Churn Prediction in Telecom Industry

Ankit Saxena

24 February 2019

Contents

1. INTRODUCTION

Customer churn is one of the mounting issues of today's rapidly growing and competitive telecom sector. The focus of the telecom sector has shifted from acquiring new customers to retaining existing customers due to the associated high cost. The retention of existing customers also leads to improved sales and reduced marketing cost as compared to new customers. These facts have ultimately resulted in customer churn prediction activity to be an indispensable part of telecom sector's strategic decision making and planning process.

1.1 Churn Prediction

Churn in the terms of telecommunication industry are the customers leaving the current company and moving to another telecom company. With the increasing number of churns, it becomes the operator‘s process to retain the profitable customers known as churn management. In telecommunication industry each company provides the customers with huge incentives to lure them to switch to their services, it is one of the reasons that customer churn is a big problem in the industry nowadays. To prevent this, the company should know the reasons for which the customer decides to move on to another telecom company. It is very difficult to keep customers intact for long duration as they move to the service that suits most of their needs.

1.2 Managing Churns

Churn management is very important for reducing churns as acquiring a new customer is more expensive than retaining the existing ones. Churn rate is the measurement for the number of customers moving out and in during a specific period of time. If the reason for churning is known, the providers can then improve their services to fulfil the needs of the customers. Churns can be reduced by analysing the past history of the potential customers systematically. Large amount of information is maintained by telecom companies for each of their customers that keeps on changing rapidly due to competitive environment. This information includes the details about billing, calls and network data. The huge availability of information arises the scope of using Data mining techniques in the telecom database. The information available can be analysed in different perspectives to provide various ways to the operators to predict and reduce churning. Only the relevant details are used in analysis which contribute to the study from the information given. Data mining techniques are used for discovering the interesting patterns within data. One of the most common data mining technique is Classification, its aim is to classify unknown cases based on the set of known examples into one of the possible classes.

Here, in case of telecom churn, Classification helps learn to predict whether a customer will churn or not based on customer‘s data stored in database.

2. MACHINE-LEARNING FOR CHURN PREDICTION

Total of 5 different classification models were used to predict the churn of customers.

These models were:

- Random Forest
- Decision Tree
- Logistic Regression
- KNN
- Naïve Bayes

Machine-learning techniques have been widely used for evaluating the probability of customer to churn

2.1 Random Forest

Random forests (RF) are an ensemble learning technique that can support classification and regression. It extends the basic idea of single classification tree by growing many classification trees in the training phase. To classify an instance, each tree in the forest generates its response (vote for a class), the model choses the class that has receive the most votes over all the trees in the forest. One major advantage of RF over traditional decision trees is the protection against overfitting which makes the model able to deliver a high performance.

2.2 Decision Tree

Decision Tree (DT) is a model that generates a tree-like structure that represents set of decisions. DT returns the probability scores of class membership. DT is composed of: a) internal Nodes: each node refers to a single variable/feature and represents a test point at feature level; b) branches, which represent the outcome of the test and are represented by lines that finally lead to c) leaf Nodes which represent the class labels. That is how decision rules are established and used to classify new instances. DT is a flexible model that supports both categorical and continuous data. Due to their flexibility they gained popularity and became one of the most commonly used models for churn prediction.

2.3 Logistic Regression

Logistic Regression (LR) is the appropriate regression analysis model to use when the dependent variable is binary. LR is a predictive analysis used to explain the relationship between a dependent binary variable and a set of independent variables. For customer churn, LR has been widely used to evaluate the churn probability as a function of a set of variables or customers' features.

2.4 Bayes Algorithm

Bayes algorithm estimates the probability that an event will happen based on previous knowledge of variables associated with it. Naïve Bayesian (NB) is a classification technique that is based on Bayes' theorem. It adopts the idea of complete variables independence, as the presence/absence of one feature is unrelated to the presence/absence of any other feature. It considers that all variables independently contribute to the probability that the instance belongs to a certain class. NB is a supervised learning technique that bases its predictions for new instances based on the analysis of their ancestors. NB model usually outputs a probability score and class membership. For churn problem, NB predicts the probability that a customer will stay with his service provider or switch to another one.

2.5 Instance – based learning

Also known as memory based learning, new instances are labelled based on previous instances stored in memory. The most widely used instance based learning techniques for classification is K-nearest neighbor (KNN). KNN does not try to construct an internal model and computations are not performed until the classification time. KNN only stores instances of the training data in the features space and the class of an instance is determined based on the majority votes from its neighbors. Instance is labelled with the class most common among its neighbors. KNN determine neighbors based on distance using Euclidian, Manhattan or Murkowski distance measures for continuous variables and hamming for categorical variables. Calculated distances are used to identify a set of training instances (k) that are the closest to the new point, and assign label from these. Despite its simplicity, KNN have been applied to various types of applications. For churn, KNN is used to analyse if a customer churns or not based on the proximity of his features to the customers in each classes.

3. EXPLORATORY ANALYSIS

Exploratory Data Analysis is an initial process of analysis, in which we can summarize characteristics of data such as pattern, trends, outliers, and hypothesis testing using descriptive statistics and visualization.

Since No variable column has null/missing values, so there is no need of doing any exploratory analysis.

Given below is a sample of the data set that we are using to predict the quality of wine:

3.1 Telecom Sample Data (Columns: 1-19)

account length	area code	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls
128	415	no	yes	25	265.1	110	45.07	197.4	99
107	415	no	yes	26	161.6	123	27.47	195.5	103
137	415	no	no	0	243.4	114	41.38	121.2	110
84	408	yes	no	0	299.4	71	50.9	61.9	88

total eve charge	total night minute	total night calls	total night charge	total intl minuts	total intl calls	total intl charge	number custom er service calls	Churn
16.78	244.7	91	11.01	10	3	2.7	1	False.
16.62	254.4	103	11.45	13.7	3	3.7	1	False.
10.3	162.6	104	7.32	12.2	5	3.29	0	False.
5.26	196.9	89	8.86	6.6	7	1.78	2	False.

3.1 Data Preparation and Feature Selection

The Table 1 reflects the list of attributes with description which are considered for evaluation.

Table 1: List of selected attributes

Attributes	Description
Int'l Plan	Whether a customer subscribed international plan or not.
VMail Plan	Whether a customer subscribed Voice Mail plan or not.
Day Charges	A continuous variable that holds day time call charges.
Day Calls	Total No. of calls made during daytime.
Day Mins	No. of minutes that a customer has used in daytime.
CustSer Calls	Total No. of calls made a customer to customer service.
VMail Msg	Indicates number of voice mail messages.
Int'l Calls	Total No. of calls that used as international calls.
Int'l Charges	A continuous variable that holds international call charges
Int'l Mins	No. of minutes that used during international calls.
Eve Calls	Total No. of calls made during evetime.
Eve Charges	A continuous variable that holds evening time call charges.
Eve Mins	No. of minutes that a customer has used at evening time
Night Calls	Total No. of calls made during night time.
Night Mins	No. of minutes that a customer has used in night-time.
Night Charges	A continuous variable that holds night time call charges.
Churn	The Class label whether a customer is churn or non-churn.

Table 2: Organization of attributes

Sets	Description
Objects	{ 3333 distinct objects} in train data { 1667 distinct objects} in test data
Conditional Attributes	{Intl Plan, VMail_Plan, VMailMsg, Day_Mins, Day Charges, Day Calls, Eve_Mins, Eve_Charges, Eve Calls, Intl_Mins, Intl_Calls, Intl_Charges, Night Calls, Night Mins, Night Charges, CustServ_Calls}
Decision Attribute	Churn

3.2 Data pre-processing

Pre-processing includes three steps: a) data transformation, b) data cleaning and c) feature selection.

a) Data transformation: Two of the explanatory variables (International plan and Voice Mail Plan) were transformed from binominal form (yes/no) into binary form (1/0) to be more suitable for the selected models.
b) Data cleaning: This stage includes missing data handling/imputation: Since our dataset have no variable column has null/missing values.
c) Feature Selection: Feature selection is an important step in knowledge discovery process, to identify those relevant variables or attributes from the large number of attributes in a dataset which are too relevant and reduce the computational cost. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. The dataset contains 19 conditional features, including one unique identifier and a decision attribute about telecom/wireless operator. After feature ranking, it include most relevant and ranked attributes in decision table. We have removed the variable “state”, “area code” & “account length” because these variables are redundant in nature and does not carry any useful information for prediction of customer churn.

4. EVALUATION MEASURES

It is nearly impossible to build a perfect classifier or a model that could perfectly characterize all the instances of the test set. To assess the classification results, we count the number of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The FN value actually belongs to Positive P (e.g. TP + FN = P) but is wrongly classified as Negative N (e.g. TN + FP = N); while, FP value is actually part of N but wrongly classified as P. The following measures were used for the evaluation of proposed classifiers.

Sensitivity/recall: it measures the fraction of churn customers who are correctly identified as true churn. Sensitivity/Recall = TP/(TP+FN)

Specificity: It measures the fraction of true non-churn customers who are correctly identified. Specificity = TN/(TN+FP)

Precision: It is characterized by the number of correctly predicted churns over the total number of churns predicted by the proposed approach. Formally, it can be expressed as: Precision = TP/(TP+FP)

Accuracy: Overall accuracy of the classifier can be calculated as:
Accuracy = TP+TN/Total observations

Misclassification error: It is referred to as a classification error where an instance is falsely classified to a class to which the instance does not belong. Different types of misclassification errors can be calculated as:
Misclassification Error = 1- Accuracy

F-Measure: A composite measure of precision and recall to compute the test's accuracy. It can be interpreted as a weighted average of precision and recall.

F-measure = 2*((Precision*Recall)/(Precision+Recall))

5. MODELLING USING PYTHON

We will use different predictive models present in Python for prediction of Churn in telecom industry.

5.1 Decision Tree

Using DecisionTreeClassifier algorithm we designed our predictive churn model with Churn as target variable.
The Confusion matrix obtained is:

	Predicted False	Predicted True
Actual False	1398(TN)	45(FP)
Actual True	67(FN)	157(TP)

Metrics	**Result**
Accuracy	93.28%
Specificity	96.88%
F-measure	0.74
FNR	29.92%
FPR	3.11%

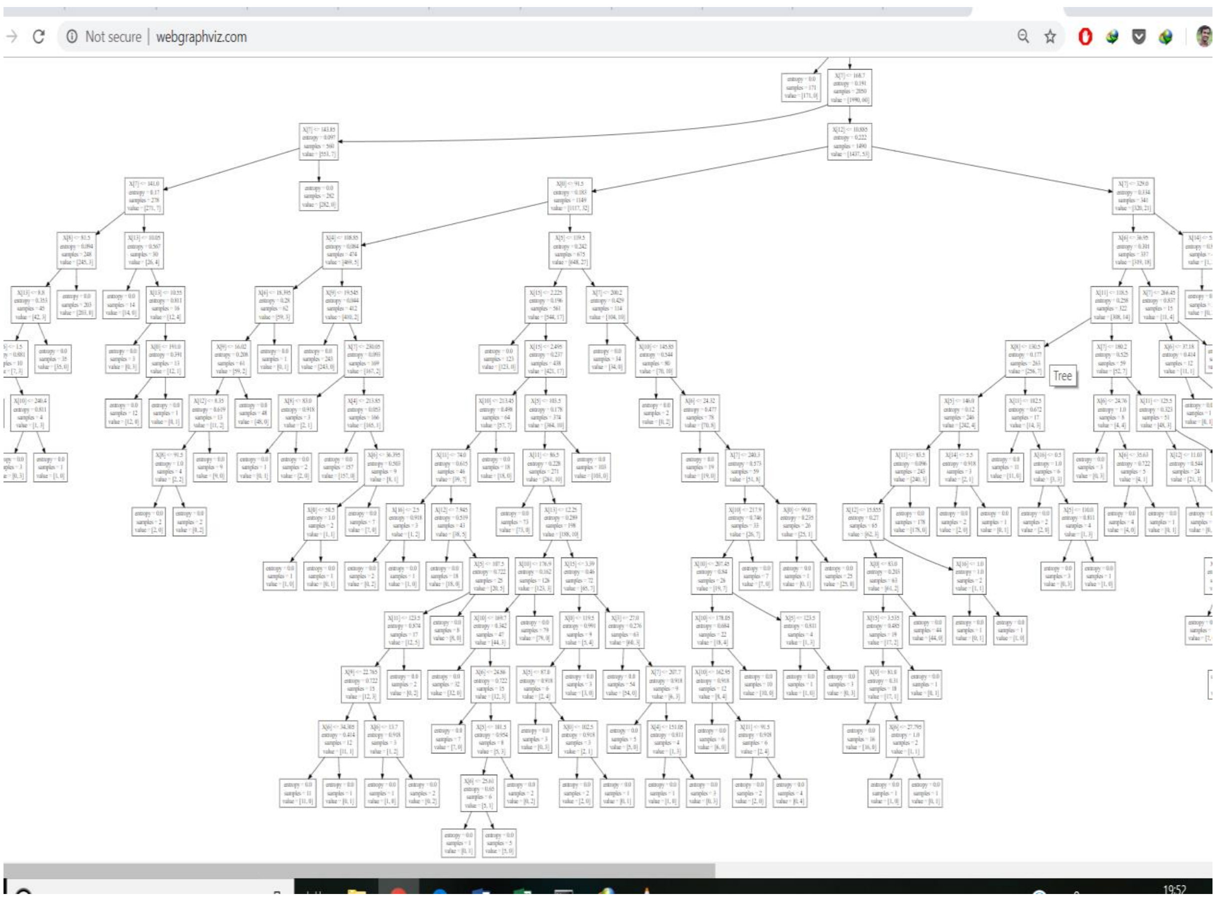


Figure 1: Decision Tree in webgraphviz

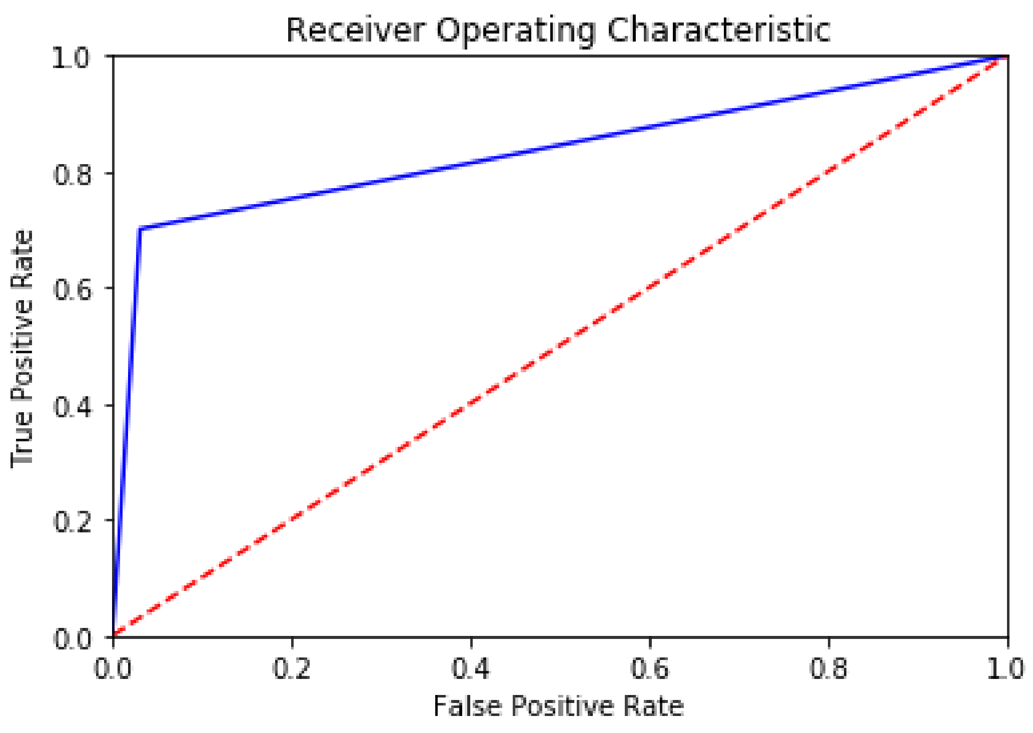


Figure 2: ROC CURVE of Decision Tree model

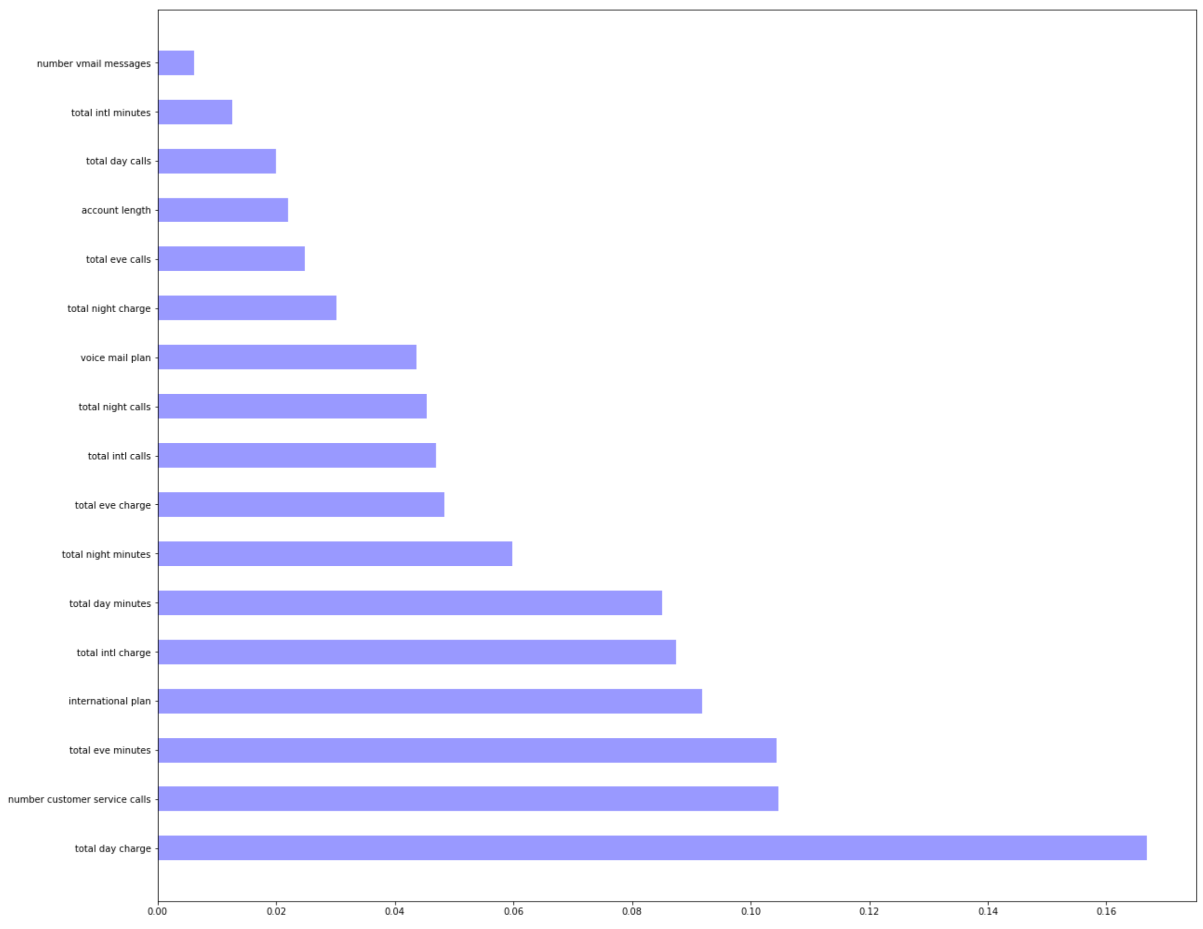


Figure 3: Feature importance in Decision tree model

5.2 Random Forest

Using RandomForestClassifier algorithm we designed our predictive churn model with Churn as target variable.

The Confusion matrix obtained is:

	Predicted False	Predicted True
Actual False	1439(TN)	4(FP)
Actual True	60(FN)	164(TP)

Metrics	**Result**
Accuracy	96.16%
Specificity	99.72%
F-measure	0.83
FNR	26.79%
FPR	0.27%

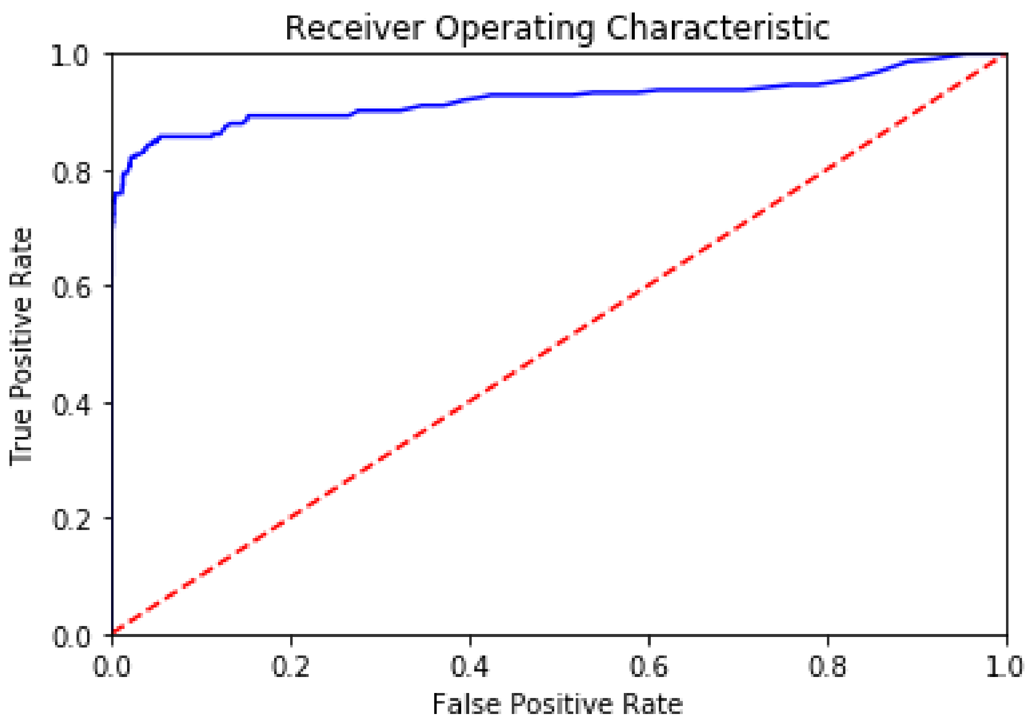


Figure 4: ROC CURVE of Random forest model

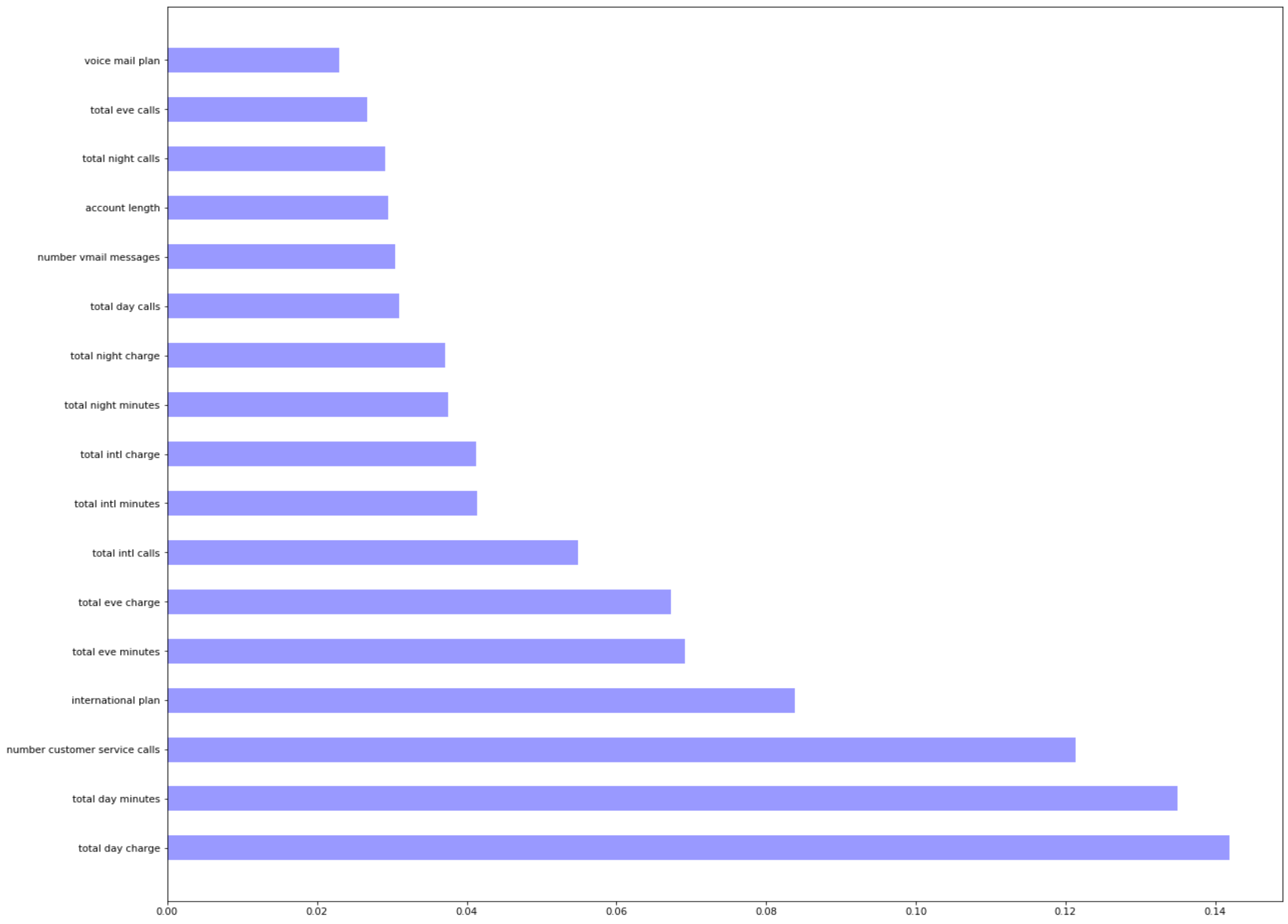
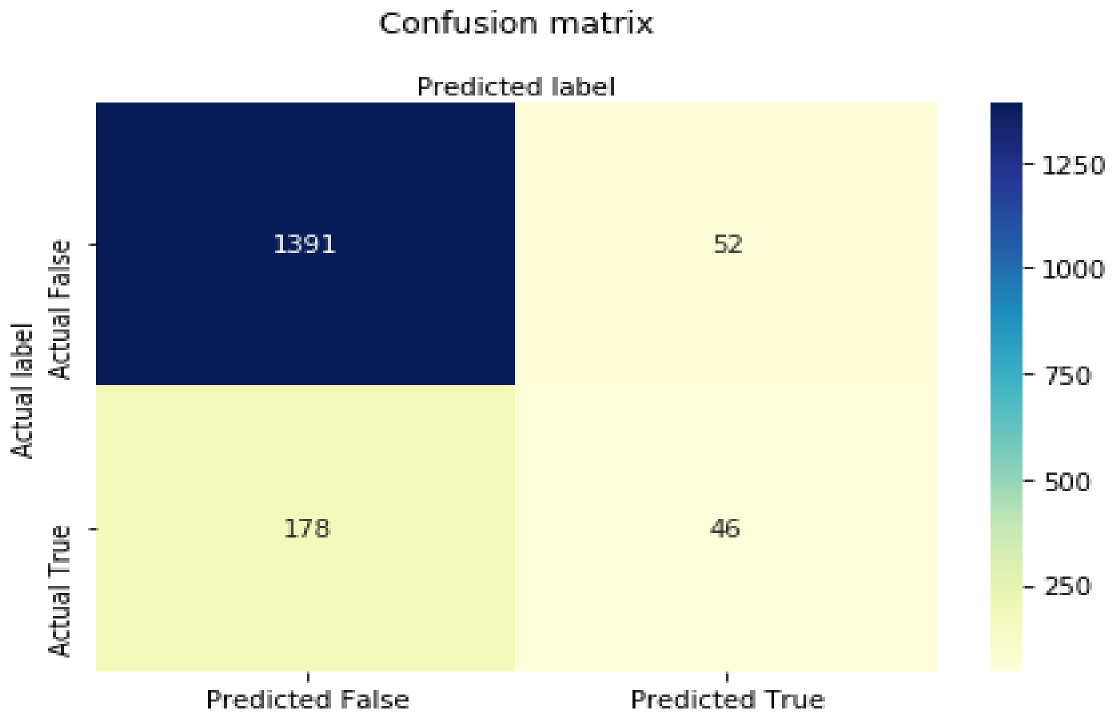


Figure 5: Feature importance in Random forest model

5.3 Logistic Regression

Using RandomForestClassifier algorithm we designed our predictive churn model with Churn as target variable.

The Confusion matrix obtained is:



Metrics	Result
Accuracy	86.20%
Specificity	96.39%
F-measure	0.38

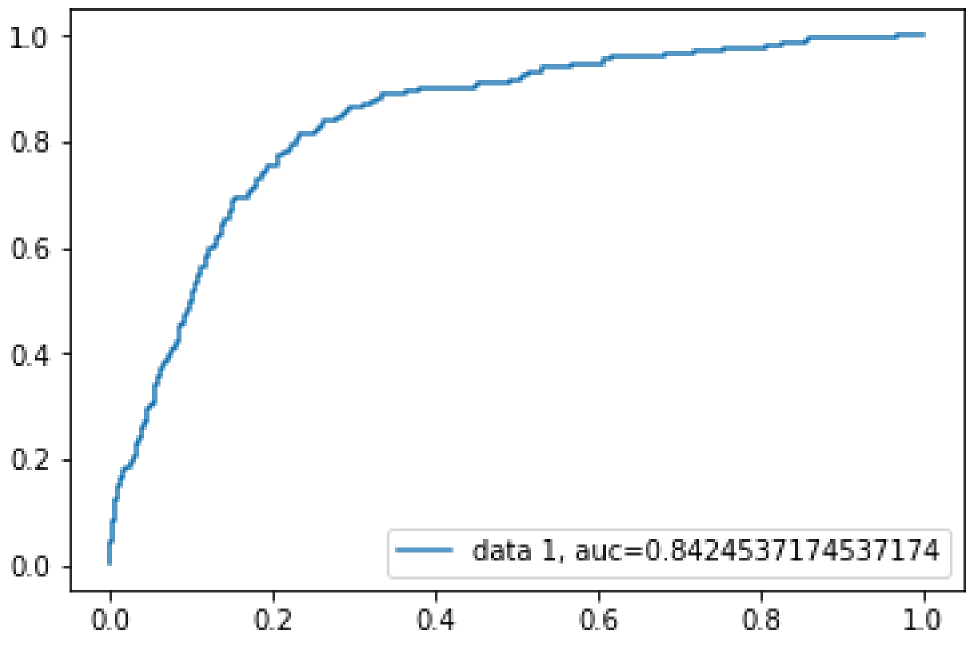


Figure 6 ROC CURVE of Logistic Regression model

Summary of Logistic Regression model built in Python:

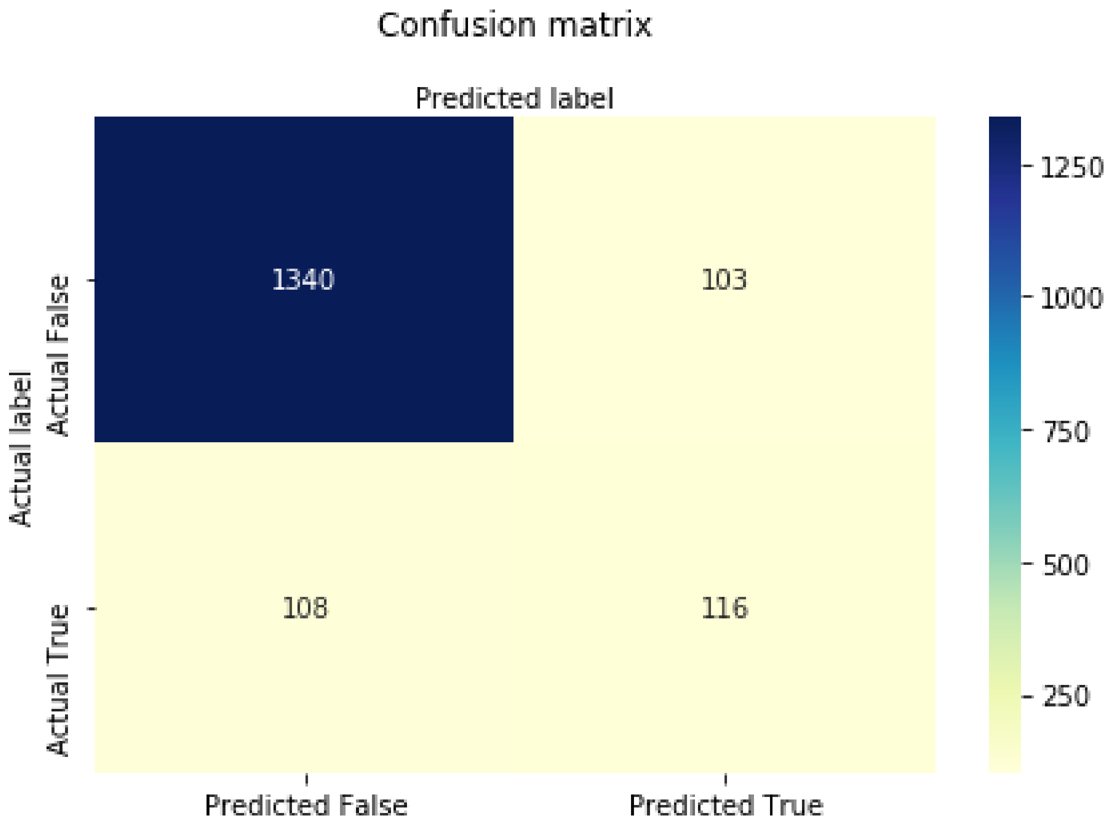
Dep. Variable:	Churn	**No. Observations:**	3333
Model:	Logit	**Df Residuals:**	3308
Method:	MLE	**Df Model:**	24
Date:	Thu, 21 Feb 2019	**Pseudo R-squ.:**	0.2637
Time:	19:09:19	**Log-Likelihood:**	-1015.5
converged:	True	**LL-Null:**	-1379.1
		LLR p-value:	4.348e-138

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
account length	0.0012	0.001	0.823	0.410	-0.002	0.004
number vmail messages	0.0412	0.019	2.201	0.028	0.005	0.078
total day minutes	0.7018	3.411	0.206	0.837	-5.984	7.388
total day calls	0.0026	0.003	0.895	0.371	-0.003	0.008
total day charge	-4.0458	20.066	-0.202	0.840	-43.375	35.283

total eve minutes	0.9710	1.699	0.572	0.568	-2.358	4.300
total eve calls	-5.856e-05	0.003	-0.020	0.984	-0.006	0.006
total eve charge	-11.3347	19.985	-0.567	0.571	-50.504	27.835
total night minutes	0.2003	0.907	0.221	0.825	-1.578	1.979
total night calls	0.0017	0.003	0.567	0.571	-0.004	0.007
total night charge	-4.3539	20.164	-0.216	0.829	-43.875	35.167
total intl minutes	-3.8278	5.507	-0.695	0.487	-14.621	6.966
total intl calls	-0.0888	0.026	-3.446	0.001	-0.139	-0.038
total intl charge	14.5069	20.397	0.711	0.477	-25.470	54.483
international plan_0	-3.5707	5.28e+06	-6.76e-07	1.000	-1.03e+07	1.03e+07
international plan_1	-1.4862	5.28e+06	-2.81e-07	1.000	-1.03e+07	1.03e+07
voice mail plan_0	-1.3659	5.28e+06	-2.59e-07	1.000	-1.03e+07	1.03e+07
voice mail plan_1	-3.6911	5.28e+06	-6.99e-07	1.000	-1.03e+07	1.03e+07
number customer service calls_0	-3.4567	1.256	-2.751	0.006	-5.919	-0.994
number customer service calls_1	-3.6390	1.254	-2.901	0.004	-6.097	-1.181
number customer service calls_2	-3.4099	1.256	-2.715	0.007	-5.871	-0.948
number customer service calls_3	-3.6573	1.262	-2.898	0.004	-6.131	-1.184
number customer service calls_4	-1.2975	1.261	-1.029	0.304	-3.769	1.174
number customer service calls_5	-0.2667	1.280	-0.208	0.835	-2.776	2.243
number customer service calls_6	0.4113	1.338	0.307	0.758	-2.210	3.033

5.4 Naive bayes

Using GaussianNB algorithm we designed our predictive churn model with Churn as target variable.
The Confusion matrix obtained is:



Metrics	Result
Accuracy	87.34%
Specificity	92.86%
F-measure	0.53
FNR	48.21%
FPR	7.13%

5.5 KNN:

Using KNeighborsClassifier algorithm we designed our predictive churn model with Churn as target variable.

We have changed the value of K from 1 to 7 and found that most optimized model is obtained at k=5.

Value of K	**Accuracy**
1	83.20%
3	88.30%
5	89.08%
7	89.02%

Table: 3 Value of K vs Accuracy

i) K=1

	Predicted False	Predicted True
Actual False	1295(TN)	148(FP)
Actual True	132(FN)	92(TP)

Table: 4 Confusion matrix at K=1

Metrics	**Result**
Accuracy	83.20%
Specificity	89.74%
F-measure	0.45
FPR	10.25%

Table: 5 At K=1 Metrics Table

ii) K=3

	Predicted False	Predicted True
Actual False	1400(TN)	43(FP)
Actual True	152(FN)	72(TP)

Table: 6 Confusion matrix at K=3

Metrics	**Result**
Accuracy	88.30%
Specificity	97.02%
F-measure	0.43
FPR	2.97%

Table: 7 At K=3 Metrics Table

iii) K=5

	Predicted False	Predicted True
Actual False	1418(TN)	25(FP)
Actual True	157(FN)	67(TP)

Table: 8 Confusion matrix at K=5

Metrics	**Result**
Accuracy	89.00%
Specificity	98.20%
F-measure	0.42
FPR	1.73%

Table: 9 At K=5 Metrics Table

iv) K=7

	Predicted False	Predicted True
Actual False	1426(TN)	17(FP)
Actual True	166(FN)	58(TP)

Table: 10 Confusion matrix at K=7

Table: 11 At K=7 Metrics Table

Metrics	**Result**
Accuracy	89.02%
Specificity	98.80%
F-measure	0.40
FPR	1.17%

After analysing we found that at K=7, we found most optimal model that can predict churn.

6. MODELLING BY R

Now we will use different predictive models present in R for prediction of Churn in telecom industry.

6.1 Decision Tree

Using package “tree” we have plot the decision tree before pruning & after doing pruning.

Confusion matrix obtained before pruning:

	Predicted False	Predicted True
Actual False	1433(TN)	10(FP)
Actual True	85(FN)	139(TP)

Metrics	**Result**
Accuracy	94.30%
Specificity	99.30%
F-measure	0.74
FNR	37.94%
FPR	0.69%

Confusion matrix obtained after pruning:

	Predicted False	Predicted True
Actual False	1411(TN)	32(FP)
Actual True	66(FN)	158(TP)

Metrics	**Result**
Accuracy	94.12%
Specificity	97.78%
F-measure	0.75
FNR	29.46%
FPR	02.21%

**Although we can clearly see that accuracy decreases after pruning the model by a small amount, but FNR(False negative rate) decreases by 22% so that’s good for our model.

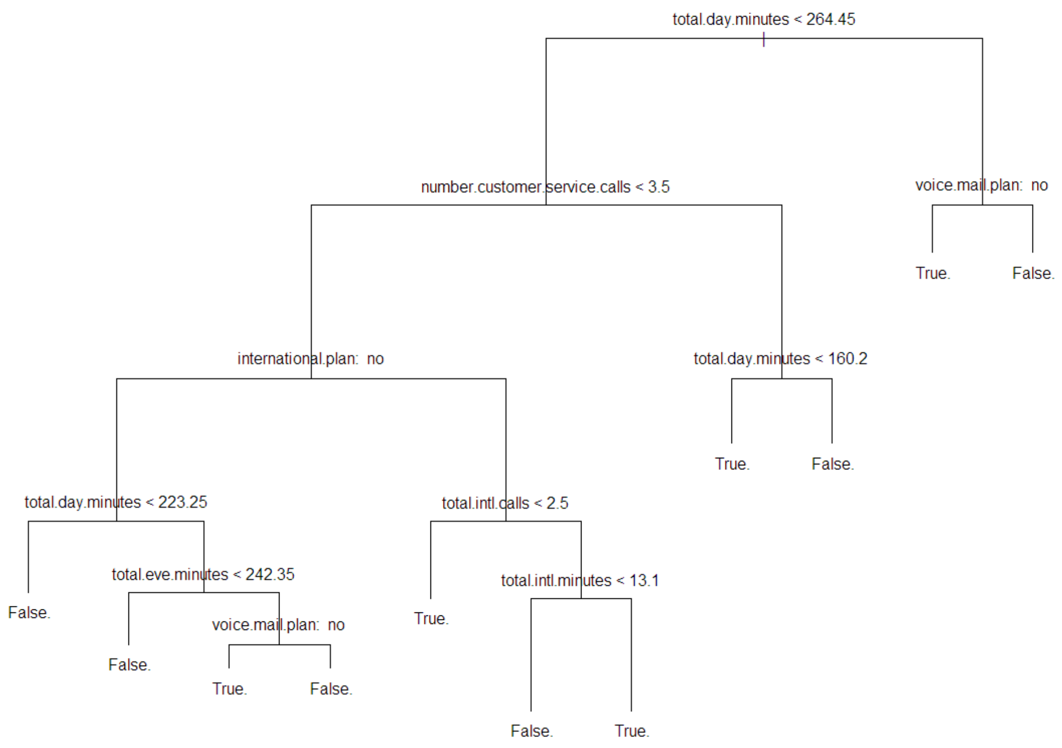

total.day.minutes < 264.45
number.customer.service.calls < 3.5
voice.mail.plan: no
True.
False.
international.plan: no
total.day.minutes < 160.2
True.
False.
total.day.minutes < 223.25
total.intl.calls < 2.5
False.
total.eve.minutes < 242.35
True.
total.intl.minutes < 13.1
False.
voice.mail.plan: no
True.
False.
False.
True.

Figure 7 Decision tree obtained after Pruning

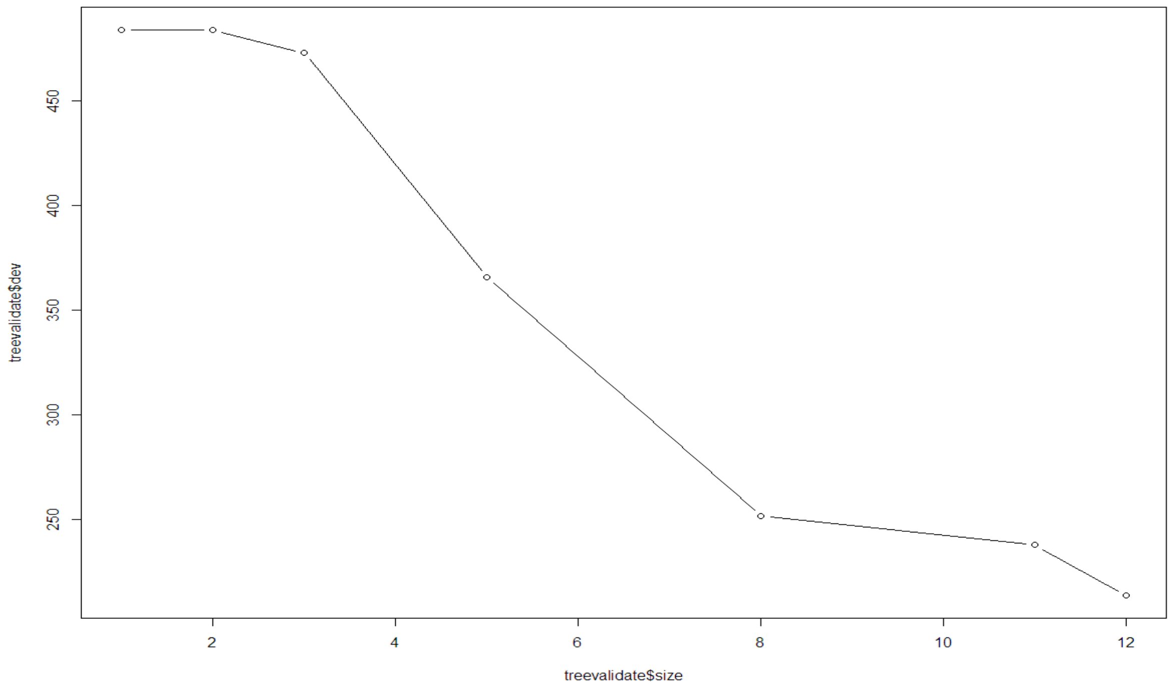


Figure 8 Tree Validation graph obtained

6.2 Random Forest

Using package "randomForest", "RF2List" we have run the random forest model. Confusion matrix obtained after pruning:

	Predicted False	Predicted True
Actual False	1439(TN)	4(FP)
Actual True	62(FN)	162(TP)

Metrics	**Result**
Accuracy	96.04%
Specificity	99.72%
F-measure	0.82
FNR	27.67%
FPR	0.27%

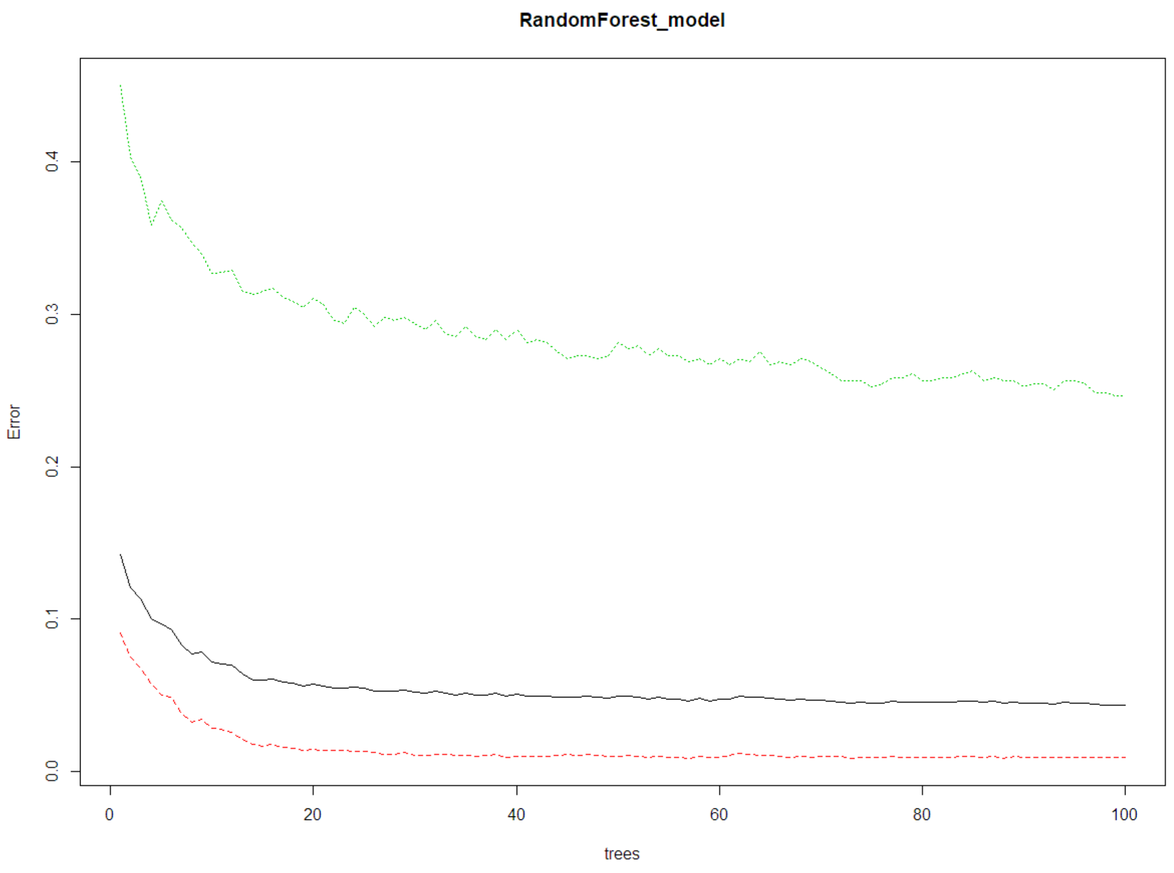


Figure 9 Error vs No. of trees in RandomForest

6.3 Naive Bayes

Using package “e1071” we have run the naïve bayes model.
Confusion matrix obtained :

	Predicted False	Predicted True
Actual False	1393(TN)	50(FP)
Actual True	127(FN)	97(TP)

Metrics	Result
Accuracy	89.38%
Specificity	96.53%
F-measure	0.52
FNR	56.69%
FPR	3.46%

6.4 KNN:

Using knn algorithm we designed our predictive with Churn as target variable.
We have changed the value of K from 1 to 5 and found that most optimized model is obtained at k=5.

Value of K	Accuracy
1	82.60%
3	87.34%
5	88.36%

Table: 12 Value of K vs Accuracy

i) K=1

	Predicted False	Predicted True
Actual False	1295(TN)	142(FP)
Actual True	148(FN)	82(TP)

Table: 13 Confusion matrix at K=1

Metrics	Result
Accuracy	82.60%
Specificity	90.11%
F-measure	0.36
FNR	64.34%
FPR	36.60%

Table: 14 At K=1 Metrics Table

ii) K=3

	Predicted False	Predicted True
Actual False	1390(TN)	158(FP)
Actual True	53(FN)	66(TP)

Table: 15 Confusion matrix at K=3

Metrics	**Result**
Accuracy	87.34%
Specificity	89.79%
F-measure	0.40
FNR	44.53%
FPR	10.20%

Table: 16 At K=3 Metrics Table

iii) K=5

	Predicted False	Predicted True
Actual False	1415(TN)	166(FP)
Actual True	28(FN)	58(TP)

Table: 17 Confusion matrix at K=5

Metrics	**Result**
Accuracy	88.36%
Specificity	89.50%
F-measure	0.42
FNR	32.55%
FPR	10.40%

Table: 18 At K=5 Metrics Table

iv) K=7

	Predicted False	Predicted True
Actual False	1424(TN)	176(FP)
Actual True	19(FN)	48(TP)

Table: 19 Confusion matrix at K=7

Metrics	**Result**
Accuracy	88.30%
Specificity	89.00%
F-measure	0.33
FNR	28.30%
FPR	11.00%

Table: 20 At K=7 Metrics Table

6.5 Logistic Regression

Using "glm" algorithm we have run the logistic regression model.

Summary of model obtained using Logistic Regression model in R:

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1595  -0.5127  -0.3402  -0.1957   3.2500

Coefficients:
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                   -8.4416761  0.9257897  -9.118  < 2e-16 ***
account.length                 0.0008311  0.0013919   0.597 0.550421
area.code                     -0.0004771  0.0013134  -0.363 0.716425
international.plan yes         2.0456277  0.1457194  14.038  < 2e-16 ***
voice.mail.plan yes           -2.0250756  0.5740856  -3.527 0.000420 ***
number.vmail.messages          0.0358748  0.0180108   1.992 0.046388 *
total.day.minutes             -0.2566903  3.2747702  -0.078 0.937522
total.day.calls                0.0032016  0.0027613   1.159 0.246278
total.day.charge               1.5861483 19.2634096   0.082 0.934376
total.eve.minutes              0.7916394  1.6374536   0.483 0.628771
total.eve.calls                0.0010531  0.0027829   0.378 0.705116
total.eve.charge              -9.2280330 19.2640598  -0.479 0.631918
total.night.minutes           -0.1126807  0.8772554  -0.128 0.897795
total.night.calls              0.0007167  0.0028425   0.252 0.800943
total.night.charge             2.5859921 19.4938976   0.133 0.894465
total.intl.minutes            -4.2873631  5.3027500  -0.809 0.418793
total.intl.calls              -0.0928850  0.0250540  -3.707 0.000209 ***
total.intl.charge             16.2026936 19.6390321   0.825 0.409357
number.customer.service.calls  0.5139115  0.0392868  13.081  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2758.3  on 3332  degrees of freedom
Residual deviance: 2158.6  on 3314  degrees of freedom
AIC: 2196.6

Number of Fisher Scoring iterations: 6
```

Confusion matrix obtained :

	Predicted False	Predicted True
Actual False	1413(TN)	30(FP)
Actual True	182(FN)	42(TP)

Metrics	Result
Accuracy	87.28%
Specificity	97.92%
F-measure	0.30
FNR	54.30%
FPR	02.07%

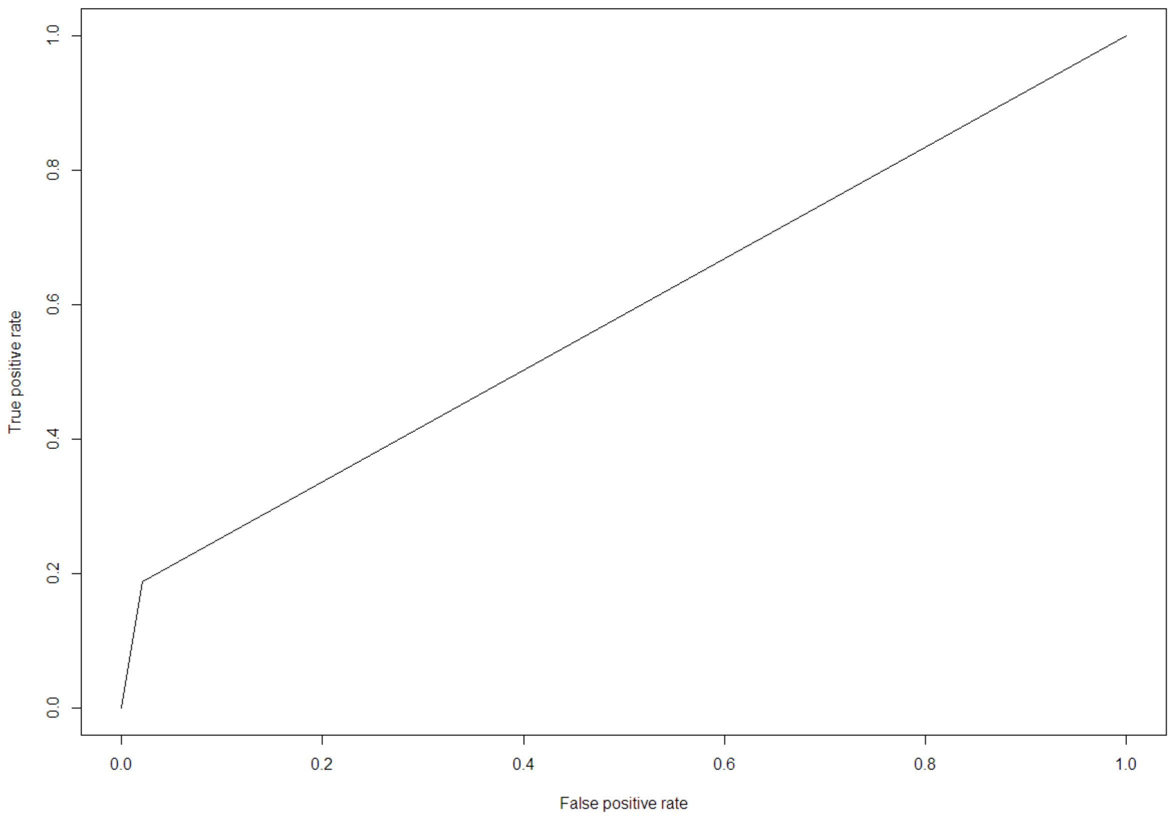


Figure 10 ROC curve obtained in Logistic Regression model

7. CONCLUSION

Customer churn is always a grievous issue for the Telecom industry as customers do not hesitate to leave if they don’t find what they are looking for. Customer churning is directly related to customer satisfaction. There is no standard model which addresses the issues of global telecom service providers accurately. So, we have tried to encounter this issue using various machine learning algorithm through R & Python language.

This model development of churn prediction tries to present a benchmark for the most widely used state of the arts for churn classification. The accuracy of the selected models was evaluated on a public dataset of customers in Telecom Company.

Based on the findings of this model development & According to interpretation of evaluation measures, we recommended both Random forest model and Decision tree model in both R & Python language gave the best accuracy.(with consideration of all other evaluation measures)

8. Appendix A - R Code

1. #DECISION TREE MODEL

```
rm(list=ls())
setwd("C:/Users/Ankit Saxena/Desktop/Edwisor/Project")
getwd()
x = c("C50", "caret", "tree", "rpart","inTrees")
lapply(x, require, character.only = TRUE)
telecom_train = read.csv("Train_data.csv", header = TRUE, sep = ",")
telecom_train$state = NULL
telecom_train$phone.number = NULL
Decision_treemodel = tree(Churn ~., data = telecom_train)
summary(Decision_treemodel)
plot(Decision_treemodel)
text(Decision_treemodel,pretty=0)
telecom_test = read.csv("Test_data.csv", header = TRUE, sep = ",")
telecom_test$phone.number = NULL
telecom_test$state = NULL
c50_predictions = predict(Decision_treemodel, telecom_test[,-21], type = "class")
Confmatrix_c50 = table(telecom_test$Churn, c50_predictions)
confusionMatrix(Confmatrix_c50)
#Accuracy = 94.3%

#After Pruning
set.seed(100)
treevalidate = cv.tree(object = Decision_treemodel,FUN = prune.misclass )
treevalidate
plot(x=treevalidate$size, y=treevalidate$dev, type="b")
PruneDecision_treemodel = prune.misclass(Decision_treemodel, best = 10)
plot(PruneDecision_treemodel)
text(PruneDecision_treemodel, pretty=0)
c50_predictions_1 = predict(PruneDecision_treemodel, telecom_test[,-21], type = "class")
Confmatrix_c50_1 = table(telecom_test$Churn, c50_predictions_1)
confusionMatrix(Confmatrix_c50_1)
#Accuracy = 94.12%
```

2. #RANDOM FOREST MODEL

```
rm(list=ls())
setwd("C:/Users/Ankit Saxena/Desktop/Edwisor/Project")
getwd()
x = c("carot", "e1071", "tree", "rpart", "randomForest","RF2List", "inTrees")
lapply(x, require, character.only = TRUE)
```

```
telecom_train = read.csv("Train_data.csv", header = TRUE, sep = ",")
telecom_train$state = NULL
telecom_train$phone.number = NULL
telecom_test = read.csv("Test_data.csv", header = TRUE, sep = ",")
telecom_test$phone.number = NULL
telecom_test$state = NULL
RandomForest_model = randomForest(Churn ~., telecom_train, importance = TRUE, ntree =
100)
treelist = RF2List(RandomForest_model)
execute = extractRules(treelist, telecom_train[,-19])
execute[1:2,]
readableRules = presentRules(execute, colnames(telecom_train))
readableRules[1:2,]
ruleMetric = getRuleMetric(execute, telecom_train[,-19], telecom_train$Churn)
ruleMetric[1:2,]
RF_predictions = predict(RandomForest_model, telecom_test[,-20])
Confmatrix_RF = table(telecom_test$Churn, RF_predictions)
confusionMatrix(Confmatrix_RF)
#Accuracy = 95.98%
```

3. #LOGISTIC REGRESSION MODEL

```
rm(list=ls())
setwd("C:/Users/Ankit Saxena/Desktop/Edwisor/Project")
getwd()
x = c("caret", "e1071", "ROCR", "caTools")
lapply(x, require, character.only = TRUE)
telecom_train = read.csv("Train_data.csv", header = TRUE, sep = ",")
telecom_train$state = NULL
telecom_train$phone.number = NULL
telecom_test = read.csv("Test_data.csv", header = TRUE, sep = ",")
telecom_test$phone.number = NULL
telecom_test$state = NULL
logistic_model = glm(Churn~., data = telecom_train, family = "binomial")
summary(logistic_model)
logit_predictions = predict(logistic_model, newdata = telecom_test, type = "response")
logit_predictions = ifelse(logit_predictions > 0.5,1,0)
ConfMatrix_Logit = table(telecom_test$Churn, logit_predictions)
ConfMatrix_Logit
#Accuracy = 86.56%
#Sensitivity = 97.92%
pr = prediction(logit_predictions, telecom_test$Churn)
ROC_CURVE = performance(pr, measure = "tpr", x.measure = "fpr")
plot(ROC_CURVE)
```

4. #KNN MODEL

```
rm(list=ls())
setwd("C:/Users/Ankit Saxena/Desktop/Edwisor/Project")
getwd()
library(caret)
library(class)
telecom_train = read.csv("Train_data.csv", header = TRUE, sep = ",")
telecom_train$state = NULL
telecom_train$phone.number = NULL
telecom_test = read.csv("Test_data.csv", header = TRUE, sep = ",")
telecom_test$phone.number = NULL
telecom_test$state = NULL
#Changing character values to numeric value
telecom_train$international.plan <- ifelse(telecom_train$international.plan == 'yes', 1,0)
telecom_train$voice.mail.plan <- ifelse(telecom_train$voice.mail.plan == 'yes', 1,0)
telecom_test$international.plan <- ifelse(telecom_test$international.plan == 'yes', 1,0)
telecom_test$voice.mail.plan <- ifelse(telecom_test$voice.mail.plan == 'yes', 1,0)
#We need not change the datatype as all variables are in numeric
KNN_predictionmodel = knn(telecom_train[,1:18], telecom_test[,1:18],
telecom_train$Churn, k = 1)
Confmatrix_KNN = table(KNN_predictionmodel, telecom_test$Churn)
Confmatrix_KNN
#Accuracy = 82.60%
#Changing value of k=1 to k=3
KNN_predictionmodel = knn(telecom_train[,1:18], telecom_test[,1:18],
telecom_train$Churn, k = 3)
Confmatrix_KNN = table(KNN_predictionmodel, telecom_test$Churn)
Confmatrix_KNN
#Accuracy = 87.34%
#Changing value from k=3 to k=5
KNN_predictionmodel = knn(telecom_train[,1:18], telecom_test[,1:18],
telecom_train$Churn, k = 5)
Confmatrix_KNN = table(KNN_predictionmodel, telecom_test$Churn)
Confmatrix_KNN
sum(diag(Confmatrix_KNN))/nrow(telecom_test)
#Accuracy = 88.36%
#Changing value from k=5 to k=7
KNN_predictionmodel = knn(telecom_train[,1:18], telecom_test[,1:18],
telecom_train$Churn, k = 7)
Confmatrix_KNN = table(KNN_predictionmodel, telecom_test$Churn)
Confmatrix_KNN
```

5. #NAIVE BAYES MODEL

```
rm(list=ls())
setwd("C:/Users/Ankit Saxena/Desktop/Edwisor/Project")
getwd()
library(e1071)
telecom_train = read.csv("Train_data.csv", header = TRUE, sep = ",")
telecom_train$state = NULL
telecom_train$phone.number = NULL
telecom_test = read.csv("Test_data.csv", header = TRUE, sep = ",")
telecom_test$phone.number = NULL
telecom_test$state = NULL
Naivebayes_model = naiveBayes(Churn ~., data = telecom_train)
Naivebayes_predictions = predict(Naivebayes_model, telecom_test[,1:18], type = 'class')
conf_matrix_NB = table(observed = telecom_test[,19], predicted = Naivebayes_predictions)
library(caret)
confusionMatrix(conf_matrix_NB)
#Accuracy = 89.38%
```

9. Appendix B – PYTHON Code

1. DECISION TREE MODEL

```
#Load Libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency
from random import randrange, uniform
import sklearn
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from IPython.display import display, HTML
os.chdir("C:/Users/Ankit Saxena/Desktop/Edwisor/Project/Python")

telecom_train = pd.read_csv("Train_data.csv")

# Drop the columns that we have decided won't be used in prediction
telecom_train  = telecom_train.drop(["phone number", "area code", "state"], axis=1)
features = telecom_train.drop(["Churn"], axis=1).columns

telecom_test = pd.read_csv("Test_data.csv")

telecom_test  = telecom_test.drop(["phone number", "area code", "state"], axis=1)
features = telecom_test.drop(["Churn"], axis=1).columns

#Replace target variable with 0 and 1
telecom_train['Churn'] = telecom_train['Churn'].str.contains('True').astype(int)

telecom_train['international plan'] = telecom_train['international plan'].str.contains('yes').asty
pe(int)

telecom_train['voice mail plan'] = telecom_train['voice mail plan'].str.contains('yes').astype(in
t)

telecom_test['Churn'] = telecom_test['Churn'].str.contains('True').astype(int)

telecom_test['international plan'] = telecom_test['international plan'].str.contains('yes').astype(
int)

telecom_test['voice mail plan'] = telecom_test['voice mail plan'].str.contains('yes').astype(int)

X_train = telecom_train.values[:, 0:17]
X_test = telecom_test.values[:, 0:17]
Y_train = telecom_train.values[:,17]
Y_test = telecom_test.values[:,17]
```

```
Decisiontree_clf = tree.DecisionTreeClassifier(criterion='entropy')

Decisiontree_clf.fit(telecom_train[features], telecom_train["Churn"])
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
DT_predictions = Decisiontree_clf.predict(telecom_test[features])
probs = Decisiontree_clf.predict_proba(telecom_test[features])
display(DT_predictions)
array([0, 0, 0, ..., 0, 0, 0])
score = Decisiontree_clf.score(telecom_test[features], telecom_test["Churn"])
print("Accuracy: ", score)
Accuracy:  0.9310137972405519
dotfile = open("pt.dot", 'w')
df = tree.export_graphviz(Decisiontree_clf, out_file=dotfile)
# Calculate the fpr and tpr for all thresholds of the classification
fpr, tpr, threshold = roc_curve(telecom_test["Churn"], probs[:,1])
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()

fig = plt.figure(figsize=(20, 18))
ax = fig.add_subplot(111)

df_f = pd.DataFrame(Decisiontree_clf.feature_importances_, columns=["importance"])
df_f["labels"] = features
df_f.sort_values("importance", inplace=True, ascending=False)
display(df_f.head(5))

index = np.arange(len(Decisiontree_clf.feature_importances_))
bar_width = 0.5
rects = plt.barh(index , df_f["importance"], bar_width, alpha=0.4, color='b', label='Main')
plt.yticks(index, df_f["labels"])
plt.show()
```

2. RANDOM FOREST MODEL

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency
from random import randrange, uniform
import sklearn
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from IPython.display import display, HTML
from sklearn.ensemble import RandomForestClassifier

os.chdir("C:/Users/Ankit Saxena/Desktop/Edwisor/Project/Python")

telecom_train = pd.read_csv("Train_data.csv")

# Drop the columns that we have decided won't be used in prediction
telecom_train  = telecom_train.drop(["phone number", "area code", "state"], axis=1)
features = telecom_train.drop(["Churn"], axis=1).columns

telecom_test = pd.read_csv("Test_data.csv")

telecom_test  = telecom_test.drop(["phone number", "area code", "state"], axis=1)
features = telecom_test.drop(["Churn"], axis=1).columns

telecom_train['international plan'] = telecom_train['international plan'].str.contains('yes').asty
pe(int)

telecom_train['voice mail plan'] = telecom_train['voice mail plan'].str.contains('yes').astype(in
t)

#Replace target variable with 0 and 1
telecom_train['Churn'] = telecom_train['Churn'].str.contains('True').astype(int)

telecom_test['Churn'] = telecom_test['Churn'].str.contains('True').astype(int)

telecom_test['international plan'] = telecom_test['international plan'].str.contains('yes').astype(
int)

telecom_test['voice mail plan'] = telecom_test['voice mail plan'].str.contains('yes').astype(int)

#Divide data
X_train = telecom_train.values[:, 0:17]
X_test = telecom_test.values[:, 0:17]
Y_train = telecom_train.values[:,17]
Y_test = telecom_test.values[:,17]

Randomforest_model = RandomForestClassifier(n_estimators=500).fit(X_train, Y_train)

RF_predictions = Randomforest_model.predict(X_test)

#Build Confusion matrix
```

```
CM = pd.crosstab(Y_test, RF_predictions)
#Let us save TP,TN,FP,FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
#Check Accuracy
((TP+TN)*100/(TP+TN+FP+FN))
```

96.16076784643072

```
get_ipython().magic('matplotlib inline')
confusion_matrix = pd.DataFrame(
    confusion_matrix(telecom_test["Churn"], RF_predictions),
    columns=["Predicted False", "Predicted True"],
    index=["Actual False", "Actual True"]
)
display(confusion_matrix)
# determine the most suitable scoring metric/s for our situation
fig = plt.figure(figsize=(20, 18))
ax = fig.add_subplot(111)

df_f = pd.DataFrame(Randomforest_model.feature_importances_, columns=["importance"])
df_f["labels"] = features
df_f.sort_values("importance", inplace=True, ascending=False)
display(df_f.head(5))

index = np.arange(len(Randomforest_model.feature_importances_))
bar_width = 0.5
rects = plt.barh(index , df_f["importance"], bar_width, alpha=0.4, color='b', label='Main')
plt.yticks(index, df_f["labels"])
plt.show()


probs = Randomforest_model.predict_proba(telecom_test[features])
# Calculate the fpr and tpr for all thresholds of the classification
fpr, tpr, threshold = roc_curve(telecom_test["Churn"], probs[:,1])
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

3.LOGISTIC REGRESSION MODEL

```
#Load Libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency
from random import randrange, uniform
import sklearn
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from IPython.display import display, HTML
from sklearn import metrics
from sklearn.linear_model import LogisticRegression

os.chdir("C:/Users/Ankit Saxena/Desktop/Edwisor/Project/Python")
telecom_train = pd.read_csv("Train_data.csv")
# Drop the columns that we have decided won't be used in prediction
telecom_train  = telecom_train.drop(["phone number", "area code", "state"], axis=1)
features = telecom_train.drop(["Churn"], axis=1).columns
telecom_test = pd.read_csv("Test_data.csv")
telecom_test  = telecom_test.drop(["phone number", "area code", "state"], axis=1)
features = telecom_test.drop(["Churn"], axis=1).columns
telecom_train['Churn'] = telecom_train['Churn'].str.contains('True').astype(int)
telecom_train['international plan'] = telecom_train['international plan'].str.contains('yes').asty
pe(int)
telecom_train['voice mail plan'] = telecom_train['voice mail plan'].str.contains('yes').astype(in
t)
#Creating logistic data
telecom_train_logit = pd.DataFrame(telecom_train['Churn'])
colnames = ["account length", "number vmail messages","total day minutes", "total day calls"
, "total day charge", "total eve minutes", "total eve calls", "total eve charge", "total night minu
tes", "total night calls", "total night charge", "total intl minutes", "total intl calls", "total intl c
harge"]
telecom_train_logit = telecom_train_logit.join(telecom_train[colnames])
cate_names = ["international plan", "voice mail plan", "number customer service calls"]
for i in cate_names:
    temp = pd.get_dummies(telecom_train[i], prefix = i)
    telecom_train_logit = telecom_train_logit.join(temp)
```

```
telecom_train_logit.head()
#Derived new variables
telecom_train_logit.shape
 (3333, 29)
telecom_test['Churn'] = telecom_test['Churn'].str.contains('True').astype(int)
telecom_test['international plan'] = telecom_test['international plan'].str.contains('yes').astype(
int)
telecom_test['voice mail plan'] = telecom_test['voice mail plan'].str.contains('yes').astype(int)
#Creating logistic data
telecom_test_logit = pd.DataFrame(telecom_test['Churn'])
telecom_test_logit = telecom_test_logit.join(telecom_test[colnames])
for i in cate_names:
    temp = pd.get_dummies(telecom_test[i], prefix = i)
    telecom_test_logit = telecom_test_logit.join(temp)
#Derived new variables
telecom_test_logit.shape
telecom_train_logit.shape
telecom_test_logit.head()
# Drop the columns are not used in telecom_test data
telecom_train_logit  = telecom_train_logit.drop(["number customer service calls_8", "numbe
r customer service calls_9"], axis=1)
features = telecom_train_logit.drop(["Churn"], axis=1).columns
telecom_train_logit.shape
 (3333, 27)
#Select column indexes
telecom_train_cols = telecom_train_logit.columns[1:27]
#Building logistic regression model
import statsmodels.api as sm
Logit_model = sm.Logit(telecom_train_logit['Churn'], telecom_train_logit[telecom_train_col
s]).fit()
Optimization terminated successfully.
         Current function value: 0.304672
         Iterations 8
#Predict test data
telecom_test_logit['Actual_prob'] = Logit_model.predict(telecom_test_logit[telecom_train_c
ols])
Logit_model.summary()
telecom_test_logit['ActualVal'] = 1
telecom_test_logit.loc[telecom_test_logit.Actual_prob < 0.5, 'ActualVal'] = 0
telecom_test_logit.head()
#Build Confusion matrix
```

```
CM = pd.crosstab(telecom_test_logit['Churn'],telecom_test_logit['ActualVal'])
#Let us save TP,TN,FP,FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
#Check Accuracy
((TP+TN)*100/(TP+TN+FP+FN))
```

```
86.20275944811038
```

```
confusion_matrix = pd.DataFrame(
    confusion_matrix(telecom_test_logit['Churn'], telecom_test_logit['ActualVal']),
    columns=["Predicted False", "Predicted True"],
    index=["Actual False", "Actual True"]
)
display(confusion_matrix)
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(confusion_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
logreg = LogisticRegression()
#Divide data
X_train = telecom_train.values[:, 0:17]
X_test = telecom_test.values[:, 0:17]
Y_train = telecom_train.values[:,17]
Y_test = telecom_test.values[:,17]
logreg.fit(X_train,Y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
Y_pred=logreg.predict(X_test)
y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(Y_test,  y_pred_proba)
auc = metrics.roc_auc_score(Y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

4. KNN MODEL

```
#Load Libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency
from random import randrange, uniform
import sklearn
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from IPython.display import display, HTML

os.chdir("C:/Users/Ankit Saxena/Desktop/Edwisor/Project/Python")

telecom_train = pd.read_csv("Train_data.csv")

# Drop the columns that we have decided won't be used in prediction
telecom_train  = telecom_train.drop(["phone number", "area code", "state"], axis=1)
features = telecom_train.drop(["Churn"], axis=1).columns

telecom_test = pd.read_csv("Test_data.csv")

telecom_test  = telecom_test.drop(["phone number", "area code", "state"], axis=1)
features = telecom_test.drop(["Churn"], axis=1).columns

telecom_train['international plan'] = telecom_train['international plan'].str.contains('yes').asty
pe(int)

telecom_train['voice mail plan'] = telecom_train['voice mail plan'].str.contains('yes').astype(in
t)

telecom_test['international plan'] = telecom_test['international plan'].str.contains('yes').astype(
int)

telecom_test['voice mail plan'] = telecom_test['voice mail plan'].str.contains('yes').astype(int)

#Divide data
X_train = telecom_train.values[:, 0:17]
X_test = telecom_test.values[:, 0:17]
Y_train = telecom_train.values[:,17]
Y_test = telecom_test.values[:,17]

from sklearn.neighbors import KNeighborsClassifier
KNN_model = KNeighborsClassifier(n_neighbors = 1).fit(X_train, Y_train)

KNN_predictions = KNN_model.predict(X_test)

#Build Confusion matrix
CM = pd.crosstab(Y_test, KNN_predictions)

confusion_matrix = pd.DataFrame(
    confusion_matrix(telecom_test["Churn"], KNN_predictions),
    columns=["Predicted False", "Predicted True"],
    index=["Actual False", "Actual True"]
)
display(confusion_matrix)
#Check Accuracy
#Let us save TP,TN,FP,FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
((TP+TN)*100/(TP+TN+FP+FN))
83.20335932813437

#Now changing the value of K subsequently to k=3
KNN_model = KNeighborsClassifier(n_neighbors = 3).fit(X_train, Y_train)
KNN_predictions_1 = KNN_model.predict(X_test)
#Build Confusion matrix
CM = pd.crosstab(Y_test, KNN_predictions_1)
#Let us save TP,TN,FP,FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]

#Check Accuracy
((TP+TN)*100/(TP+TN+FP+FN))
88.30233953209358
#Now changing the value of K subsequently to k=5
KNN_model = KNeighborsClassifier(n_neighbors = 5).fit(X_train, Y_train)
KNN_predictions = KNN_model.predict(X_test)
#Build Confusion matrix
CM = pd.crosstab(Y_test, KNN_predictions)
#Let us save TP,TN,FP,FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
#Check Accuracy
((TP+TN)*100/(TP+TN+FP+FN))
89.08218356328734
#Now changing the value of K subsequently to k=7
KNN_model = KNeighborsClassifier(n_neighbors = 7).fit(X_train, Y_train)
KNN_predictions = KNN_model.predict(X_test)
```

```
#Build Confusion matrix
CM = pd.crosstab(Y_test, KNN_predictions)
#Let us save TP,TN,FP,FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
#Check Accuracy
((TP+TN)*100/(TP+TN+FP+FN))
89.02219556088782
#Hence, we can say that most optimal solution is obtained when k=5.

confusion_matrix = pd.DataFrame(
    confusion_matrix(telecom_test["Churn"], KNN_predictions),
    columns=["Predicted False", "Predicted True"],
    index=["Actual False", "Actual True"]
)
display(confusion_matrix)
```

5. NAÏVE BAYES MODEL

```
#Load Libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency
from random import randrange, uniform
import sklearn
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from IPython.display import display, HTML
from sklearn.naive_bayes import GaussianNB

os.chdir("C:/Users/Ankit Saxena/Desktop/Edwisor/Project/Python")
telecom_train = pd.read_csv("Train_data.csv")
# Drop the columns that we have decided won't be used in prediction
telecom_train  = telecom_train.drop(["phone number", "area code", "state"], axis=1)
features = telecom_train.drop(["Churn"], axis=1).columns
```

```
telecom_test = pd.read_csv("Test_data.csv")

telecom_test  = telecom_test.drop(["phone number", "area code", "state"], axis=1)
features = telecom_test.drop(["Churn"], axis=1).columns

telecom_train['international plan'] = telecom_train['international plan'].str.contains('yes').asty
pe(int)

telecom_train['voice mail plan'] = telecom_train['voice mail plan'].str.contains('yes').astype(in
t)

telecom_test['international plan'] = telecom_test['international plan'].str.contains('yes').astype(
int)

telecom_test['voice mail plan'] = telecom_test['voice mail plan'].str.contains('yes').astype(int)

#Divide data
X_train = telecom_train.values[:, 0:17]
X_test = telecom_test.values[:, 0:17]
Y_train = telecom_train.values[:,17]
Y_test = telecom_test.values[:,17]

Naive_bayes_model = GaussianNB().fit(X_train, Y_train)

#Predict test cases
NB_predictions = Naive_bayes_model.predict(X_test)

#Build Confusion matrix
CM = pd.crosstab(Y_test, NB_predictions)

#Let us save TP,TN,FP,FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]

#Check Accuracy
((TP+TN)*100/(TP+TN+FP+FN))

confusion_matrix = pd.DataFrame(
    confusion_matrix(telecom_test["Churn"], NB_predictions),
    columns=["Predicted False", "Predicted True"],
    index=["Actual False", "Actual True"]
display(confusion_matrix)
```

```
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(confusion_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

10. REFERENCES

[1] Anshul Goyal and Rajni Mehta, Assistant Professor,CSE. ―Performance Comparison of Naïve Bayes and J48 Classification Algorithms.

[2] R Data: http://cran.r-project.org/

[3] Data Mining And Business Analytics with R‖, Johannes Ledolter, Dept. of Management Studies, University of IOWA.

[4] International Journal of Engineering and Technical Research (IJETR) ISSN: 2321-0869, Volume-3, Issue-5, May 2015 "Prediction of Churn by Manpreet Kaur, Dr. Prerna Mahajan "

[5] A Proposed Churn Prediction Model‖, Essam Shaaban, Yehia Helmy, Ayman Khedr, Mona Nasr,Department of Information Systems, MUST University, 6 October, Cairo, Egypt and Department of Information Systems, Helwan University, Cairo, Egypt.

[6] (IJACSA) International Journal of Advanced Computer Science and Applications, Churn Prediction in Telecommunication Using Data Mining Technology‖, Rahul J. Jadhav of Bharati Vidyapeeth Deemed University,Pune, Yashwantrao Mohite of Institute of Management, Karad and Usharani T. Pawar2, Shivaji University Kolhapur.

[7] Elegant Graphics for Data Analysis, Wickham, H. ―http://www.springer.com/978-0-387-981040-6.

[8] Han and Kamber. Data Mining: Concepts and Techniques. Second Morgan Kaufman Publisher, 2006

[9] Abolfazl Kazemi, Mohammad Esmaeil Babaei, "Modelling Customer Attraction Prediction in Customer Relation Management using Decision Tree: A Data Mining Approach", journal of Optimization in Industrial Engineering , 2011.

[10] Wikipedia ―www.wikipedia.com‖.