A Project on Santander Transaction Problem

By Sapna Krishnan

11 September 2019

Contents

Chapter 1

Introduction

The data analytics lifecycle defines analytics process best practises spanning discovery to project completion. The Data Analytics Life Cycle has 6 major phases. Our project will be directed through each phase to completion.

PHASE – 1 – DISCOVERY

Important activities in this phase include framing the business problem as an analytics challenge that can be addressed in subsequent phases and formulating initial hypothesis to test and begin learning the data.

1.1 Problem Statement

In this challenge, we need to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted.

1.2 Data

Our task is to build a suitable model which will help identify if a customer will make a transaction or not irrespective of the amount. Given below is a sample of the provided data sets:

ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7
train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266
train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338
train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155
train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250
train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514

Table 1.1: Sample Data of Train Dataset (Columns: 1-10)

ID_code	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8
test_0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337
test_1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131
test_2	5.4827	-10.3581	10.1407	7.0479	10.2628	9.8052	4.8950	20.2537	1.5233
test_3	8.5374	-1.3222	12.0220	6.5749	8.8458	3.1744	4.9397	20.5660	3.3755
test_4	11.7058	-0.1327	14.1295	7.7506	9.1035	-8.5848	6.8595	10.6048	2.9890

Table 1.2: Sample Data of Test Dataset (Columns: 1-10)

Let us now try to understand the different variables of the data set.

We are provided with two anonymized datasets – Train and Test. The Train Dataset contains 20000 Observations with 202 variables. The “ID_code” variable is a String variable which uniquely identifies each observation in the Dataset. The “Target” column is binary in nature and has 2 values – ‘0’ represents Customer has not made transaction and ‘1’ represents Customer has made a transaction. The Target variable is the Dependent variable. Besides these 2 variables, there are 200 numeric feature variables which are independent variables.

The Test Dataset contains 20000 Observations with 201 variables. The “ID_code” variable is a String variable which uniquely identifies each observation in the Dataset. There are 200 numeric feature variables which are independent variables. The task is to predict the value of “Target” column in the Test Dataset.

Chapter 2

Methodology

PHASE 2 – DATA PREPARATION

The second phase of the Data Analytics Lifecycle involves data preparation, which includes the steps to explore, pre-process and condition data prior to modelling and analysis. In this phase, we must decide how to condition and transform data to get it into a format to facilitate subsequent analysis. Data visualizations help us understand the data including trends, outliers and relationships among data variables. Data preparation tends to be the most labour-intensive step in the analytics lifecycle.

2.1 Exploratory Data Analysis

Being the first stage of data preparation, in this stage, the data is analysed to understand the different variables and their data types. Studying the structure of the data set. Understanding the shape and properties of data.

- The Train data has a total of 202 variables with 200000 observations.
- The Test data has a total of 201 variables with 200000 observations.
- Checking for different data types and converting as required
- Identifying categorical and continuous variables from the data-set.
    - There are 2 Categorical Variables – ID_Code with 200000 levels and Target with 2 levels. There are 200 Continuous Variables namely var_0, var_1 and so on till var_199 in the Train data set.
    - There is 1 Categorical Variable - ID_Code with 200000 levels and 200 Continuous Variables namely var_0, var_1 and so on till var_199 in the Test data set.

2.1.1 Missing Value Analysis

Identifying the number of missing values in each variable and imputing to minimize errors.

We use the below function to analyse missing values in our data sets:

apply(train,2,function(x){sum(is.na(x))

apply(test,2,function(x){sum(is.na(x))

On running the above lines of code, it is observed that there are no missing values in both the datasets.

2.1.2 Outlier Analysis

One of the other steps of pre-processing apart from checking for normality is the presence of outliers. We visualize the outliers using boxplots. The below Outliers graph clearly depicts there are a number of outliers in multiple variables and it needs to be eliminated.

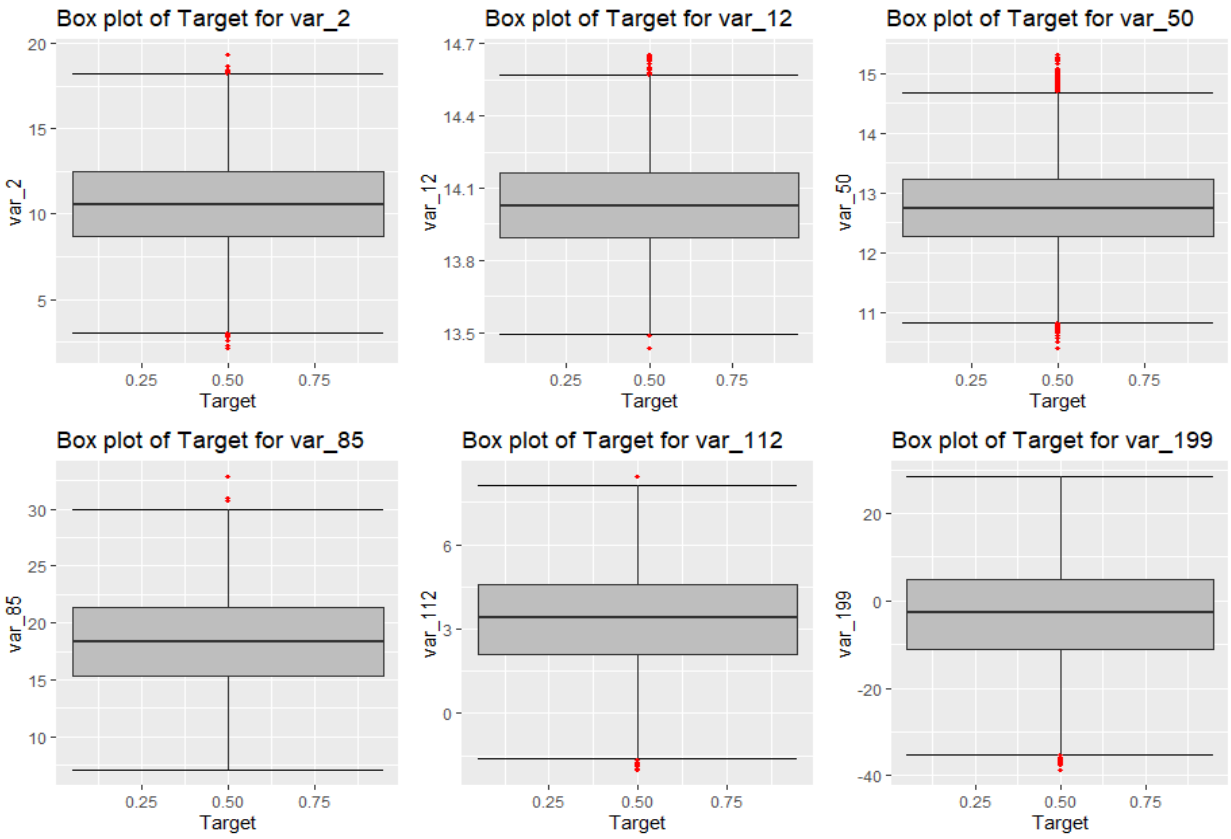

Fig 2.1: Box_Plots for Outlier Detection

We remove outliers using the Box-Plots method. We fill in the outlier values with NA and impute the NA values with mean.

After eliminating outlier values from both datasets, we are left with below observations in our datasets:

test_org	174578 obs. of 201 variables
train_org	175073 obs. of 202 variables

Table 2.1: Observation Count after Eliminating Outliers

2.1.3 Collinearity

	target	var_0	var_1	var_2	var_3	var_4	var_5
target	1.0000000000	5.063232e-02	4.754487e-02	0.0571972031	1.123699e-02	1.204530e-02	3.095772e-02
var_0	0.0506323246	1.000000e+00	3.410495e-04	0.0056808100	3.611778e-03	1.487048e-03	3.467478e-03
var_1	0.0475448668	3.410495e-04	1.000000e+00	0.0025024191	-5.913449e-05	1.733724e-03	-6.693399e-04
var_2	0.0571972031	5.680810e-03	2.502419e-03	1.0000000000	8.646717e-04	7.639243e-04	7.432512e-04
var_3	0.0112369901	3.611778e-03	-5.913449e-05	0.0008646717	1.000000e+00	1.980473e-04	3.603783e-03
var_4	0.0120452969	1.487048e-03	1.733724e-03	0.0007639243	1.980473e-04	1.000000e+00	-1.276677e-03

Table 2.2: Sample Collinearity values for Train Dataset

	var_0	var_1	var_2	var_3	var_4
var_0	1.000000e+00	4.736159e-03	2.032102e-03	2.363845e-03	-2.586652e-03
var_1	4.736159e-03	1.000000e+00	4.280416e-03	-3.105166e-04	-1.053454e-04
var_2	2.032102e-03	4.280416e-03	1.000000e+00	-5.535729e-03	3.237663e-03
var_3	2.363845e-03	-3.105166e-04	-5.535729e-03	1.000000e+00	3.954645e-05
var_4	-2.586652e-03	-1.053454e-04	3.237663e-03	3.954645e-05	1.000000e+00

Table 2.3: Sample Collinearity values for Test Dataset

It is observed that there is neither strong nor weak collinearity among variables of both test and train datasets.

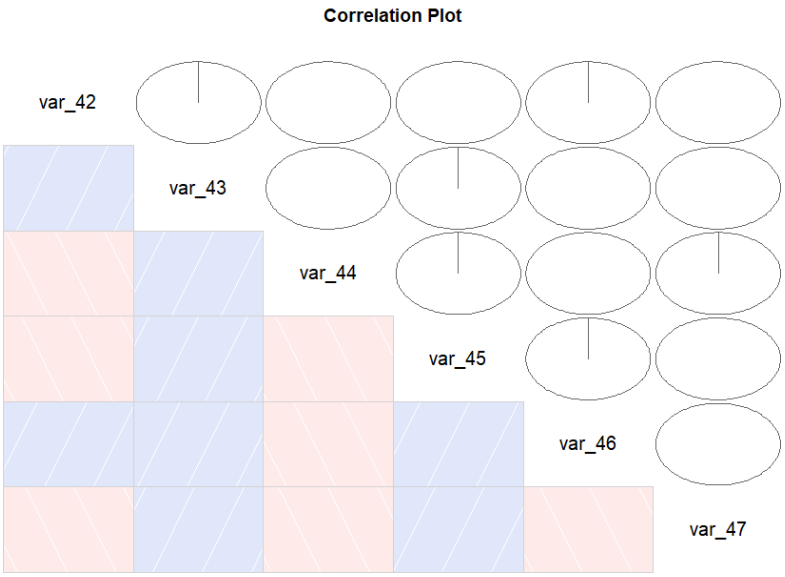

Fig 2.2: Sample Collinearity graph for Train Dataset

PHASE 3: MODEL PLANNING

2.1.4 Distribution of Target Variable in Train Dataset

0	1
157970	17103

Table 2.4: Figures of Distribution of Target variable

Table 2.4 depicts numerical figures of classification of the Target variable in the train dataset.

0	1
90.230932	9.769068

Table 2.5: Percentage of Distribution of Target variable

Table 2.5 depicts percentage value of classification of the Target variable in the train dataset

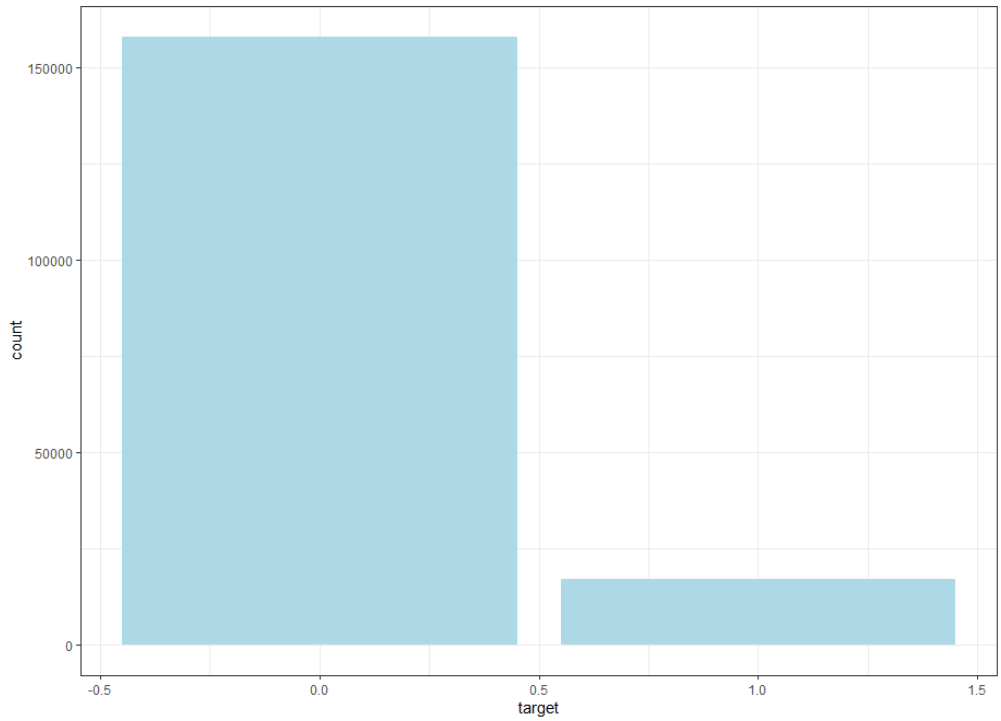

Fig 2.3: Distribution of Target Variable in Train Dataset

Fig 2.3 shows the bar plot showing the distribution of Target variable in the Train Dataset. The above figures depict that 90% customers will not make a transaction and 10% of customers will make a transaction.

2.1.4 Distribution of Continuous Variables

This phase includes learning relationships between variables and subsequently selecting key variables and the most suitable models

Continuous Variable vs. Target Variable in Train DataSet

Let us now analyse the relation of continuous variables with target variable

- It is observed that a considerable number of variables significantly have different distributions for two target variables
- It is also observed that a considerable number of variables significantly have same distributions for two target variables.

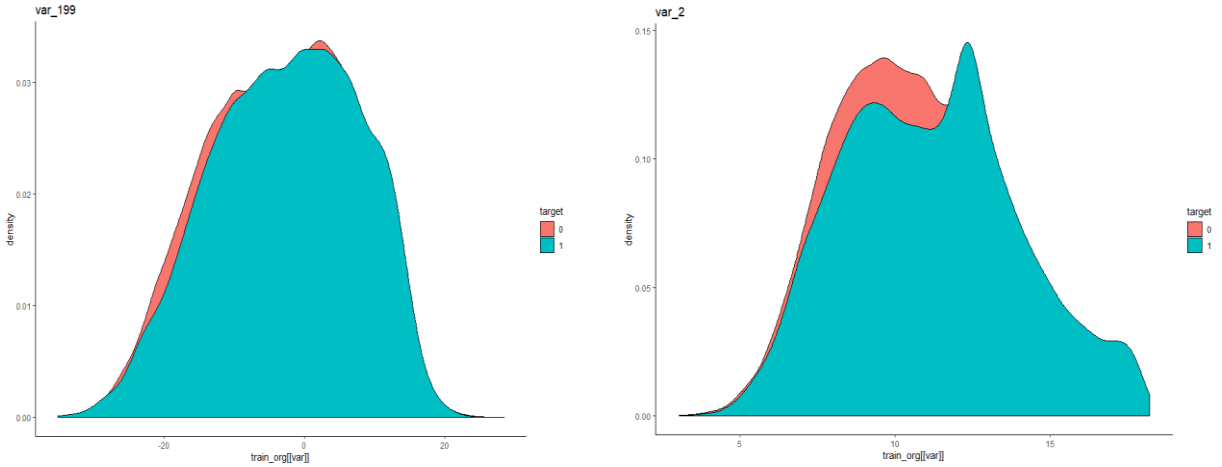

Fig 2.4: Distribution of sample variables over target variable

Distribution of Variables in Test DataSet

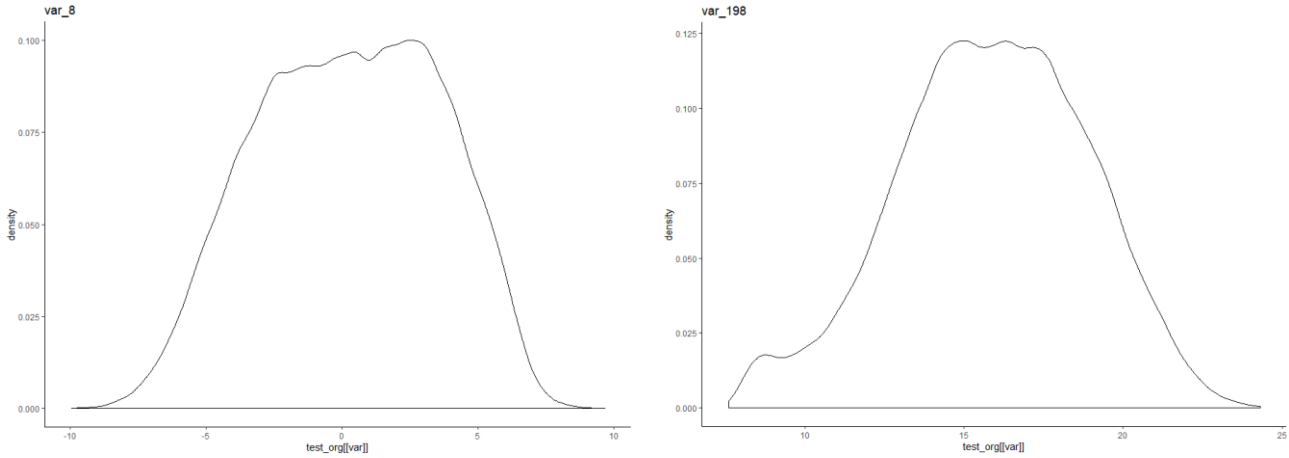

Fig 2.5: Distribution of sample Test variables

What we can infer from above graph:

- It is observed that a considerable number of variables significantly have different distributions.
- It is also observed that a considerable number of variables significantly have same distributions.

2.1.5 Distribution of Skewness in the DataSets

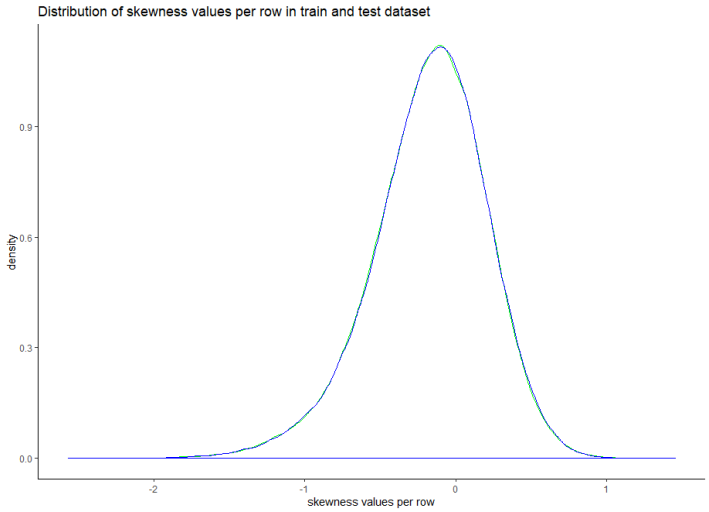

Fig 2.6: Distribution of Skewness Row-Wise

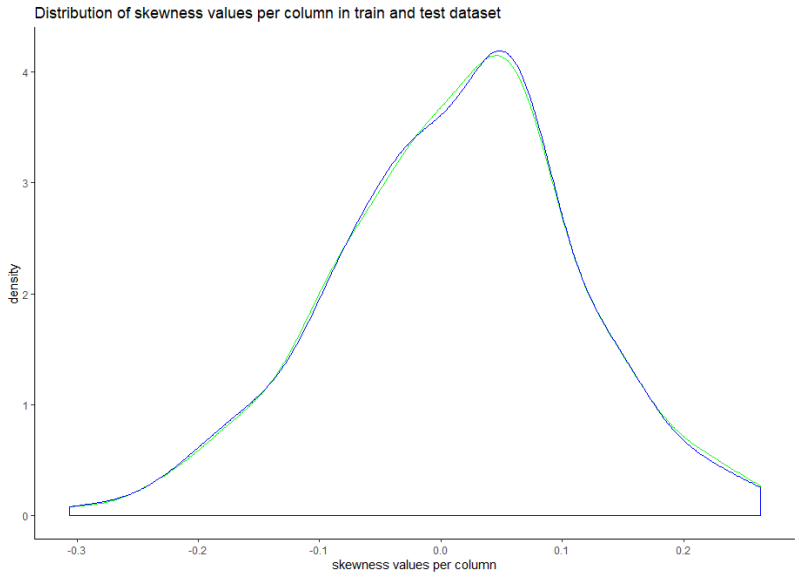

Fig 2.7: Distribution of Skewness Column-Wise

From above figure, it is observed that the data in both the data sets is almost similar with not much skewness.

2.2 Feature Selection

Before performing any type of modelling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that. Below we have used Random Forest Algorithm to perform feature selection and choose important variables.

As part of training model, we consider 75% of standardized data as training data and rest 25% as test data.

We use the below code to analyse important features in the dataset.

```
#Training the Random forest classifier to identify Important Variables
set.seed(2732)
train_data$target<-as.factor(train_data$target)

#Setting the mtry
mtry<-floor(sqrt(200))

#Setting the tunegrid
tuneGrid<-expand.grid(.mtry=mtry)

#Fitting the Random Forest
rf<-randomForest(target~.,train_data[,-c(1)],mtry=mtry,ntree=10,importance=TRUE)

#Variable importance
VarImp<-importance(rf,type=2)
VarImp
```

Fig 2.8: Code for Feature Selection

Based on MeanGiniIndex value, important variables are chosen. The higher the value of MeanGiniIndex, greater the importance.

Since we have multiple variables, I have considered the mean of MeanGiniIndex and considered variables with a value greater than mean value.

	MeanDecreaseGini
var_81	257.71519
var_139	225.43587
var_12	216.50461
var_53	212.09358
var_166	195.38054
var_110	192.45065
var_80	190.40117
var_174	184.70609
var_26	183.35406
var_198	180.35392
var_6	177.37612
var_22	173.35020
var_133	164.58162
var_109	164.09609
var_76	161.87058

Table 2.6: Snap of Selected Variables

We have considered 72 variables for further analysis.

2.3 Modeling

PHASE 4: MODEL BUILDING

Develop datasets for training, testing and production purposes along with building and executing model.

2.3.1 Model Building

In our early stages of analysis during pre-processing we have come to understand how continuous and categorical variables are related to target variable.

Since the problem is a Classification, I have used the below three classification models:

- Decision trees
- Random Forest
- Logistic Regression

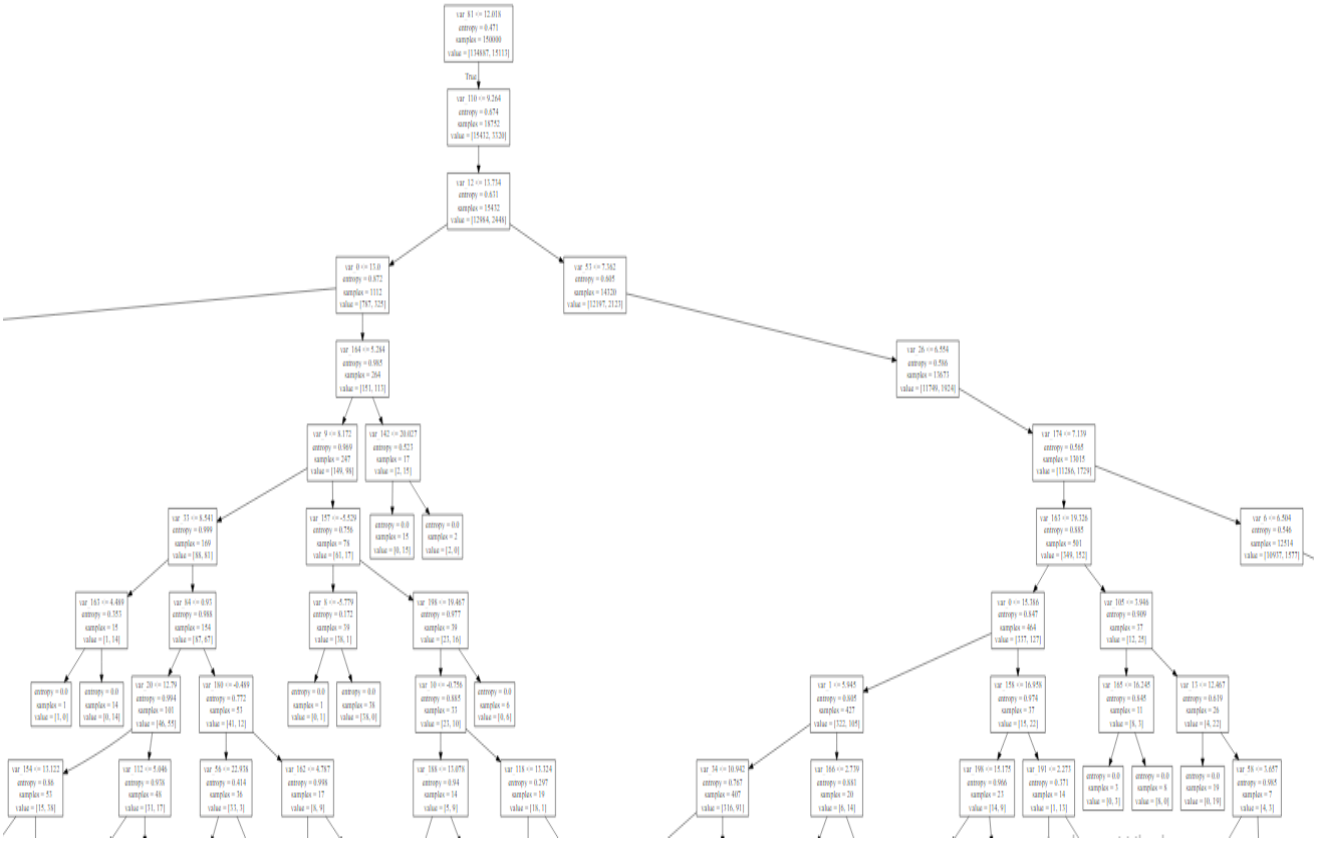

Fig 2.9: Sample snip of Decision Tree Rules

Chapter 3

Conclusion

PHASE 5 : COMMUNICATE RESULTS - Checking if results are successful or not.

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using the below metrics:

- Accuracy – How accurately model is classified?
- Precision – Proportion of actual negative cases correctly identified.
- Recall - Proportion of actual positive cases correctly identified.
- FNR – False Negative Rate

We use confusion matrix in each model to further analyse.

The below table gives values of the 3 models that I have considered for analysis:

Algorithm	Accuracy	FNR	Recall	Precision
Decision Tree	83.472%	81.344%	18.655%	90.649%
Random Forest	90.086%	99.297%	0.700%	99.984%
Logistic Regression	91.011%	73.996%	26.003%	98.543%

Table 3.1: Metrics Evaluation an Model Selection

As observed from table 3.1, Logistic Regression Algotihm has High Accuracy and High precision with Low FNR and Low Recall.

Hence, it is the best choice.

PHASE 6 : OPERATIONALIZE – Delivering final reports, briefings, code and other documents.

COMPLETE R CODE

```
#Clear R environment
rm(list=ls(all=T))

#Set Current Working Directory
setwd("C:/Users/HP/Desktop/Edwisor/Project 2")

#Load Libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",
    "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')

#Install Packages
install.packages(x)
lapply(x, require, character.only = TRUE)
rm(x)

#Read the data
train_org = read.csv("train.csv", header = T, na.strings = c(" ","", "NA"))
test_org = read.csv("test.csv", header = T, na.strings = c(" ","","NA"))

#####-------------------------EXPLORATORY DATA ANALYSIS-------------------------------#####
#Summary of Data Sets
str(train_org)
str(test_org)

#Dimensions of DataSets
dim(train_org)
dim(test_org)
```

```
#convert to factor
train_org$target<-as.factor(train_org$target)
class(train_org$target)

#####-------------------------------MISSING VALUE ANALYSIS----------------------------#####
#Finding the missing values in train data
missing_val_train<-data.frame(missing_val_train=apply(train_org,2,function(x){sum(is.na(x))}))
missing_val_train<-sum(missing_val_train)
missing_val_train

#Finding the missing values in test data
missing_val_test<-data.frame(missing_val_test=apply(test_org,2,function(x){sum(is.na(x))}))
missing_val_test<-sum(missing_val_test)
missing_val_test

#####---------------------------------OUTLIER ANALYSIS-------------------------------#####
#BoxPlots - Distribution and Outlier Check
cnames = c("var_2","var_12","var_50","var_85","var_112","var_199")
for (i in 1:6){
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "target"), data = subset(train_org))+
                  stat_boxplot(geom = "errorbar", width = 0.5) +
                  geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,outlier.size=1,
notch=FALSE) +
                  theme(legend.position="bottom")+
                  labs(y=cnames[i],x="Target")+
                  ggtitle(paste("Box plot of Target for",cnames[i])))
}
```

```
gridExtra::grid.arrange(gn1,gn2,gn3,gn4,gn5,gn6,ncol=3)
rm(list = "i")

#Remove outliers using boxplot method from TRAIN dataset
for(i in 3:202){
  print(i)
  val = train_org[,i][train_org[,i] %in% boxplot.stats(train_org[,i])$out]
  print(length(val))
  train_org = train_org[which(!train_org[,i] %in% val),]
}

#Remove outliers using boxplot method from TEST dataset
for(i in 2:201){
 print(i)
 val = test_org[,i][test_org[,i] %in% boxplot.stats(test_org[,i])$out]
 print(length(val))
 test_org = test_org[which(!test_org[,i] %in% val),]
}

#Replace all outliers in TRAIN Dataset with NA
for(i in 3:202){
 val = train_org[,i][train_org[,i] %in% boxplot.stats(train_org[,i])$out]
 print(length(val))
 train_org[,i][train_org[,i] %in% val] = NA
}

#Replace all outliers in TEST Dataset with NA
for(i in 2:201){
 val = test_org[,i][test_org[,i] %in% boxplot.stats(test_org[,i])$out]
```

```
  print(length(val))
  test_org[,i][test_org[,i] %in% val] = NA
}

#Impute NA with mean
for(i in 3:202){
  train_org[is.na(train_org[,i]),i] = mean(train_org[,i], na.rm = T)
}

for(i in 2:201){
  test_org[is.na(test_org[,i]),i] = mean(test_org[,i], na.rm = T)
}

######################Distribution of Target Variable#############################
table(train_org$target)

#Percenatge counts of target classes
table(train_org$target)/length(train_org$target)*100

#Bar plot for count of target classes
plot1<-ggplot(train_org,aes(target))+theme_bw()+geom_bar(stat='count',fill='lightblue')
grid.arrange(plot1, ncol=1)

##############################Feature Selection##############################
## Correlation Plot
corrgram(train_org[,45:50], order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
```

```
#Correlations in train data
#convert factor to int
train_org$target<-as.numeric(train_org$target)
train_correlations<-cor(train_org[,c(2:202)])
train_correlations

#Correlations in test data
test_correlations<-cor(test_org[,c(2:201)])
test_correlations

rm(list = "i","val")

#Distribution of Train attributes
train_org$target<-as.factor(train_org$target)
for (var in names(train_org)[c(3:202)]){
 target<-train_org$target
 plot<-ggplot(train_org, aes(x=train_org[[var]], fill=target)) +
  geom_density(kernel='gaussian') + ggtitle(var)+theme_classic()
 print(plot)
}

#Distribution of Test attributes
for (var in names(test_org)[c(3:201)]){
 plot<-ggplot(test_org, aes(x=test_org[[var]])) +
  geom_density(kernel='gaussian') + ggtitle(var)+theme_classic()
 print(plot)
}

rm(list = "i")
```

```
#Applying the function to find skewness values per row in train and test data.
train_skew<-apply(train_org[,-c(1,2)],MARGIN=1,FUN=skewness)
test_skew<-apply(test_org[,-c(1)],MARGIN=1,FUN=skewness)

#Distribution of skewness values per row in train and test data
ggplot()+geom_density(aes(x=train_skew),kernel='gaussian',show.legend=TRUE,color='green')+t
heme_classic()+
geom_density(aes(x=test_skew),kernel='gaussian',show.legend=TRUE,color='blue')+
labs(x='skewness values per row',title="Distribution of skewness values per row in train and test
dataset")

#Applying the function to find skewness values per column in train and test data.
train_skew<-apply(train_org[,-c(1,2)],MARGIN=2,FUN=skewness)
test_skew<-apply(test_org[,-c(1)],MARGIN=2,FUN=skewness)

#Distribution of skewness values per column in train and test data
ggplot()+geom_density(aes(x=train_skew),kernel='gaussian',show.legend=TRUE,color='green')+t
heme_classic()+
geom_density(aes(x=test_skew),kernel='gaussian',show.legend=TRUE,color='blue')+
labs(x='skewness values per column',title="Distribution of skewness values per column in train
and test dataset")

rm(list="test_skew","train_skew")

###############----------------------------Variable Importance-------------------------------##########
#Split the training data using simple random sampling
train_index<-sample(1:nrow(train_org),0.75*nrow(train_org))
#train data
```

```
train_data<-train_org[train_index,]
#validation data
valid_data<-train_org[-train_index,]
#dimension of train and validation data
dim(train_data)
dim(valid_data)

#Training the Random forest classifier to identify Important Variables
set.seed(2732)
train_data$target<-as.factor(train_data$target)

#Setting the mtry
mtry<-floor(sqrt(200))

#Setting the tunegrid
tuneGrid<-expand.grid(.mtry=mtry)

#Fitting the Random Forest
rf<-randomForest(target~.,train_data[,-c(1)],mtry=mtry,ntree=10,importance=TRUE)

#Variable importance
VarImp<-importance(rf,type=2)
VarImp

data = cbind(rownames(VarImp), VarImp)
rownames(data) = NULL
colnames(data)[1] = "Var"

#Considering variables having High Imp in Prediction
```

```
data = data[VarImp>mean(VarImp),]
train_org_2 =
train_org[,c("target","var_0","var_1","var_2","var_5","var_6","var_9","var_12","var_13","var_1
8","var_21","var_22","var_24","var_26","var_33","var_34","var_40","var_44","var_49","var_51
","var_53","var_56","var_75","var_76","var_78","var_80","var_81","var_83","var_86","var_91",
"var_92","var_93","var_94","var_95","var_99","var_106","var_108","var_109","var_110","var_
115","var_119","var_121","var_122","var_123","var_133","var_139","var_141","var_145","var_
146","var_147","var_148","var_150","var_154","var_155","var_162","var_164","var_165","var_
166","var_169","var_170","var_172","var_174","var_177","var_179","var_180","var_184","var_
188","var_190","var_191","var_194","var_197","var_198")]

rm(list = "i")
write.csv(train_org_2, "train_org_final.csv", row.names = F)

#Standardisation
for(i in 2:72){
  print(i)
  train_org_2[,i] = (train_org_2[,i] - mean(train_org_2[,i]))/sd(train_org_2[,i])
}

#####-------------------------MODEL DEVELOPMENT-----------------------------------#########
#Divide data into train and test using stratified sampling method
set.seed(2375)
train.index = createDataPartition(train_org_2$target, p = .75, list = FALSE)
train = train_org_2[ train.index,]
train_test  = train_org_2[-train.index,]

##Decision tree for classification
#Develop Model on training data
```

```
train$target = as.factor(train$target)
C50_model = C5.0(target ~., train, trials = 50, rules = TRUE)

#Summary of DT model
summary(C50_model)

#write rules into disk
write(capture.output(summary(C50_model)), "c50Rules.txt")

#Lets predict for test cases
C50_Predictions = predict(C50_model, train_test, type = "class")

##Evaluate the performance of classification model
ConfMatrix_C50 = table(train_test$target, C50_Predictions)
confusionMatrix(ConfMatrix_C50)

#False Negative rate
FNR = FN/FN+TP

#Recall
RC = TP/TP+FN

#Precision
P = (TN*100)/TN+FP
#Accuracy: 83.472%
#FNR: 81.344%
#RC: 18.655%
#Precision : 90.649%
```

```
###Random Forest
RF_model = randomForest(target ~ ., train, importance = TRUE, ntree = 500)

#Presdict test data using random forest model
RF_Predictions = predict(RF_model, train_test)

##Evaluate the performance of classification model
ConfMatrix_RF = table(test$responded, RF_Predictions)
confusionMatrix(ConfMatrix_RF)

#False Negative rate
FNR = FN/FN+TP
#Recall
RC = TP/TP+FN
#Precision
P = (TN*100)/TN+FP

#Accuracy: 90.086%
#FNR: 99.297%
#RC: 0.70%
#Precision : 99.984%

#Logistic Regression
logit_model = glm(target ~ ., data = train, family = "binomial")

#summary of the model
summary(logit_model)

#predict using logistic regression
```

```
logit_Predictions = predict(logit_model, newdata = train_test, type = "response")

#convert prob
logit_Predictions = ifelse(logit_Predictions > 0.5, 1, 0)

##Evaluate the performance of classification model
ConfMatrix_RF = table(train_test$target, logit_Predictions)
confusionMatrix(ConfMatrix_RF)
#False Negative rate
FNR = FN/FN+TP
#Recall
RC = TP/TP+FN
#Precision
P = (TN*100)/TN+FP

#Accuracy: 91.011%
#FNR: 73.996%
#RC: 26.003%
#Precision : 98.543%
```

References

Data Science and Big Data Analytics 2017, Wiley , EMC EDUCATION Services

THANK YOU