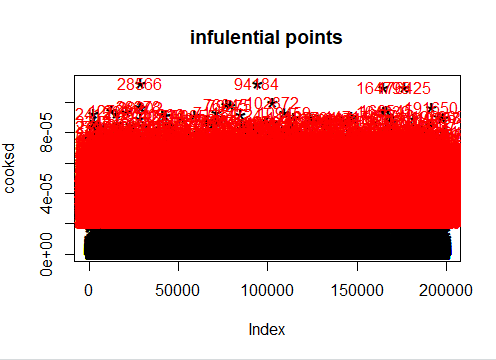


Image 5 – influential points in our data

All the red color points are the outliers in our data. We have detected all the outlier. Now, we need to replace them by using the capping method.

Capping method is a very simple and fast method to replace the outliers. It will consider the value as an outlier which lie outside the 1.5\*IQR.

It will use the IQR method to replace all the outlier in the data.

Lower limit of acceptable range = Q1 – 1.5\*(Q3-Q1)

Upper limit of acceptable range = Q3 + 1.5\*(Q3-Q1)

*Step 3:*

**Feature selection**

Feature selection is nothing but to select the all independent variables which are required to build to model.

We can’t use the traditional correlation plotting method in this case because of the number of independent variables. We are using the “ Boruta algorithm” for feature selection.

Boruta method is based on the random forest classification algorithm. It will create some random sample of the data and mix that sample with original data and after that will run the random forest algorithm using every variable each by each (first it will start with one variable and after that check the accuracy and then add one more variable in the algorithm and again check the accuracy and it will go on till the last variable) and after that it will select all the variables which are important for the model and which are not.

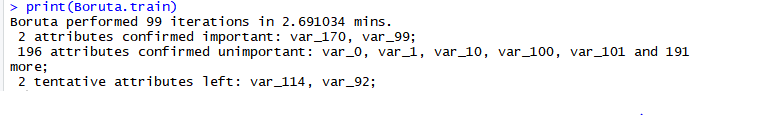


Image 6 – Feature selection

We using only 3 variables from this data that are :

* Target variable
* Var\_99
* Var\_170

*Step 4:*

**Feature scaling**

We are using the normalization to scale the data because our data is not normally distributed.

We scale our data because it helps to normalize the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm.

After this process our data will look this:

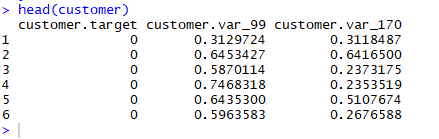


Image 7 – Data after scaling

***Section 3***

*Modelling*

We are using supervised Machine learning algorithms are procedures to automatically generalize from historical observations

We have binary problem for which we need to run the classification algorithms. We are using the supervised machine learning.

We have selected 2 supervised machine learning for our project and those are:

* Decision Tree
* Random Forest

First we will do “Decision Tree”. Decision tree use information gain to check the parent node. Parent node selected on the basis of highest value of information gain and further it will split the node and create the branches and leave. It will very easy to read and present.

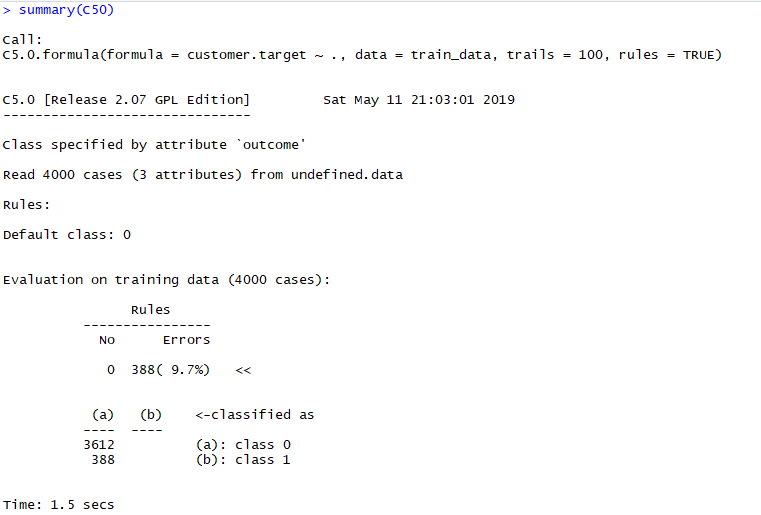


Image 8 – Summary of Decision Tree

Second model is “Random Forest”. This algorithm works on gini index in this it check which variables are not providing much information in the data will become the nodes and leave of the model.

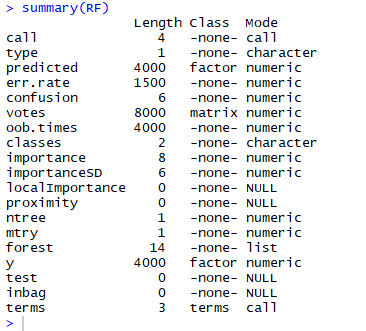


Image 9 – Summary of Decision Tree

Using Error matrix we would like to freeze the “Decision Tree model”

The reason being Accuracy of the decision tree is more than the regression model.

***Section 4***

*R codes*

#Clearing R enviroment

rm(list = ls(all = T))

#setting the working directory

setwd("C:/Users/Lenovo/Desktop/Edwisor/Project 3")

#data

read.csv("train\_customer.csv", header = TRUE , stringsAsFactors = FALSE) -> customer

customer<- customer[1:5000,]

#summary of the data

summary(customer)

head(customer)

colnames(customer)

View(customer)

#removing one variable we removed variable "ID\_Code"

customer <- customer[, -1, drop = FALSE]

#Loading the library

library(DMwR)

library(rpart)

install.packages("ggplot")

library(ggplot2)

library(corrgram)

install.packages("ggpubr")

library(ggpubr)

install.packages("devtools")

install.packages("dpylr")

install.packages("C50")

library(C50)

library(MASS)

library(dplyr)

install.packages("RODBC")

library(RODBC)

install.packages("outliers")

library(outliers)

install.packages("car")

library(car)

install.packages("Boruta")

library(Boruta)

install.packages("Metrics")

library(Metrics)

library(randomForest)

install.packages("ggthemes")

library(ggthemes)

#histgram

hist(customer$var\_0)

#calculating the standard deviation pf the data

class(customer$var\_0)

apply(customer, 2, sd, na.rm = TRUE) -> sd

#checking the count of all the na's in data

sum(is.na(customer))

#we dont have any missing value in the data so it doesn't require to do the NA Removing process

#customer[] <- lapply(customer[], as.numeric)

# now, we can move further for our Second step outliers, we are using the cooks Distance

#first we will bliud the model and on that bases we will bliud the model

mod <- lm(target ~., data = customer)

cooks.distance(mod) -> cooksd

plot(cooksd, pch = "\*", cex = 2, main = "infulential points")

abline(h = 4\*mean(cooksd, na.rm = T), col = "red")

text(x=1:length(cooksd)+1, y=cooksd, labels = ifelse(cooksd >4\*mean(cooksd, na.rm = T), names(cooksd), ""), col = "red")

influential <- as.numeric(names(cooksd)[(cooksd >4\*mean(cooksd, na.rm = T))])

head(customer[influential,])

#we doing outliers test using car package

car:: outlierTest(mod)

#it show that 94184 in this row it has the most extreme values

#imputation of the outliers we are using the capping function

x<- as.data.frame(customer)

caps <- data.frame(apply(customer,2, function(x){

quantiles <- quantile(x, c(0.25, 0.75))

x[x < quantiles[1]] <- quantiles[1]

x[x > quantiles [2]] <- quantiles[2]

}))

caps

#step 3 doing the correlation

#checking the VIF

library(usdm)

vif(customer[,-1])

VIF <- order(VIF)

View(VIF)

# it shows the 1.3/4 value for each variable

#feature selection of the data using the Boruta method

#Boruta method

set.seed(123)

Boruta.train <- Boruta(target~., data = customer, doTrace = 2)

print(Boruta.train)

#Boruta performed 99 iterations in 7.526 hrs.

#2 attributes confirmed important: var\_170, var\_99;

#196 attributes confirmed unimportant: var\_0, var\_1, var\_10, var\_100, var\_101 and 191 more;

#2 tentative attributes left: var\_114, var\_92;

#Ploting graph

plot(Boruta.train, xlab = "", xaxt = "n")

lz<-lapply(1:ncol(Boruta.train$ImpHistory),function(i)

Boruta.train$ImpHistory[is.finite(Boruta.train$ImpHistory[,i]),i])

names(lz) <- colnames(Boruta.train$ImpHistory)

Labels <- sort(sapply(lz,median))

axis(side = 1,las=2,labels = names(Labels),

at = 1:ncol(Boruta.train$ImpHistory), cex.axis = 0.7)

#tentative variable test

final.boruta <- TentativeRoughFix(Boruta.train)

print(final.boruta)

getSelectedAttributes(final.boruta, withTentative = F)

boruta.df <- attStats(final.boruta)

#selecting variables

customer <- data.frame(customer$target, customer$var\_99, customer$var\_170)

colnames(customer)

View(customer)

#normalization method

cnames = c("customer.var\_99", "customer.var\_170")

for (i in cnames) {

customer [,i] = (customer[,i] - min(customer[,i]))/ (max(customer[,i] - min(customer[,i])))

}

head(customer)

##Building model

#Decision Tree

#creating smaple for the data using simple sample method because all variables are continous

#we are using only Train data for our testing of data will use Test data later

library(caret)

library(C50)

library(DataCombine)

rmExcept("customer")

customer <- lapply(customer, as.factor)

customer <- as.data.frame(customer)

train\_index = sample(1:nrow(customer), 0.8\*nrow(customer))

train\_data = customer[train\_index,]

test\_data = customer[-train\_index,]

C50 <- C5.0(customer.target~., train\_data, trails = 100, rules = TRUE)

write(capture.output(summary(C50)), "summary\_customer.text")

prediction <- predict(C50, test\_data[,-1], type = "class")

#error matrix

install.packages("e1071")

library(e1071)

conmatrix <- table(test\_data$customer.target, prediction)

confusionMatrix(conmatrix)

#false negative rate

#FNR <- FN/(FN+TP)

FNR <- 88/(88+68)

#tPR = tp+(tp+fn)

tpr = 893/(893+107)

tpr

#FNR = 56.41

#accuracy = 89.3%

#tpr = 89.3%

#Random forest

library(DMwR)

install.packages("inTrees")

library(inTrees)

install.packages("caret")

library(caret)

library(randomForest)

rmExcept("customer")

customer <- (customer$customer.target, as.factor)

customer<- as.numeric(as.character(customer))

customer<- as.data.frame(customer)

train\_index = sample(1:nrow(customer), 0.8\*nrow(customer))

train\_data = customer[train\_index,]

test\_data = customer[-train\_index,]

RF <- randomForest(customer.target~., train\_data, importance = TRUE)

treelist <- RF2List(RF)

rules <- extractRules(treelist, train\_data[,-1])

rules[5,]

readrule <- presentRules(rules, colnames(train\_data))

head(readrule)

rulematrix <- getRuleMetric(rules, train\_data[,-1], train\_data$customer.target)

head(rulematrix)

prediction <- predict(RF, test\_data[,-1])

confmt <- table(test\_data$customer.target, prediction)

confusionMatrix(confmt)

conmatrix(confusionmt)

summary(RF)

#accuracy = 88.4%

#true postive rate

#tPR = tp+(tp+fn)

tpr = 884/(884+110)

tpr

#tpr = 88.93%

#fnr = 61.06%

***Appendix***

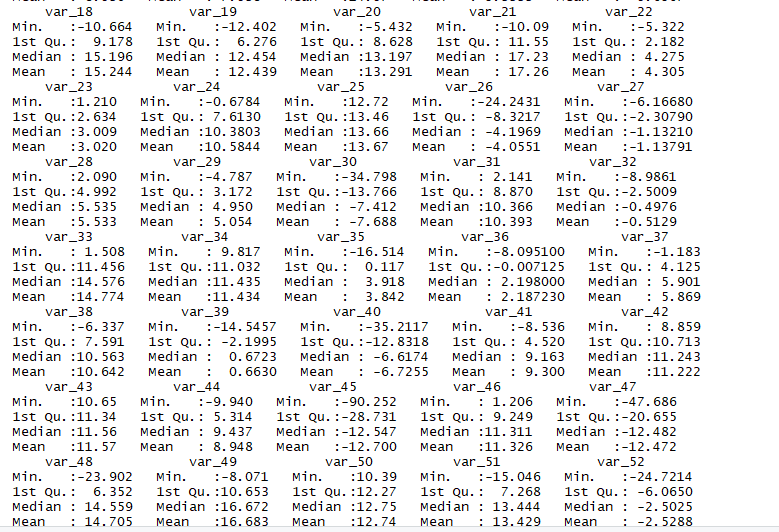
*Libraries and packages which are used*:

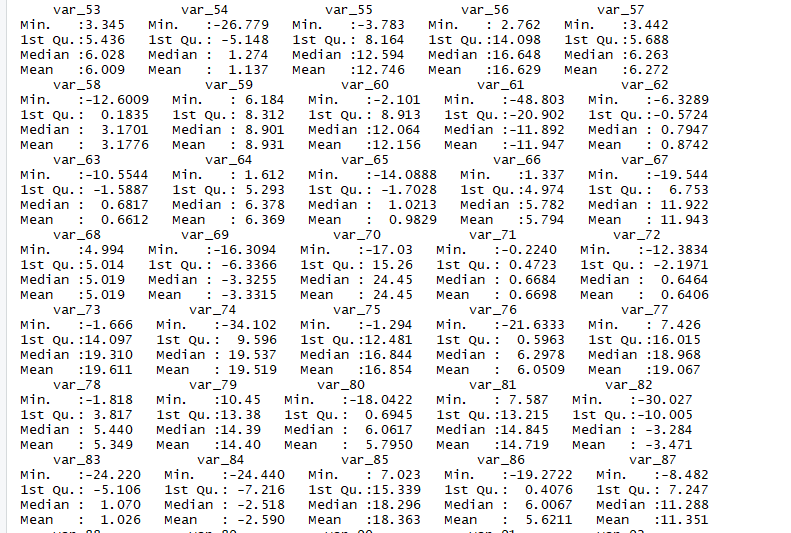
* install.packages("usdm")
* library(usdm)
* library(DMwR)
* library(rpart)
* install.packages("ggplot")
* library(ggplot2)
* library(corrgram)
* install.packages("ggpubr")
* library(ggpubr)
* install.packages("devtools")
* install.packages("dpylr")
* install.packages("C50")
* library(C50)
* library(MASS)
* install.packages("RODBC")
* library(RODBC)
* install.packages("outliers")
* library(outliers)
* install.packages("car")
* library(car)
* install.packages("Boruta")
* library(Boruta)
* install.packages("Metrics")
* library(Metrics)
* library(randomForest)
* install.packages("ggthemes")
* library(ggthemes)

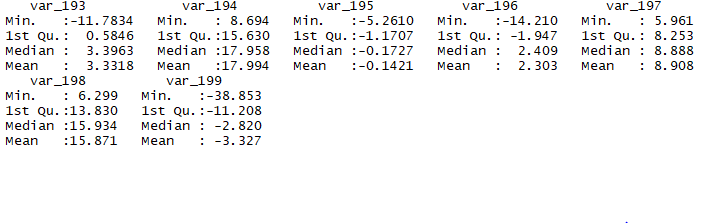
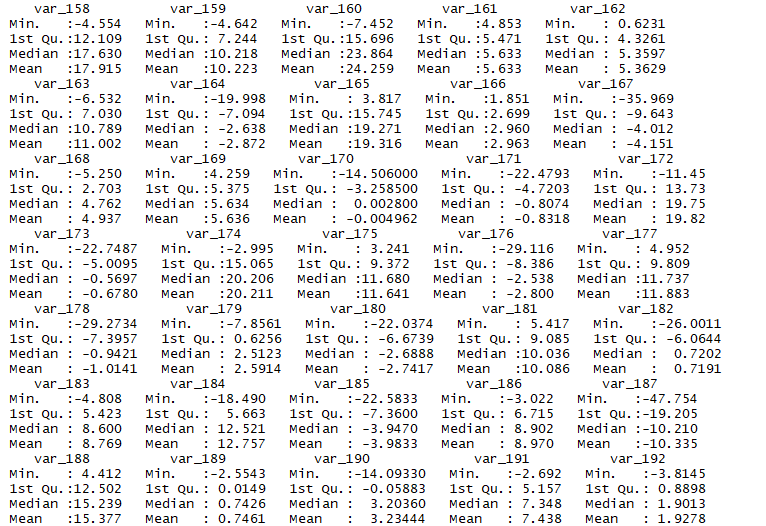
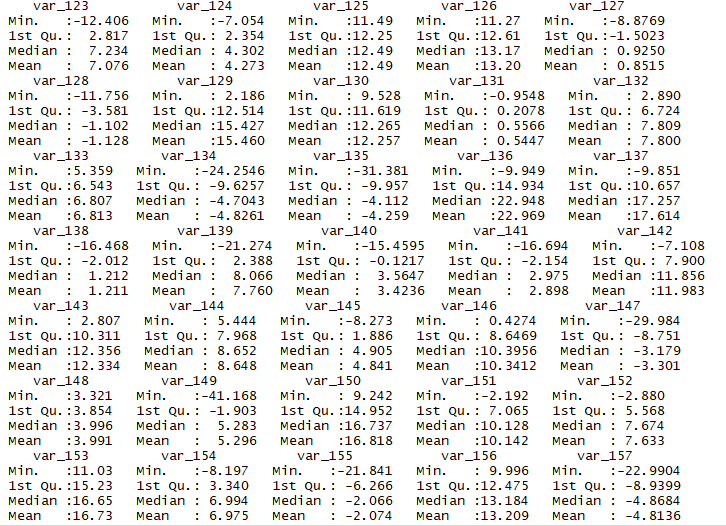
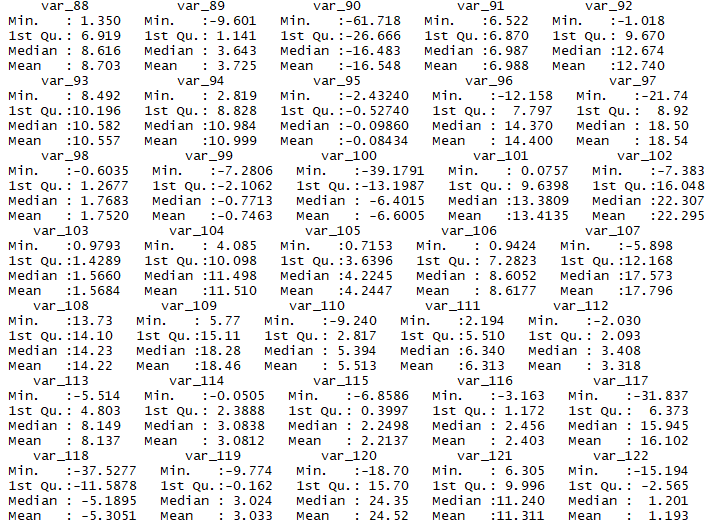
*summary*

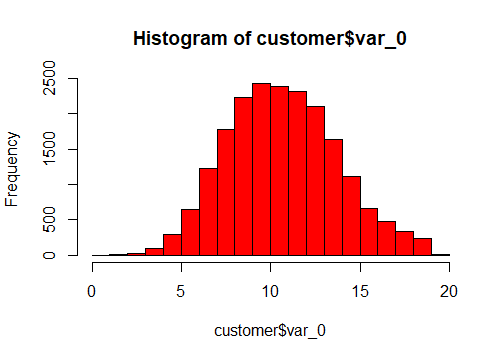
Some central tendency of data to us about the data like – mean, medium, etc. about each variable.

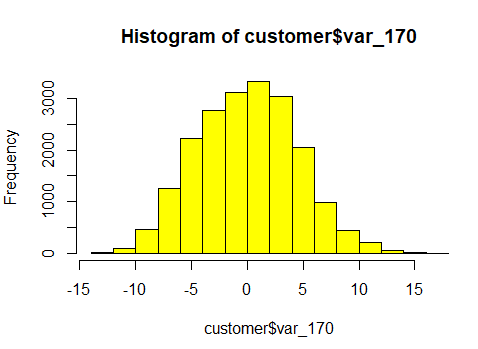
(summary of the rest variables)

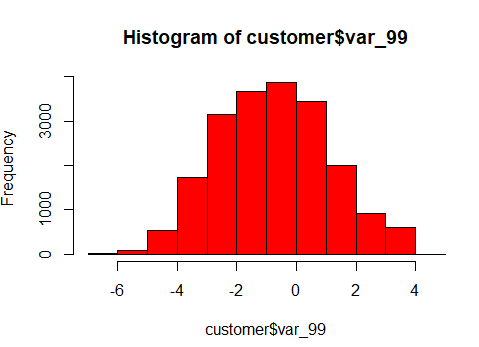


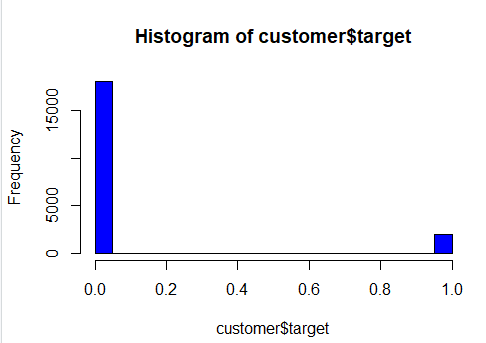




*Histogram*







*Boruta graph*

