Telecom Churn

Amit Maity

August 10, 2018

Business Problem

The telecom industry continues to face growing pricing pressure worldwide. While regional differences apply, wireless penetration is reaching a saturation point across multiple markets. In addition, the longstanding ability to differentiate products and services based on handset selection and network quality is disappearing, and product life cycles are shortening. Simultaneously, wire line businesses are facing increasing competition from cable operators and a rising risk of disruption from OTT players. All of these powerful trends are forcing telecom companies to respond through more competitive offers, bundles, and price cuts.

Given these challenging industry dynamics, managing the customer base to reduce churn should be among any senior telecom executives highest priorities. And our work with telecom companies around the world reveals that those companies that implement a comprehensive, analytics-based approach to base management can reduce their churn by as much as 15%.

Exploratory Data Analysis

Columns	**Description**
state	Which state in the US the Accounts belongs to
Account Length	Total Length of the account
area code	Code of the area
phone number	Phone numbers
international plan	Customer opted for an international plan
voice mail plan	Customer opted for an voice mail plan
number vmail messages	Number of voice messages
total day calls	Total daily calls
total day charge	Total daily charge
total eve calls	Total Eve calls
total eve charge	Total eve charge
total night calls	Total night calls
total night charge	Total night charge
total intl calls	Total intl calls
total intl charge	Total intl charge
customer service calls	Customer service calls
churn	Churn (Yes/No)

```
library(dplyr)
library(corrplot)
library(car)
library(randomForest)
library(caret)
library(pROC)
library(e1071)
library(lattice)
library(ggplot2)
library(InformationValue)
library(rpart)
library(rpart.plot)
telecom_train <- read.csv("E:/PGD Data
Science/PROJECT/Train_tele.csv")
telecom_test <- read.csv("E:/PGD Data Science/PROJECT/Test_tele.csv")
colnames(telecom_train)
##  [1] "state"                  "account.length"
##  [3] "area.code"              "phone.number"
##  [5] "international.plan"     "voice.mail.plan"
##  [7] "number.vmail.messages"  "total.day.calls"
##  [9] "total.day.charge"       "total.eve.calls"
## [11] "total.eve.charge"       "total.night.calls"
## [13] "total.night.charge"     "total.intl.calls"
## [15] "total.intl.charge"      "customer.service.calls"
## [17] "churn"
head(telecom_train)
##   state account.length area.code phone.number international.plan
## 1    KS            128       415     382-4657                 no
## 2    OH            107       415     371-7191                 no
## 3    NJ            137       415     358-1921                 no
## 4    OH             84       408     375-9999                yes
## 5    OK             75       415     330-6626                yes
## 6    AL            118       510     391-8027                yes
##   voice.mail.plan number.vmail.messages total.day.calls
total.day.charge
## 1             yes                    25             110
45.07
## 2             yes                    26             123
27.47
## 3              no                     0             114
41.38
## 4              no                     0              71
50.90
## 5              no                     0             113
28.34
## 6              no                     0              98
37.98
##   total.eve.calls total.eve.charge total.night.calls
```

```
total.night.charge
## 1             99           16.78                91
11.01
## 2            103           16.62               103
11.45
## 3            110           10.30               104
7.32
## 4             88            5.26                89
8.86
## 5            122           12.61               121
8.41
## 6            101           18.75               118
9.18
##   total.intl.calls total.intl.charge customer.service.calls churn
## 1                3              2.70                      1 FALSE
## 2                3              3.70                      1 FALSE
## 3                5              3.29                      0 FALSE
## 4                7              1.78                      2 FALSE
## 5                3              2.73                      3 FALSE
## 6                6              1.70                      0 FALSE
```

Dummy variable for categorical value

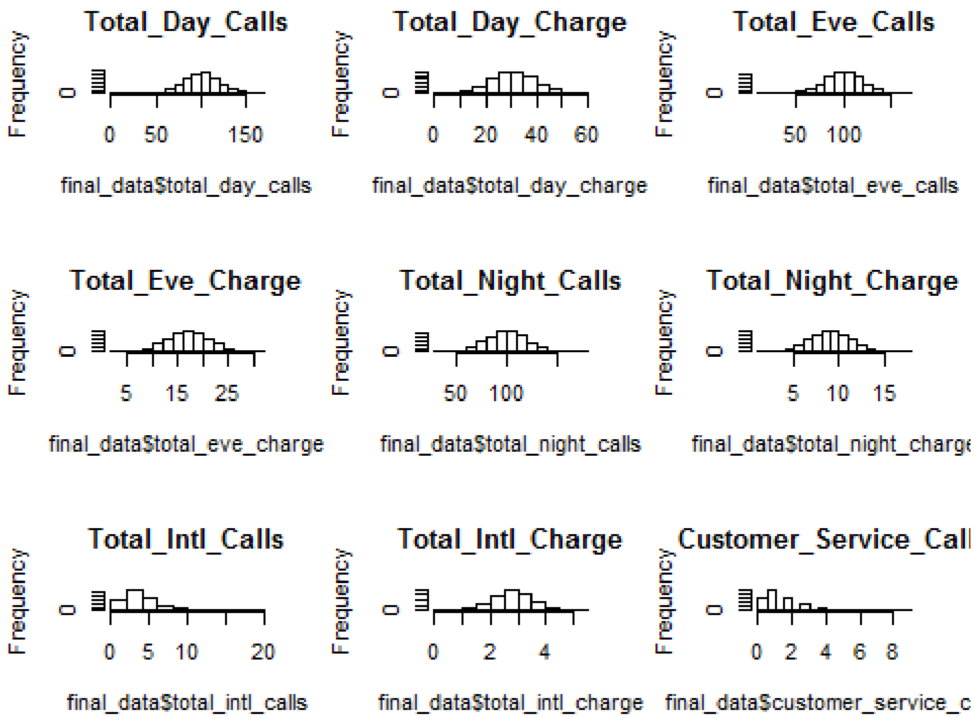
```
telecom_train$international_plan <-
ifelse(telecom_train$international.plan=="yes",1,0)
telecom_train$vmail_plan <-
ifelse(telecom_train$voice.mail.plan=="yes",1,0)
```

State,Account.length,Area.code, Phone.number are not relevant to our analysis. Will remove these columns and stored in a new variable called final_data.

```
final_data = telecom_train[-c(1,2,3,4,5,6)]
colnames(final_data) <-
c("number_vmail_messages","total_day_calls","total_day_charge","total_
eve_calls","total_eve_charge", "total_night_calls",
"total_night_charge", "total_intl_calls","total_intl_charge",
"customer_service_calls", "churn", "international_plan", "vmail_plan")
filter(final_data,final_data$vmail_plan ==
0,final_data$number_vmail_messages > 0)
##  [1] number_vmail_messages  total_day_calls        total_day_charge

##  [4] total_eve_calls        total_eve_charge
total_night_calls
##  [7] total_night_charge     total_intl_calls
total_intl_charge
## [10] customer_service_calls churn
international_plan
## [13] vmail_plan
## <0 rows> (or 0-length row.names)
```

One voice messages value is missing. We found a close relation between vmail_plan and number_vmail_messages. It is intuitive that when customer is opting voice mail plan then only ‘number_vmail_messages’ field has some non-zero values.So here we will replace zero to the missing vmail messages value

```
filter(select(final_data,-
churn),is.na(final_data$number_vmail_messages ))
##   number_vmail_messages total_day_calls total_day_charge
total_eve_calls
## 1                    NA              72             36.4
104
##   total_eve_charge total_night_calls total_night_charge
total_intl_calls
## 1            13.97               113               7.99
3
##   total_intl_charge customer_service_calls international_plan
vmail_plan
## 1              2.21                      2                  0
0
final_data$number_vmail_messages[is.na(final_data$number_vmail_message
s)] = 0
summary(final_data$number_vmail_messages)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   8.125  20.000  51.000
par(mfrow=c(3,3))
hist(final_data$total_day_calls, main="Total_Day_Calls")
hist(final_data$total_day_charge, main="Total_Day_Charge")
hist(final_data$total_eve_calls, main="Total_Eve_Calls")
hist(final_data$total_eve_charge, main="Total_Eve_Charge")
hist(final_data$total_night_calls, main="Total_Night_Calls")
hist(final_data$total_night_charge, main="Total_Night_Charge")
hist(final_data$total_intl_calls, main="Total_Intl_Calls")
hist(final_data$total_intl_charge, main="Total_Intl_Charge")
hist(final_data$customer_service_calls, main="Customer_Service_Calls")
```



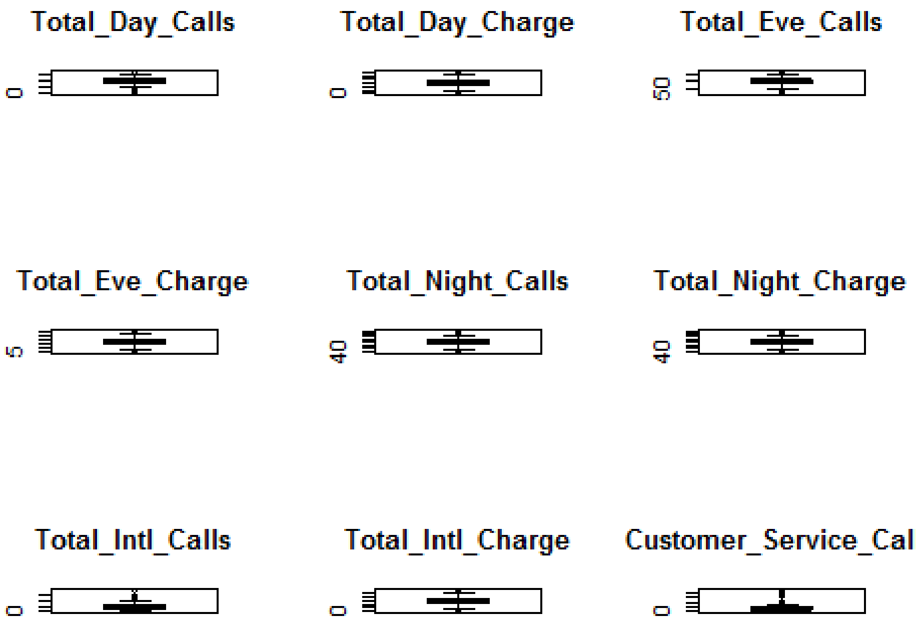
From individual column's histogram, we observed total_day_calls, total_eve_calls, total_night_calls,total_intl_calls,customer_service_calls are normally distributed and discrete in nature. So we will do **median imputation**. And total_day_charge, total_eve_charge, total_night_charge, total_intl_charge are normally distributed and continuous in nature. So we will do **mean imputation**.

```
final_data$total_day_calls[is.na(final_data$total_day_calls)] = 101
final_data$total_day_charge[is.na(final_data$total_day_charge)] =
30.60
final_data$total_eve_calls[is.na(final_data$total_eve_calls)] = 100
final_data$total_eve_charge[is.na(final_data$total_eve_charge)] =
17.10
final_data$total_night_calls[is.na(final_data$total_night_calls)] =
100
final_data$total_night_charge[is.na(final_data$total_night_charge)] =
9
final_data$total_intl_calls[is.na(final_data$total_intl_calls)] = 4
final_data$total_intl_charge[is.na(final_data$total_intl_charge)] =
2.77
final_data$customer_service_calls[is.na(final_data$customer_service_ca
lls)] = 1
```

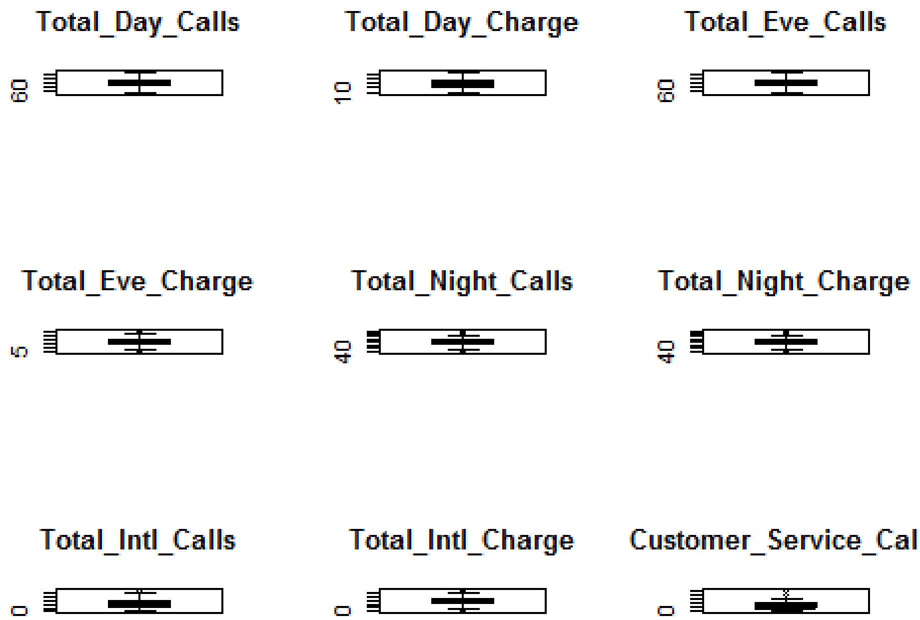
Univariate Analysis

```
par(mfrow=c(3,3))
boxplot(final_data$total_day_calls, main="Total_Day_Calls")
boxplot(final_data$total_day_charge, main="Total_Day_Charge")
```

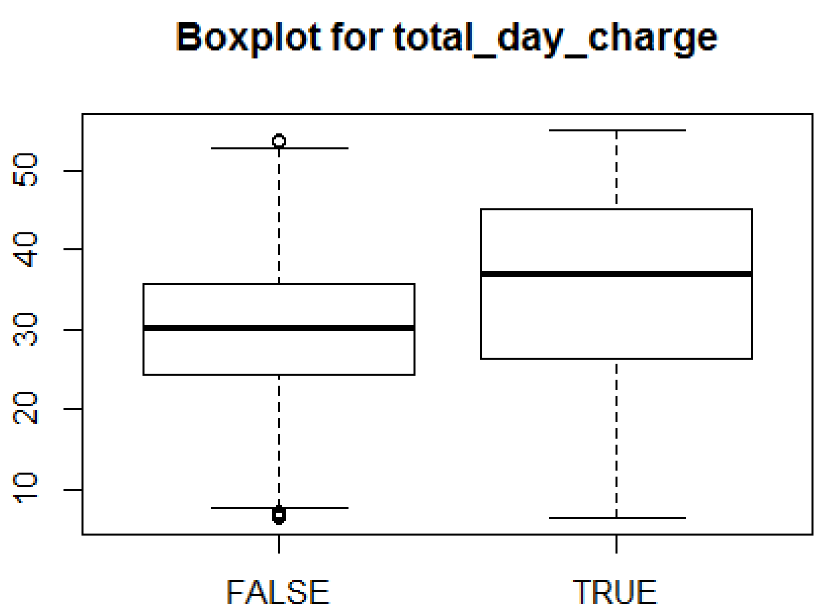
```
boxplot(final_data$total_eve_calls, main="Total_Eve_Calls")
boxplot(final_data$total_eve_charge, main="Total_Eve_Charge")
boxplot(final_data$total_night_calls, main="Total_Night_Calls")
boxplot(final_data$total_night_calls, main="Total_Night_Charge")
boxplot(final_data$total_intl_calls, main="Total_Intl_Calls")
boxplot(final_data$total_intl_charge, main="Total_Intl_Charge")
boxplot(final_data$customer_service_calls,
main="Customer_Service_Calls")
```



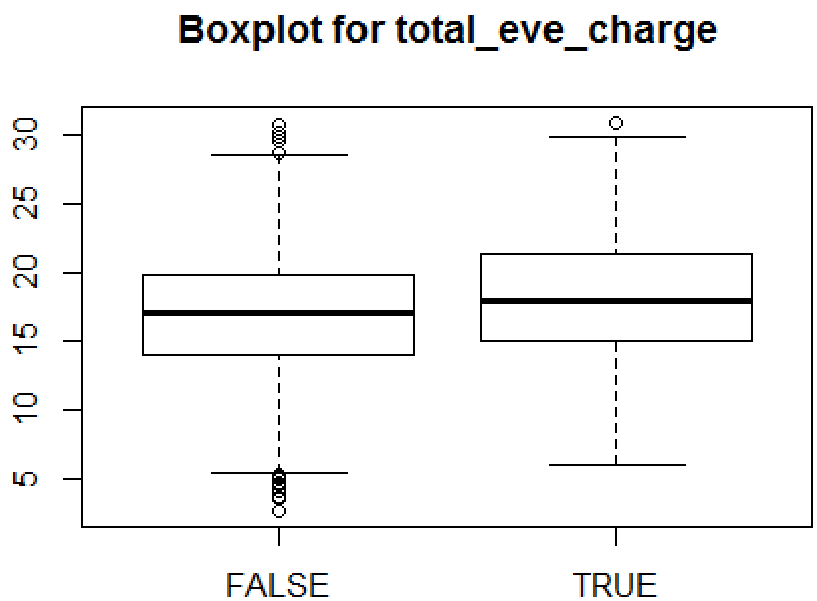
Based on the *boxplot* we are capping outliers at 2% & above 98%. After removal of outliers:



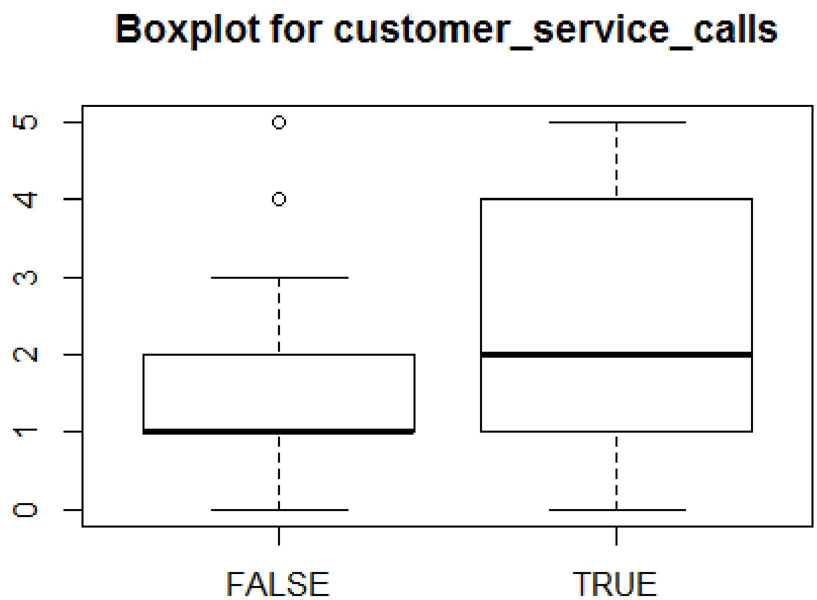

```
boxplot(final_data$total_day_charge ~ final_data$churn, main="Boxplot
for total_day_charge")
```



```r
boxplot(final_data$total_eve_charge ~ final_data$churn, main="Boxplot for total_eve_charge")
```




```r
boxplot(final_data$customer_service_calls ~ final_data$churn, main="Boxplot for customer_service_calls")
```

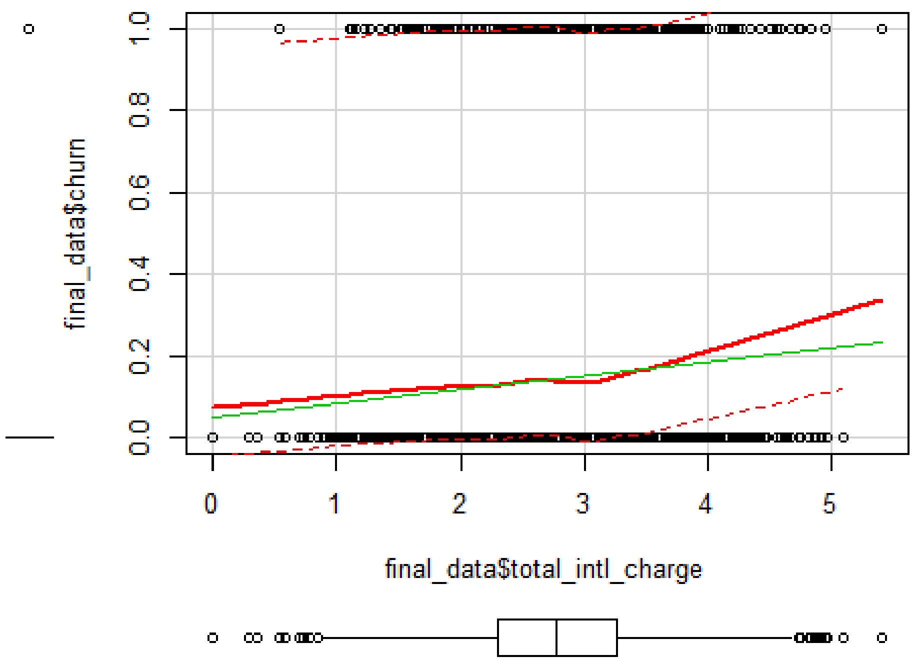


Its intuitive that Customer with high day charge and evening charge have left and obvious that customer with less satisfied have more customer_service_calls and chunred.

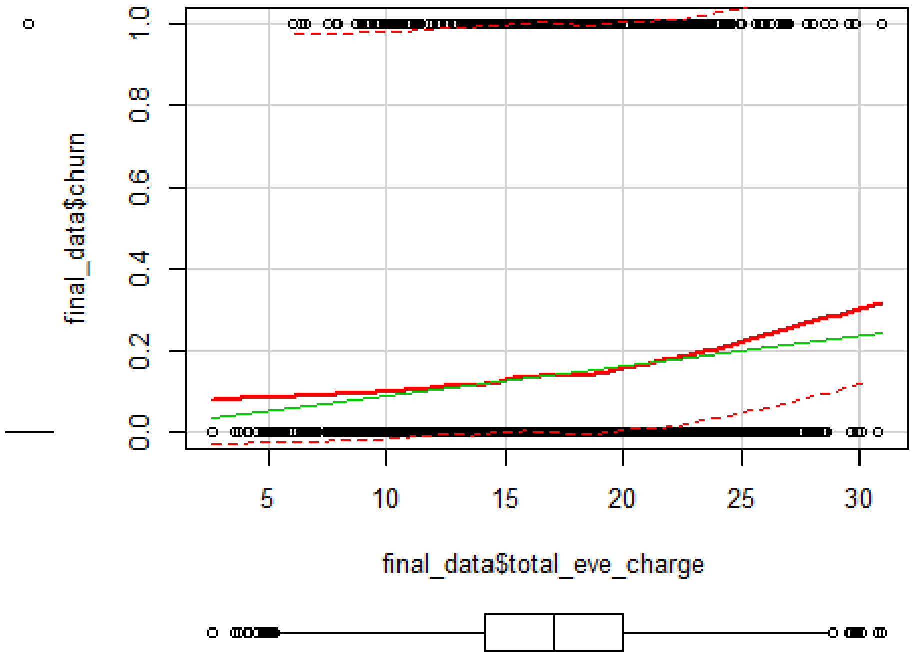
Bi-variant Analysis

Scatter Plot

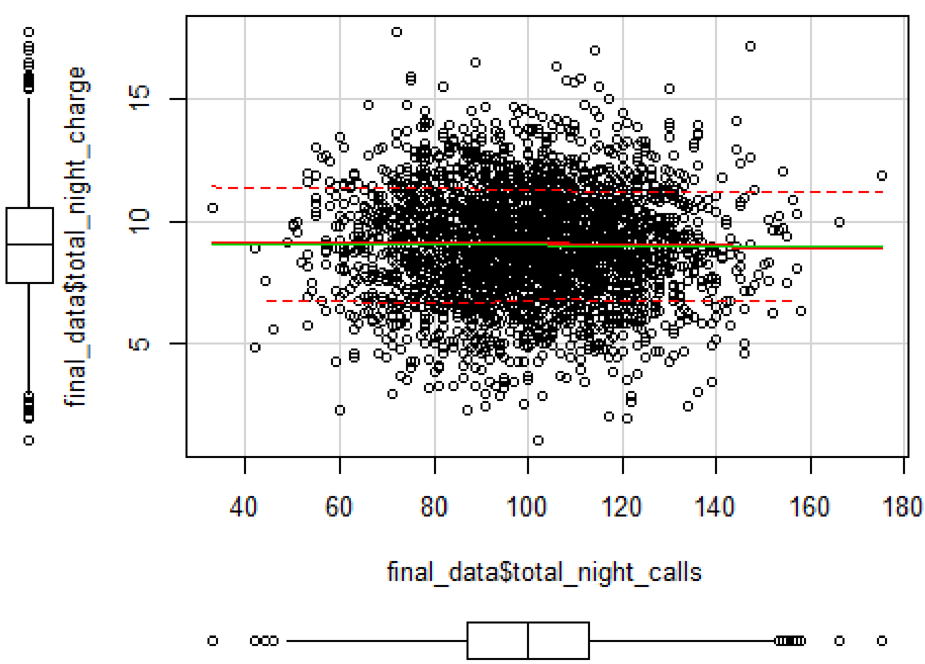
```
scatterplot(final_data$total_intl_charge,final_data$churn)
```




```
scatterplot(final_data$total_eve_charge,final_data$churn)
```



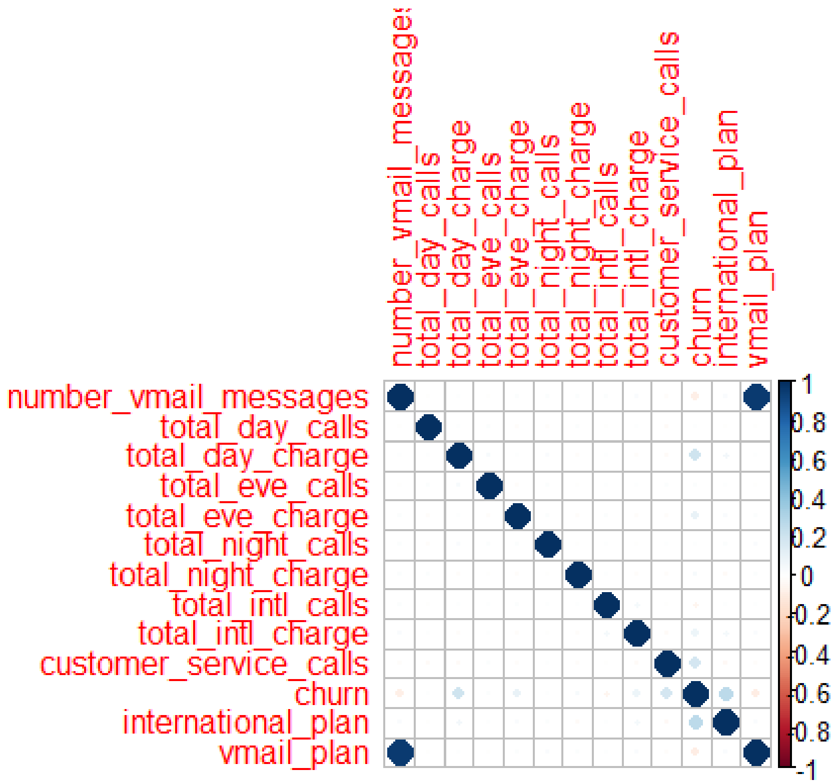

```
scatterplot(final_data$total_night_calls,final_data$total_night_charge
)
```



From above scatter plot, we can conclude with incease in total_intl_charge *(after 3)*, total_eve_charge *(after 20)* there is increase in churn.

Correlation Plot

```
corrplot(cor(final_data[,1:13]), method="circle")
```



Let divide data into Test and Train dataset

```
set.seed(222)
t=sample(1:nrow(final_data),0.7*nrow(final_data))
t_train=final_data[t,]
t_test=final_data[-t,]
```

Multi-collinearity Check

```
library(car)
mod<- lm(churn ~ ., data=t_train)
t = vif(mod)
sort(t, decreasing = T)
##  number_vmail_messages             vmail_plan
international_plan
##              12.770730              12.758197
1.015473
##       total_day_charge      total_night_calls
total_intl_charge
##               1.008680               1.007850
1.006986
##       total_intl_calls customer_service_calls
```

```
total_night_charge
##                 1.005332               1.004902
1.003513
##          total_eve_calls        total_day_calls
total_eve_charge
##                 1.002289               1.002268
1.002139
```

In above observations, number_vmail_messages & vmail_plan are highly correlated. We could choose any one of them.

Model Selection

Telecom churn is a classification problem and *dependent variable* is **categorical/binomial** in nature. The dependent variable is either **YES** or **NO**, so could solve it by different supervised learning algoritms. Here we will use **Logistic Regression**, **Decision Tree**, **Random Forest** and **Naive Bayes** techniques to build a model and will findout the best one.

Logistic Regression

```
mod1 <- glm(as.factor(churn) ~ ., family="binomial", data=t_train)
summary(mod1)
##
## Call:
## glm(formula = as.factor(churn) ~ ., family = "binomial", data =
t_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9834  -0.5067  -0.3413  -0.2028   2.8757
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -8.6357370  0.9409987  -9.177  < 2e-16 ***
## number_vmail_messages   0.0065188  0.0227113   0.287 0.774092
## total_day_calls         0.0055018  0.0036763   1.497 0.134508
## total_day_charge        0.0792345  0.0085507   9.266  < 2e-16 ***
## total_eve_calls         0.0007766  0.0036720   0.212 0.832496
## total_eve_charge        0.0750328  0.0174162   4.308 1.65e-05 ***
## total_night_calls      -0.0030186  0.0037439  -0.806 0.420096
## total_night_charge      0.0854415  0.0322200   2.652 0.008006 **
## total_intl_calls       -0.0899237  0.0339933  -2.645 0.008161 **
## total_intl_charge       0.3807265  0.1007615   3.778 0.000158 ***
## customer_service_calls  0.5326522  0.0563026   9.461  < 2e-16 ***
## international_plan      2.4494021  0.1977808  12.384  < 2e-16 ***
## vmail_plan             -1.0162302  0.7099642  -1.431 0.152321
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 1632.5  on 1993  degrees of freedom
## Residual deviance: 1265.6  on 1981  degrees of freedom
## AIC: 1291.6
##
## Number of Fisher Scoring iterations: 6
```

Instead of removing all these variables one by one, we use the step function, which automatically calculated the best equation

```
stpmod = step(mod1, direction = "both")
## Start:  AIC=1291.58
## as.factor(churn) ~ number_vmail_messages + total_day_calls +
##     total_day_charge + total_eve_calls + total_eve_charge +
total_night_calls +
##     total_night_charge + total_intl_calls + total_intl_charge +
##     customer_service_calls + international_plan + vmail_plan
##
##                          Df Deviance    AIC
## - total_eve_calls         1   1265.6 1289.6
## - number_vmail_messages   1   1265.7 1289.7
## - total_night_calls       1   1266.2 1290.2
## <none>                        1265.6 1291.6
## - vmail_plan              1   1267.7 1291.7
## - total_day_calls         1   1267.8 1291.8
## - total_night_charge      1   1272.7 1296.7
## - total_intl_calls        1   1272.8 1296.8
## - total_intl_charge       1   1280.2 1304.2
## - total_eve_charge        1   1284.6 1308.6
## - customer_service_calls  1   1358.1 1382.1
## - total_day_charge        1   1361.0 1385.0
## - international_plan      1   1419.5 1443.5
##
## Step:  AIC=1289.62
## as.factor(churn) ~ number_vmail_messages + total_day_calls +
##     total_day_charge + total_eve_charge + total_night_calls +
##     total_night_charge + total_intl_calls + total_intl_charge +
##     customer_service_calls + international_plan + vmail_plan
##
##                          Df Deviance    AIC
## - number_vmail_messages   1   1265.7 1287.7
## - total_night_calls       1   1266.3 1288.3
## <none>                        1265.6 1289.6
## - vmail_plan              1   1267.7 1289.7
## - total_day_calls         1   1267.9 1289.9
## + total_eve_calls         1   1265.6 1291.6
## - total_night_charge      1   1272.7 1294.7
## - total_intl_calls        1   1272.9 1294.9
## - total_intl_charge       1   1280.2 1302.2
## - total_eve_charge        1   1284.6 1306.6
## - customer_service_calls  1   1358.1 1380.1
```

```
## - total_day_charge        1   1361.2 1383.2
## - international_plan      1   1419.6 1441.6
##
## Step:  AIC=1287.71
## as.factor(churn) ~ total_day_calls + total_day_charge +
total_eve_charge +
##     total_night_calls + total_night_charge + total_intl_calls +
##     total_intl_charge + customer_service_calls + international_plan
+
##     vmail_plan
##
##                          Df Deviance    AIC
## - total_night_calls       1   1266.3 1286.3
## <none>                         1265.7 1287.7
## - total_day_calls         1   1267.9 1287.9
## + number_vmail_messages   1   1265.6 1289.6
## + total_eve_calls         1   1265.7 1289.7
## - total_night_charge      1   1272.8 1292.8
## - total_intl_calls        1   1272.9 1292.9
## - total_intl_charge       1   1280.3 1300.3
## - total_eve_charge        1   1284.7 1304.7
## - vmail_plan              1   1287.5 1307.5
## - customer_service_calls  1   1358.1 1378.1
## - total_day_charge        1   1361.3 1381.3
## - international_plan      1   1420.3 1440.3
##
## Step:  AIC=1286.35
## as.factor(churn) ~ total_day_calls + total_day_charge +
total_eve_charge +
##     total_night_charge + total_intl_calls + total_intl_charge +
##     customer_service_calls + international_plan + vmail_plan
##
##                          Df Deviance    AIC
## <none>                         1266.3 1286.3
## - total_day_calls         1   1268.6 1286.6
## + total_night_calls       1   1265.7 1287.7
## + number_vmail_messages   1   1266.3 1288.3
## + total_eve_calls         1   1266.3 1288.3
## - total_night_charge      1   1273.5 1291.5
## - total_intl_calls        1   1273.8 1291.8
## - total_intl_charge       1   1281.0 1299.0
## - total_eve_charge        1   1285.3 1303.3
## - vmail_plan              1   1288.3 1306.3
## - customer_service_calls  1   1358.7 1376.7
## - total_day_charge        1   1361.4 1379.4
## - international_plan      1   1420.3 1438.3
formula(stpmod)
## as.factor(churn) ~ total_day_calls + total_day_charge +
total_eve_charge +
```

```
##     total_night_charge + total_intl_calls + total_intl_charge +
##     customer_service_calls + international_plan + vmail_plan
summary(stpmod)
##
## Call:
## glm(formula = as.factor(churn) ~ total_day_calls + total_day_charge
+
##     total_eve_charge + total_night_charge + total_intl_calls +
##     total_intl_charge + customer_service_calls + international_plan
+
##     vmail_plan, family = "binomial", data = t_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0003  -0.5064  -0.3400  -0.2060   2.8689
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -8.847658   0.798580 -11.079  < 2e-16 ***
## total_day_calls         0.005486   0.003676   1.493 0.135566
## total_day_charge        0.079102   0.008555   9.246  < 2e-16 ***
## total_eve_charge        0.074880   0.017395   4.305 1.67e-05 ***
## total_night_charge      0.086038   0.032176   2.674 0.007495 **
## total_intl_calls       -0.090863   0.033915  -2.679 0.007380 **
## total_intl_charge       0.380360   0.100656   3.779 0.000158 ***
## customer_service_calls  0.532022   0.056283   9.453  < 2e-16 ***
## international_plan      2.442143   0.197008  12.396  < 2e-16 ***
## vmail_plan             -0.822527   0.184869  -4.449 8.62e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1632.5  on 1993  degrees of freedom
## Residual deviance: 1266.3  on 1984  degrees of freedom
## AIC: 1286.3
##
## Number of Fisher Scoring iterations: 6
mod2 <- glm(formula = as.factor(churn) ~ total_day_calls +
total_day_charge +
              total_eve_charge + total_night_charge + total_intl_calls
+
              total_intl_charge + customer_service_calls +
international_plan +
              vmail_plan, family = "binomial", data = t_train)
summary(mod2)
##
## Call:
## glm(formula = as.factor(churn) ~ total_day_calls + total_day_charge
+
```

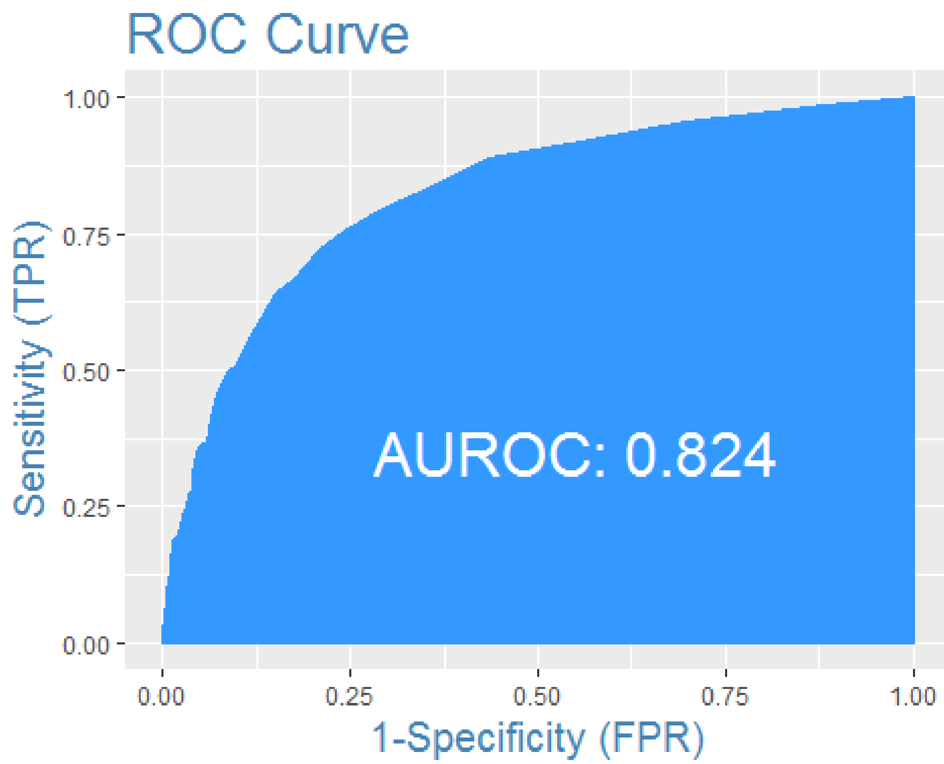
```
##     total_eve_charge + total_night_charge + total_intl_calls + 
##     total_intl_charge + customer_service_calls + international_plan 
+ 
##     vmail_plan, family = "binomial", data = t_train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.0003  -0.5064  -0.3400  -0.2060   2.8689  
## 
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)    
## (Intercept)            -8.847658   0.798580 -11.079  < 2e-16 ***
## total_day_calls         0.005486   0.003676   1.493 0.135566    
## total_day_charge        0.079102   0.008555   9.246  < 2e-16 ***
## total_eve_charge        0.074880   0.017395   4.305 1.67e-05 ***
## total_night_charge      0.086038   0.032176   2.674 0.007495 ** 
## total_intl_calls       -0.090863   0.033915  -2.679 0.007380 ** 
## total_intl_charge       0.380360   0.100656   3.779 0.000158 ***
## customer_service_calls  0.532022   0.056283   9.453  < 2e-16 ***
## international_plan      2.442143   0.197008  12.396  < 2e-16 ***
## vmail_plan             -0.822527   0.184869  -4.449 8.62e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1632.5  on 1993  degrees of freedom
## Residual deviance: 1266.3  on 1984  degrees of freedom
## AIC: 1286.3
## 
## Number of Fisher Scoring iterations: 6
```

Lets try to analyse the confusion matrix and model accuracy

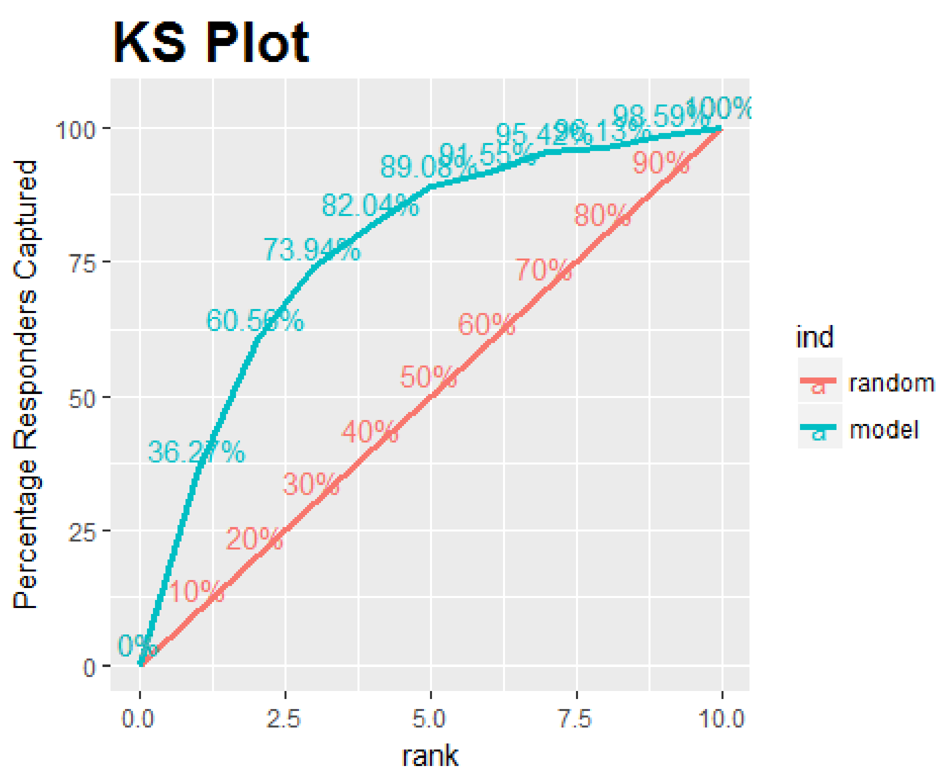
```
t_train$score=predict(mod2,newdata=t_train,type = "response")
t_train$churn <- as.factor(t_train$churn)
prediction<-ifelse(t_train$score>=0.6,TRUE,FALSE)
prediction <- as.factor(prediction)
confusionMatrix(prediction,t_train$churn)
## Warning in Ops.factor(predictedScores, threshold): '<' not 
meaningful for
## factors
## [1] FALSE TRUE 
## <0 rows> (or 0-length row.names)
```

Lets check the AUC and ROC

```
t_train$churn1 <- ifelse(t_train$churn=="TRUE",1,0)
plotROC(actuals = t_train$churn1,predictedScores = 
as.numeric(fitted(mod2)))
```




```
ks_plot(actuals = t_train$churn1,predictedScores =
as.numeric(fitted(mod2)))
```



```
ks_stat(actuals = t_train$churn1,predictedScores =
as.numeric(fitted(mod2)))
## [1] 0.5131
```

Model Validation with Test Data

```
t_test$score= predict(mod2, t_test, type="response")
t_test$churn <- as.factor(t_test$churn)
test_pred<-ifelse(t_test$score>=0.65,TRUE,FALSE)
test_pred <- as.factor(test_pred)
confusionMatrix(test_pred,t_test$churn)
## Warning in Ops.factor(predictedScores, threshold): '<' not
meaningful for
## factors
## [1] FALSE TRUE
## <0 rows> (or 0-length row.names)
```

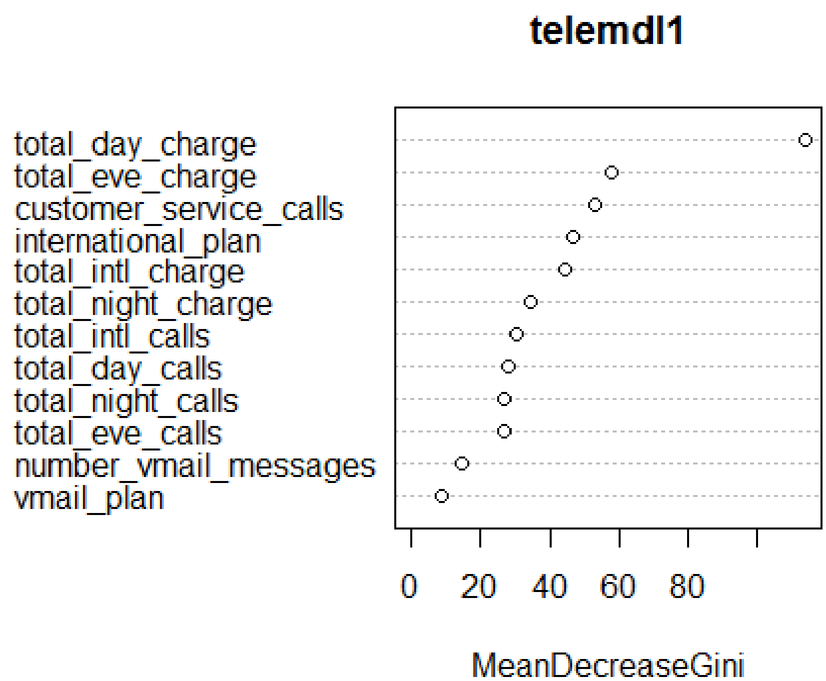
Random Forest

A commonly used class of ensemble algorithms are forests randomized trees. **Random Forest** algorithm is a supervised classification algorithm. As the name suggest, the algorithm creates the forest with a number of trees.

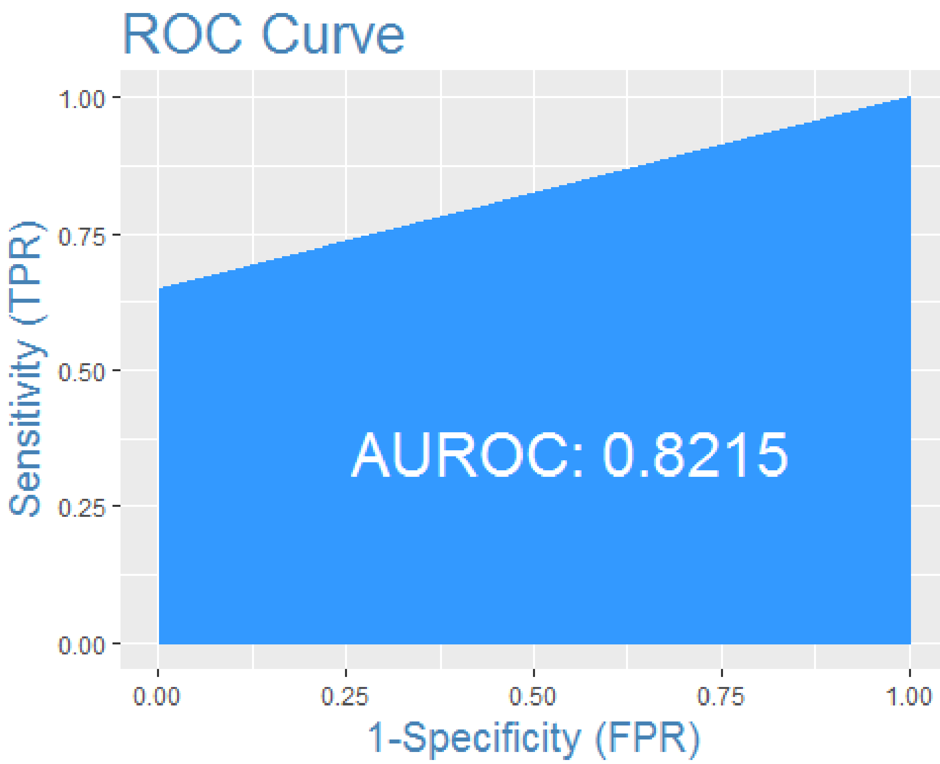
telemdl1=randomForest(as.factor(churn)~. , data = t_train, do.trace=T)

```
telemdl1
##
## Call:
##  randomForest(formula = as.factor(churn) ~ ., data = t_train,
do.trace = T)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 5.82%
## Confusion matrix:
##       FALSE TRUE class.error
## FALSE  1697   13 0.007602339
## TRUE    103  181 0.362676056
importance(telemdl1)
##                        MeanDecreaseGini
## number_vmail_messages         14.518450
## total_day_calls               27.671449
## total_day_charge             114.088851
## total_eve_calls               26.896091
## total_eve_charge              58.001064
## total_night_calls             27.039974
## total_night_charge            34.288640
## total_intl_calls              30.090700
## total_intl_charge             44.174759
## customer_service_calls        53.336674
```

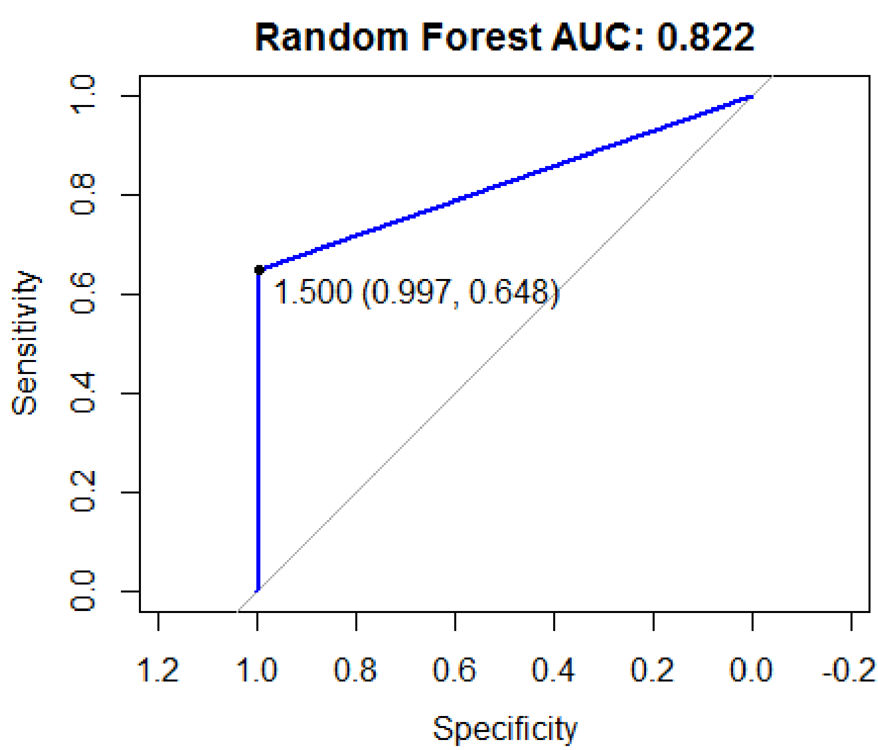
```
## international_plan            46.821798
## vmail_plan                     8.614316
varImpPlot(telemdl1)
```




```
predtele1=predict(telemdl1,t_test)
confusionMatrix(as.factor(predtele1),as.factor(t_test$churn))
## [1] FALSE TRUE
## <0 rows> (or 0-length row.names)
aucrf_test <- roc(as.numeric(t_test$churn), as.numeric(predtele1),
ci=TRUE)
plotROC(as.numeric(t_test$churn), as.numeric(predtele1))
```




```r
plot(aucrf_test, ylim=c(0,1), print.thres=TRUE, main=paste('Random Forest AUC:',round(aucrf_test$auc[[1]],3)),col = 'blue')
```



```
RFTABLE=table(predtele1,as.factor(t_test$churn))

Error=(RFTABLE[1,2]+RFTABLE[2,1])/nrow(t_test)
Error*100
## [1] 5.257009
```

Decision Tree With Stratified Sampling

Decision Trees are the popular and powerful tool used for classification and prediction purposes. it works for both categorical and continuous input and output variables. In this technique, we spilt the population or sample into two or more homogeneous sets based on most significant differentiator in input variables.

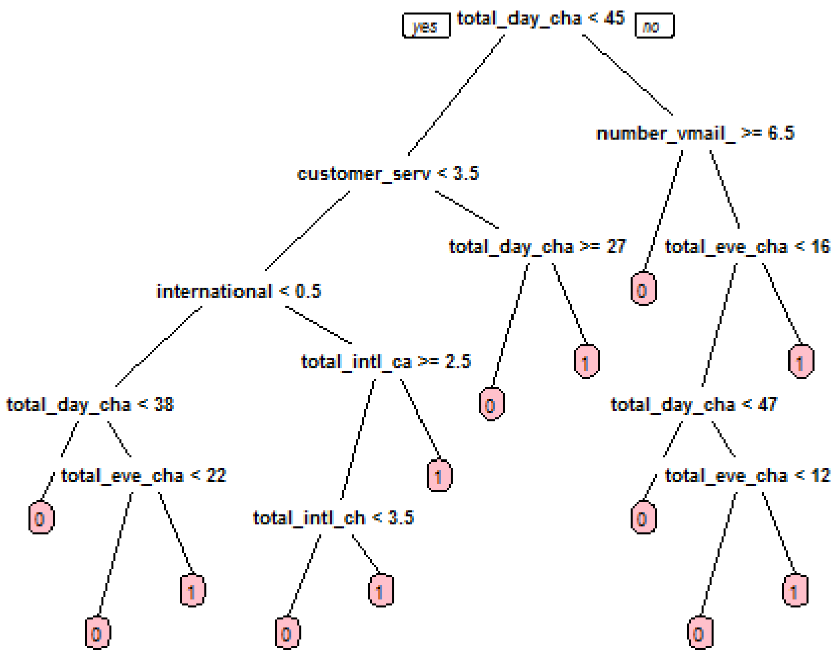
Stratified Sampling

```
tcom_data = final_data

tcom_data$churn <- ifelse(tcom_data$churn=="TRUE",1,0)
d=tcom_data[,"churn"] == "1"
table(d)
## d
## FALSE  TRUE
##  2444   406
classone=tcom_data[d,]
classzero = tcom_data[!d,]

set.seed(1)
d=sample(1:nrow(classone),floor(0.7*nrow(classone)))
classonetrain=classone[d,]
classonetest = classone[-d,]


set.seed(1)
d=sample(1:nrow(classzero),floor(0.7*nrow(classzero)))
classzerotrain=classzero[d,]
classzerotest = classzero[-d,]

P1Index=which(names(tcom_data) %in% "churn")
xtrain=rbind(classonetrain[,-P1Index] , classzerotrain[,-P1Index])
xtest=rbind(classonetest[,-P1Index] , classzerotest[,-P1Index])

ytrain=rbind(classonetrain[P1Index] , classzerotrain[P1Index])
ytest=rbind(classonetest[P1Index] , classzerotest[P1Index])
ytrain = unlist(ytrain)
ytest = unlist(ytest)
```

Decision Tree

```
set.seed(1)
classTree=rpart(as.factor(ytrain)~. , method = "class", data =
xtrain , control = rpart.control(minsplit = 20, cp=0.001))
```

```
BestCP=classTree$cptable[which.min(classTree$cptable[,"xerror"]),"CP"]
BestCP
## [1] 0.01408451
tree.prune=prune(classTree,cp=BestCP)
prp(tree.prune,box.col = c("pink","palegreen3")[classTree$frame$yval])
```




```
classTreeTrain=predict(tree.prune,xtrain)
TrainClassify=apply(classTreeTrain, 1, which.max)
classTreeMat=table(TrainClassify,ytrain)

classTreeTest=predict(tree.prune,xtest,type = "class")
confusionMatrix(classTreeTest,as.factor(ytest))
## Warning in Ops.factor(predictedScores, threshold): '<' not
meaningful for
## factors
## [1] 0 1
## <0 rows> (or 0-length row.names)
Error=(classTreeMat[1,2]+classTreeMat[2,1])/nrow(xtrain)
Error*100
## [1] 5.616851
classTreeTest=predict(tree.prune,xtest)
TestClassify=apply(classTreeTest, 1, which.max)
classTreeTestMat=table(TestClassify,ytest)

Error=(classTreeTestMat[1,2]+classTreeTestMat[2,1])/nrow(xtest)
Error*100
```

```
## [1] 6.658879
```

NaiveBayes Model

It is a classification technique based on Bayes' Theorem with an assumption of *independence among predictors*. The algorithm learns the probability of an object with certain features belonging to a particular group/class.The class with highest probability is considered as the most likely class.In short, it is a **probabilistic classifier**.

```
Bayes=naiveBayes(as.factor(ytrain)~. , data = xtrain)
Bayes
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.8575727 0.1424273
##
## Conditional probabilities:
##    number_vmail_messages
## Y       [,1]     [,2]
##   0 8.662573 13.96682
##   1 4.936620 11.76280
##
##    total_day_calls
## Y       [,1]     [,2]
##   0 100.5953 19.55616
##   1 101.1937 21.34872
##
##    total_day_charge
## Y       [,1]      [,2]
##   0 29.82922  8.474278
##   1 35.19123 12.011882
##
##    total_eve_calls
## Y        [,1]     [,2]
##   0  99.66725 19.76287
##   1 101.07394 19.25204
##
##    total_eve_charge
## Y       [,1]     [,2]
##   0 16.96371 4.393785
##   1 18.19835 4.400033
##
##    total_night_calls
## Y       [,1]     [,2]
```

```
##   0 100.2678 19.60982
##   1 100.2394 20.01216
##
##    total_night_charge
## Y       [,1]     [,2]
##   0 8.954819 2.272377
##   1 9.213521 2.071105
##
##    total_intl_calls
## Y       [,1]     [,2]
##   0 4.483041 2.313748
##   1 4.267606 2.440574
##
##    total_intl_charge
## Y       [,1]      [,2]
##   0 2.746427 0.7390414
##   1 2.835704 0.7279420
##
##    customer_service_calls
## Y       [,1]     [,2]
##   0 1.448538 1.155284
##   1 2.077465 1.704593
##
##    international_plan
## Y         [,1]      [,2]
##   0 0.06374269 0.2443655
##   1 0.28169014 0.4506171
##
##    vmail_plan
## Y        [,1]      [,2]
##   0 0.2970760 0.4571040
##   1 0.1584507 0.3658077
ConfMat=table(predict(Bayes,xtrain),ytrain)
ConfMat
##    ytrain
##        0    1
##   0 1590  165
##   1  120  119
Error=(ConfMat[1,2]+ConfMat[2,1])/nrow(xtrain)
Error*100
## [1] 14.29288
ConfMatTest=table(predict(Bayes,xtest),ytest)
ConfMatTest
##    ytest
##       0   1
##   0 676  67
##   1  58  55
confusionMatrix(predict(Bayes,xtest),as.factor(ytest))
## [1] 0 1
## <0 rows> (or 0-length row.names)
```

```
Error=(ConfMatTest[1,2]+ConfMatTest[2,1])/nrow(xtest)
Error*100
## [1] 14.6028
```

Summary of Model Performance

Model	Accuracy
Logistic Regression	85.75%
Decision Tree	93.34%
Random Forest	***94.74%***
Naïve Bayes	85.4

Conclusion

From above observations, we can conclude **Random Forest** is the best model with accuracy **94.74%** respect to others. So we can predict the test data using Random Forest Model. And we can customize some offer in order to reduce their churn.

Appendix

Packages used for the Classification Analysis:

dplyr	For SQL queries
corrplot	For correlation plot
car	For scatter plot
randomForest	For Random Forest Algorithm
caret	For data pre-processing
pROC	For ROC curve
e1071	for ROC curve and Confusion matrix
lattice	Used for Data Visualization
Rpart, rpart.plot	For rpart & rpart plot
ggplot2	Used for Data Visualization
InformationValue	Used for ROC & KS plot