CUSTOMER CHURN PREDICTON

CONTENTS

Chapter 1

Introduction

1.1 Problem statement

Churn (loss of customers to competition) is a problem for companies because it is more expensive to acquire a new customer than to keep your existing one from leaving. This problem statement is targeted at enabling churn reduction using analytics concepts. The objective of this case is to predict customer behaviour- whether he moved out or not based on the customer usage pattern. The aim is thus to predict the churn score based on usage pattern.

1.2 Data

Our task is to build a classification model which predicts whether a customer will churn (move out or not).

We have been provided data that is already split into train and test dataset.

The following table shows a sample of the whole data formed after combining the train and test dataset. To be able to correctly identify the records of the train and test dataset, we have first created a new variable "isTrain" in both datasets. The value of "isTrain" is set to TRUE for training dataset and FALSE for testing dataset. This helps to perform all the pre-processing on the whole data and later successfully dividing the complete data records into the correct training and testing dataset.

Table 1.1 Customer Churn Dataset (Columns 1 to 11)

State	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes
KS	128	415	382-4657	No	yes	25	265.1	110	45.07	197.4
OH	107	415	371-7191	No	yes	26	161.6	123	27.47	195.5
NJ	137	415	358-1921	No	no	0	243.4	114	41.38	121.2
OH	84	408	375-9999	Yes	no	0	299.4	71	50.9	61.9
OK	75	415	330-6626	Yes	no	0	166.7	113	28.34	148.3

Table 1.1 Customer Churn Dataset (Columns 11 to 22)

total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	number customer service calls	Churn	isTrain
99	16.78	244.7	91	11.01	10	99	16.78	244.7	91	TRUE
103	16.62	254.4	103	11.45	13.7	103	16.62	254.4	103	TRUE
110	10.3	162.6	104	7.32	12.2	110	10.3	162.6	104	TRUE
88	5.26	196.9	89	8.86	6.6	88	5.26	196.9	89	TRUE
122	12.61	186.9	121	8.41	10.1	122	12.61	186.9	121	TRUE

The dataset has 21 independent variables or predictors that help to estimate the class of the target variable “Churn”. The following table lists all the independent variables in the dataset.

Table 1.3 Predictor variables

s.no.	Predictor
1	state
2	account length
3	area code
4	phone number
5	international plan
6	voice mail plan
7	number vmail messages
8	total day mnutes
9	total day calls
10	total day charge
11	total eve mnutes
12	total eve calls
13	total eve charge
14	total night mnutes
15	total night calls
16	total night charge
17	total intl mnutes
18	total intl calls
18	total intl charge
20	number customer service calls

Target Variable:

Churn: It has 2 labels yes and no.

“Yes” indicates that the customer will churn and indicates a positive class. While “no” indicates a negative class.

Chapter 2

Methodology

2.1 Pre-processing

During this stage the data is explored and cleaned so as to make it fit for modelling. The data is visualized using different graphs to gain insight about it. Exploratory data analysis begins by exploring the class or data type of the different predictor variables. The data is searched for presence of any missing values that can be either ignored (if more than 30% of data is missing) or imputed using different methods like mean, median, KNN (for numeric data) or mode (for categorical data). The variables are visualized to analyse their distribution (e.g. histograms can be used to visualize the distribution of variables). Outliers from the data are removed as they are inconsistent with the rest of the data. However, in some cases outliers provide interesting insights in cases like fraud detection, cancer detection, churn detection etc and so sometimes they are not removed. Further variables are selected that contribute in future selection. Predictors that carry repetitive information are removed. Feature engineering may also be performed to generate new variables that will have a relation with the target variable. The following subsections will describe the pre-processing steps followed.

2.1.1 Missing value Analysis

The dataset obtained after combining the train and test data is checked for presence of missing values, however, no predictor showed presence of missing value.

2.1.2 Data Exploration

The complete data had 5000 records (3333 records belonged to training set and 1667 belonged to testing set). The data types and class of each predictor was checked. The categorical variables were analyzed to check its impact on the dataset. The data type of each of the 20 predictors is given in table 2.1. Unique values of each variable is analysed to see how the data in each variable is distributed according to the target class. Figure 2.1 illustrates the distribution of each level of area code (408, 415, 510) into the 2 classes (True and False) of the target variable “Churn”. Similar visualizations are developed to view the count of levels in each class of the target variable.

Table 1.3 Data types of predictor variables

s.no.	PREDICTOR	DATA TYPE	CATEGORICAL/CONTINUOUS
1	state	object	Categorical
2	account length	int	Continuous
3	area code	int	Categorical
4	phone number	object	Continuous
5	international plan	object	Categorical
6	voice mail plan	object	Categorical
7	number vmail messages	int	Continuous
8	total day minutes	float	Continuous
9	total day calls	int	Continuous
10	total day charges	float	Continuous
11	total eve minutes	float	Continuous
12	total eve calls	int	Continuous
13	total eve charges	float	Continuous
14	total night minutes	float	Continuous
15	total night calls	int	Continuous
16	total night charges	float	Continuous
17	total intl minutes	float	Continuous
18	total intl calls	int	Continuous
19	total intl charges	float	Continuous
20	number customer service calls	int	Continuous
21	istrain	bool	Categorical

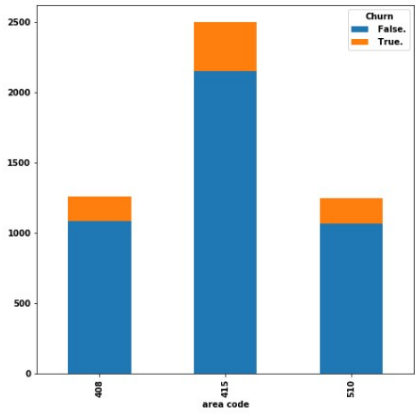


Fig 2.1: area code distribution as per “Churn”

The value counts of each class of the target variable are visualised in figure 2.2. In the complete data 4293 records belong to class FALSE while 707 records belong to class TRUE.

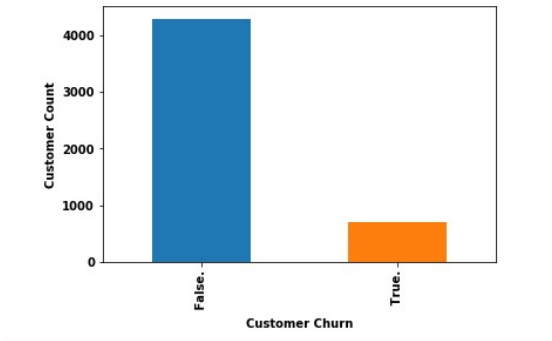


Fig 2.2 Count of each class of target variable “Churn”

2.1.3 Feature Selection

Feature selection is performed to remove variables that carry redundant information. So predictors that are dependent on each other have to be removed. Numeric or continuous variables are tested for dependency using Correlation analysis. Correlation analysis measures the direction and strength of 2 quantitative variables. So in our dataset we apply pair wise correlation between the continuous variables which include- 'account length', 'area code', 'number vmail messages', 'total day minutes', 'total day calls', 'total day charge', 'total eve minutes', 'total eve calls', 'total eve charge', 'total night minutes', 'total night calls', 'total night charge', 'total intl minutes', 'total intl calls', 'total intl charge' and 'number customer service calls'.

As per the results of correlation analysis, we see there are four pairs that are highly correlated with correlation coefficient > 0.95. The visualization of values of correlation matrix is shown in figure 2.3. Hence four variables are removed which are- 'total day charge', 'total eve charge', 'total night charge', 'total intl charge'.

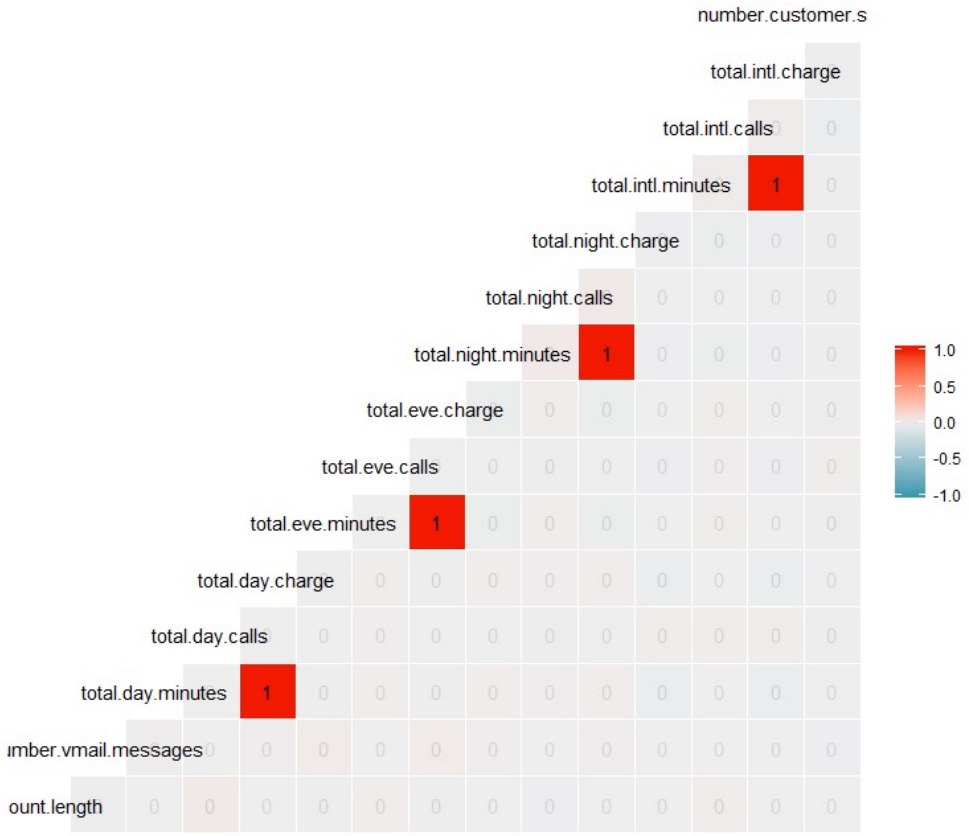


Fig 2.3 Correlation matrix

To measure the dependency between two categorical variables, we employ the Chi-square test. This test assumes that there is no dependency between two independent categorical variables. However there is high dependency between an independent and dependent variable. Hypothesis testing is applied to come to a conclusion.

As per our null hypothesis, H0: the variables are independent.

Alternate hypothesis, H1: the variables are dependent.

After applying the Chi-square test, the p-value of the variables are analyzed. If the p-value of a variable is less than the alpha (acceptance) value, then the null hypothesis is rejected indicating there is a relationship between the independent and target variable. Hence drop those variables for which the p-value is greater than alpha value (0.05). Hence, in our dataset the variable "area code" is dropped.

There are certain other variables that need to be removed like "phone number" as it does not contribute much to the decision making. Another variable "state" is removed as it has numerous levels. Hence, the number of variables in our dataset reduces to 15.

2.1.4 Data Conversion

Before feeding the data to machine learning algorithms the data needs to be converted into the correct form. Most algorithms take only numerical data as input. However our dataset has mixed data. So the categorical variables in the dataset have to be converted into numeric type. There are different ways by which categorical variables can be converted to numeric type like label encoding, one hot encoding, count label encoding etc. We employed label encoding to convert the categorical variables "international plan", "voice mail plan" because it did not suffer with the drawback of internal ordering of the labels as both the variables had only 2 classes- yes and no. The two classes of the target variable "Churn" were also converted into numerical labels.

2.1.5 Handling Target class imbalance problem

Fig 2.2 shows what proportion of Churn variable fall into each class (True or False). This clearly indicates that the target data is imbalanced. 4293 records fall into class no (negative class) while only 707 records fall into class yes (positive class). Thus only 14% data fell under the negative class. According to the problem statement it is clear that the telecom company is more concerned about the positive class as it is more expensive to acquire a new customer than keep the existing ones from leaving. This is because if a customer belonging to negative class (not churn) is classified as being in positive class (will churn) will only lead to companies effort and some expense in preventing this.

However if a customer who is likely to churn (belonging to positive class) is classified into negative class (will not churn) will cause the company to lose the customer. So, to check the model performance different error metrics are used like accuracy, false negative rate, false positive rate, sensitivity, specificity precision etc.

Accuracy specifies the total records that are correctly classified. However, accuracy is misleading for an imbalanced dataset. If all the records are classified as belonging to the majority class (here, negative class) then the accuracy will be nearly 86% which is a good estimate. However it falsely proves a model to be performing good even when all the positive class records were misclassified. So to deal with imbalance datasets we can follow different techniques like using more suitable error metrics on the imbalanced data or try to balance the data so that approximately equal records belong to each class.

a) Error metrics:

i) False negative rate: This metric is important as it will indicate the number of customers that are actually belonging to positive class (likely to churn) but are misclassified. The value of this metric should be as low as possible.

ii) Sensitivity: This metric indicates the number of customers who were correctly classified into positive class that were actually likely to churn. Value of sensitivity should be high for a good model.

iii) Precision: This metric measures the probability of correct detection of positive values from the positive predicted values. The precision value should be high.

iv) ROC (Receiver operating characteristic) curve: This curve shows a trade-off between the true positive rate (sensitivity) and false positive rate (1- specificity = 1- true negative rate). Greater the area under the curve better is the model. An AUC of 1 means that the model is perfect. Fig 2.4 shows a ROC curve generated after applying Decision Tree classifier on the un-sampled data.

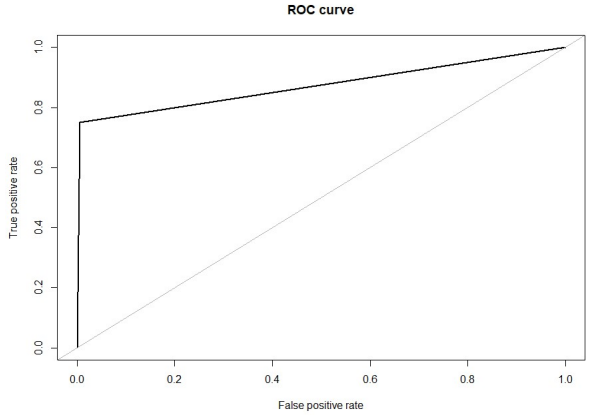


Fig 2.4 example of a ROC curve

b) Re-sampling the existing dataset: There are different methods by which we can bring a balance between the majority and minority class. We can either decrease the records from the majority class or increase the records in the minority class. This technique can be implemented in number of ways:

i) Random under sampling: This sampling technique randomly eliminates majority class instance from the dataset and brings the count, down to the number of minority class instances. However this technique comes with a drawback of losing important piece of information. In our dataset the count of both majority and minority class becomes 707. Earlier the negative (majority) class had 4293 instances, while the positive (minority) class had 707 instances. The total instances in the training set now become 938 and the test set becomes 476 instances.

ii) Random over sampling: This technique randomly selects instances from minority class and replicates them until the count increases to the number of instances in the majority or negative class. However, this technique causes overfitting of the model as the replicated instances leak into the validation set. So over sampling should be performed only on the training set. The training set instances are distributed as: 2850 into majority (negative) class and 483 into minority (positive) class. After random oversampling the count of instances in each class became 2850 making the total count of training data to be 5700. While the number of instances in the test set remain the same as 1667.

iii) Under sampling using Tomek links: Tomek links are pairs of very close instances, but of opposite classes. Removing the instances of the majority class of each pair increases the space between the two classes, facilitating the classification process. After performing under sampling of the whole dataset the distribution of points in the target class is as: Churn (positive/ minority class) -707 and not Churn (majority/ negative class) – 4088.

iv) Over sampling using SMOTE: SMOTE is a “Synthetic Minority Over Sampling Technique”. This technique generates synthetic elements of the minority class by randomly picking elements from minority class, computing their k nearest neighbours and then adding k synthetic points between the chosen point and its k nearest neighbours. After applying SMOTE oversampling, the size of the minority class increases and becomes comparable to the size of the majority class.

v) Hybrid approach: Hybrid approach uses a combination of under-sampling the majority class and oversampling the minority class to achieve a balanced class distribution.
The comparison of results obtained after employing the different sampling techniques are described in the subsequent sections.

2.1.6 Feature scaling:

Feature scaling is performed on all the variables using normalisation so as to bring all variables within the same range (0 to 1). Many classification algorithms like Decision tree, logistic regression etc are unaffected by variable scaling. However, algorithms that work on distance computation like KNN require variable scaling. We have applied normalisation to scale all the variables in the complete dataset. Fig 2.5 shows the distribution of numeric variables.

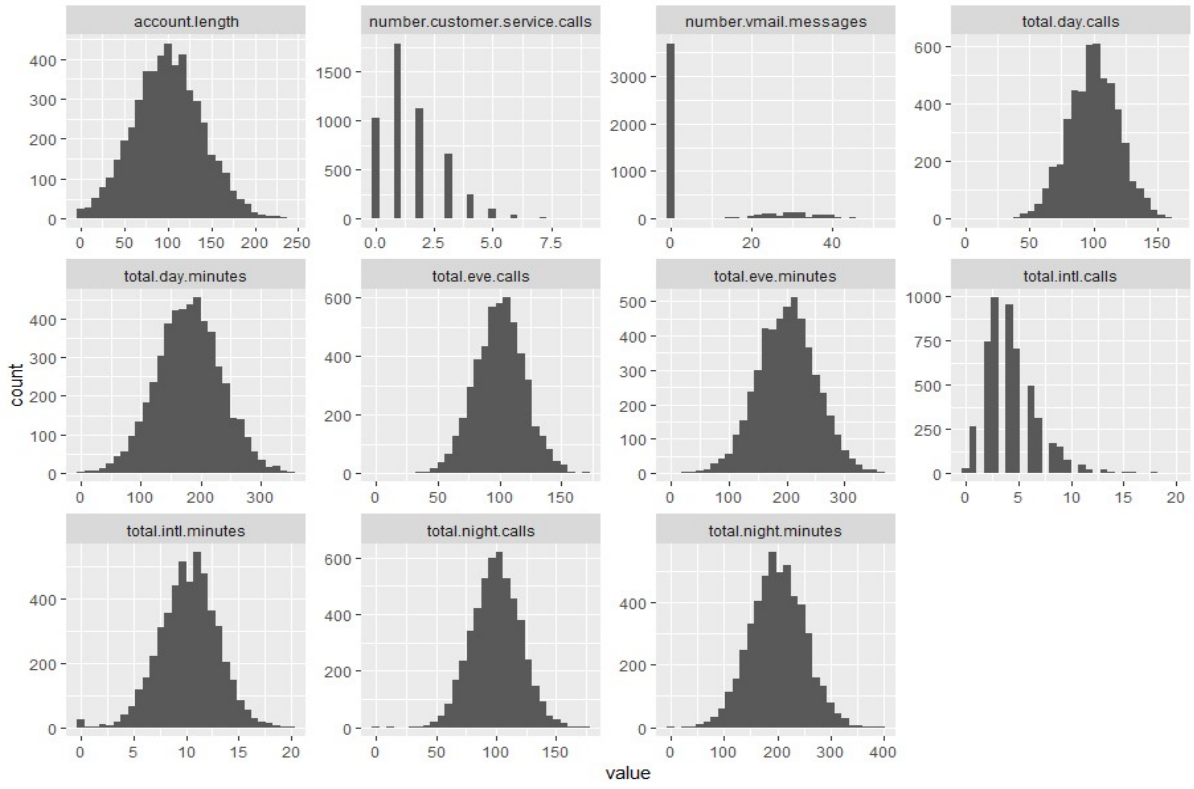


Fig 2.5 Histogram of numeric variables

2.2 Model Development

This section describes the five classification algorithms applied on the dataset to predict the class of the customer to know whether the customer will churn or not churn. Based on the conclusion the telecom company may take suitable actions to reduce the customer churn. Since, the target variable is categorical so this makes it a classification problem. We have five classification algorithms: Decision Tree, Random Forest, Logistic Regression, KNN and Naïve Bayes. The complete dataset is split into train and test set based on the labels of the "isTrain" variable. If the label is True then the record belongs to train set, otherwise it belongs to test set. The classification algorithms are applied on the training set and the performance of the model is tested on the test set using the different error metrics. A confusion matrix is generated based on the actual labels of the test data samples and the predicted test data samples. The confusion matrix returns different values like false negative, false positive, true negative and true positive. These values are used in calculating the different error metrics

2.2.1 Model Performance

This section will present the values of the error metrics for each classification algorithm when applied over raw data and when applied on the re-sampled data. Based on the values of the error metrics the most suitable classifier can be chosen.

a) Decision Tree

	error metric	sampling technique					
		no sampling	random under sampling	random over sampling	under sampling using tomek links	over sampling using smote	hybrid approach
1.	false negative	25	15.62	24.11	24.55	16.52	23.66
2.	accuracy	96.16	86.79	96.28	96.07	94.30	96.28
3.	sensitivity	75	84.38	75.89	75.45	83.48	76.34
4.	specificity	99.45	89.30	99.45	99.36	95.98	99.38
5.	precision	99.45	89.15	95.51	94.94	76.33	95
6.	roc-auc	87.2	86.8	87.7	87.4	89.7	87.9

b) Random Forest

	error metric	sampling technique					
		no sampling	random under sampling	random over sampling	under sampling using tomek links	over sampling using smote	hybrid approach
1.	false negative	28.57	15.18	26.34	29.46	16.52	25
2.	accuracy	95.92	87.02	95.80	95.76	90.94	95.92
3.	sensitivity	71.43	84.82	73.66	70.54	83.48	75
4.	specificity	99.72	89.30	99.24	99.79	92.10	99.17
5.	precision	97.56	89.20	93.75	98.14	62.13	93.33
6.	roc-auc	85.6	87.1	86.4	85.2	87.8	87.1

c) Logistic Regression

	error metric	sampling technique					
		no sampling	random under sampling	random over sampling	under sampling using tomek links	over sampling using smote	hybrid approach
1.	false negative	55.36	6.25	5.80	52.23	11.61	6.70
2.	accuracy	85.30	73.35	55.85	84.95	67.43	57.23
3.	sensitivity	44.64	93.75	94.20	47.77	88.39	93.30
4.	specificity	91.61	52.09	49.90	90.88	34.17	51.63
5.	precision	45.25	67.09	22.59	45.53	27.69	23.04
6.	roc-auc	68.1	72.9	72	69.3	76.3	72.5

d) KNN

K=3	error metric	sampling technique					
		no sampling	random under sampling	random over sampling	under sampling using tomek links	over sampling using smote	hybrid approach
1.	false negative	60.27	23.66	29.46	55.80	34.82	29.46
2.	accuracy	89.62	79.04	81.76	89.80	81.82	81.82
3.	sensitivity	39.73	76.34	70.54	44.20	65.18	70.54
4.	specificity	97.37	81.86	83.51	97.08	84.41	83.58
5.	precision	70.08	81.43	39.90	70.71	39.35	40
6.	roc-auc	68.5	79.1	77	70.6	74.8	77.1

K=5	error metric	sampling technique					
		no sampling	random under sampling	random over sampling	under sampling using tomek links	over sampling using smote	hybrid approach
1.	false negative	59.82	23.66	24.55	57.14	32.14	22.32
2.	accuracy	90.64	79.50	76.72	90.60	84.52	76.84
3.	sensitivity	40.18	76.34	75.45	42.86	67.86	77.68
4.	specificity	98.48	82.79	76.92	98.22	87.11	76.72
5.	precision	80.36	82.21	33.67	79.34	44.97	34.12
6.	roc-auc	69.3	79.6	76.2	70.5	77.5	77.2

K=7	error metric	sampling technique					
		no sampling	random under sampling	random over sampling	under sampling using tomek links	over sampling using smote	hybrid approach
1.	false negative	66.52	23.66	22.32	62.50	29.46	17.41
2.	accuracy	90.04	79.50	74.81	90.17	84.82	76
3.	sensitivity	33.48	76.34	77.68	37.50	70.54	82.59
4.	specificity	98.82	82.79	74.36	98.58	87.04	74.98
5.	precision	81.52	82.21	31.99	80.77	45.80	33.88
6.	roc-auc	66.2	79.6	76	68	78.8	78.8

K=9	error metric	sampling technique					
		no sampling	random under sampling	random over sampling	under sampling using tomek links	over sampling using smote	hybrid approach
1.	false negative	68.30	20.54	20.98	64.29	28.12	18.30
2.	accuracy	90.10	81.78	75.82	90.05	86.26	77.50
3.	sensitivity	31.70	79.46	79.02	35.71	71.88	81.70
4.	specificity	99.17	84.19	75.33	98.72	88.50	76.85
5.	precision	85.54	83.96	33.21	81.63	49.24	35.40
6.	roc-auc	65.4	81.8	77.2	67.2	80.2	79.3

K=11	error metric	sampling technique					
		no sampling	random under sampling	random over sampling	under sampling using tomek links	over sampling using smote	hybrid approach
1.	false negative	71.88	20.09	18.30	65.62	29.02	17.86
2.	accuracy	89.68	82.23	77.68	89.74	86.32	78.46
3.	sensitivity	28.12	79.91	81.70	34.38	70.98	82.14
4.	specificity	99.24	84.65	77.06	98.58	88.70	77.89
5.	precision	85.14	84.43	35.60	79.38	49.38	36.58
6.	roc-auc	63.7	82.3	79.4	66.5	79.8	80

K=13	error metric	sampling technique					
		no sampling	random under sampling	random over sampling	under sampling using tomek links	over sampling using smote	hybrid approach
1.	false negative	73.66	18.75	19.64	66.07	26.79	20.54
2.	Accuracy	89.44	83.37	78.76	89.93	86.50	79.30
3.	Sensitivity	26.34	81.25	80.36	33.93	73.21	79.46
4.	Specificity	99.24	85.58	78.52	98.86	88.57	79.28
5.	Precision	84.29	85.45	36.73	82.61	49.85	37.32
6.	roc-auc	62.8	83.4	79.4	66.4	80.9	79.4

e) Naïve Bayes

	error metric	sampling technique					
		no sampling	random under sampling	random over sampling	under sampling using tomek links	over sampling using smote	hybrid approach
1.	false negative	75	12.95	14.29	73.21	35.27	16.96
2.	Accuracy	87.64	81.09	80.38	87.65	81.82	77.38
3.	Sensitivity	25	87.05	85.71	26.79	64.73	83.04
4.	Specificity	97.37	74.88	79.56	97.36	84.48	76.51
5.	Precision	59.57	78.31	39.43	61.86	39.30	35.43
6.	roc-auc	61.2	81	82.6	62.1	74.6	79.8

The models are compared mainly on the basis of false negative rate, sensitivity, and AUC of ROC curve. A good model has very low false negative rate and high values of sensitivity and ROC-AUC. Accuracy for un-sampled data is slightly misleading.

Chapter 3

Conclusion

From the results described in section 2.2.1 we can come to the conclusion for the best model. The metrics that are of most importance are false negative rate, sensitivity, and AUC-ROC. A good model has low false negative rate (because the positive class holds more importance in this case study), high sensitivity and high AUC-ROC. The above tables also describe the results obtained from each classifier over different re-sampled dataset. It is also clear that the error metrics improve when classifiers are applied over re-sampled data.

Best results are shown in the following cases:

a) Logistic Regression: when applied over data obtained using random under sampling, random over sampling and hybrid approach.
b) Naïve Bayes: when applied over data obtained using random under sampling, random over sampling and hybrid approach.
c) KNN: with k=13 when applied over data obtained using random under sampling, random over sampling and hybrid approach.

The R code and Python code files are attached.

Please find:

1) PROJECT1.R
2) PROJECT1.ipynb

Appendix A – R code

```
rm(list=ls(all=T))
##set working directory
setwd("C:/Users/parul/Desktop/Data Science/PROJECT/project1")

##load libraries
#loading multiple packages at once
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest",'imbalance',
"unbalanced", "C50", "dummies",  "e1071", "Information","MASS", "rpart", "gbm",
"ROSE",'sampling','class','e1071','Metrics','DataCombine','gplots','inTrees','GGally
','purrr','ROCR','tidyr','ggplot2','pROC')

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

## Read the data
train = read.csv("Train_data.csv", header = T, na.strings = c(" ", "",
"NA"),stringsAsFactors = FALSE)
test = read.csv("Test_data.csv", header = T, na.strings = c(" ", "",
"NA"),stringsAsFactors = FALSE)
train$isTrain=TRUE
test$isTrain=FALSE

##combine train and test data to preprocess data before feeding it to ML
algorithms
data1=rbind(train,test)


##************************DATA EXPLORATION*****************************
dim(data1)
str(data1)
data1$international.plan=as.factor(data1$international.plan)
data1$voice.mail.plan=as.factor(data1$voice.mail.plan)
data1$area.code=as.factor(data1$area.code)
data1$Churn=as.factor(data1$Churn)
data1$state=as.factor(data1$state)
```

```
#**************************MISSING VALUE ANALYSIS**********************
#create dataframe with missing percentage
missing_val = data.frame(apply(data1,2,function(x){sum(is.na(x))}))
#convert row names into columns
missing_val$Columns = row.names(missing_val)
row.names(missing_val) = NULL
#Rename the variable conating missing values
names(missing_val)[1] =  "Missing_percentage"
#calculate missing percentage
missing_val$Missing_percentage                                    =
(missing_val$Missing_percentage/nrow(data1)) * 100
missing_val = missing_val[,c(2,1)]
##NO MISSING DATA##
```

```
#*******************************DATA VISUALIZATION************************
print("proportion of Churn in each class: 1: negative class,   2: positive class")
prop.table(table(data1$Churn))
#1. target variable: Churn
ggplot(data1,aes(factor(Churn)))      +geom_bar(fill      =      "coral",alpha      =
0.7)+labs(y="count",x="Churn") + theme_classic()+ggtitle("Customer Churn")

#2.#effect of area code on churn
ggplot(data1,  aes(area.code,  Churn))   +  geom_bar(stat  =  "identity",  aes(fill  =
factor(Churn)))

#3.#effect of state on churn
ggplot(data1,  aes(state,  Churn))   +  geom_bar(stat  =  "identity",  aes(fill  =
factor(Churn)))

#4.#effect of voice mail plan on churn
ggplot(data1, aes(voice.mail.plan, Churn))  + geom_bar(stat = "identity", aes(fill
= factor(Churn)))

#5.#effect of international plan on churn
ggplot(data1,  aes(international.plan,  Churn))   +  geom_bar(stat  =  "identity",
aes(fill = factor(Churn)))

#6.#effect of number of service calls on churn
ggplot(data1,  aes(number.customer.service.calls,  Churn))   +  geom_bar(stat  =
"identity", aes(fill = factor(Churn)))
```

```
##convert factor strings to numeric factor##
##Data Manupulation; convert string categories into factor numeric
for(i in 1:ncol(data1)){

 if(class(data1[,i]) == 'factor'){

  data1[,i] = factor(data1[,i], labels=(1:length(levels(factor(data1[,i])))))

 }
}
#*****************************FEATURE SELECTION*********************
# ## Find correlated independent variables
numeric_index = sapply(data1,is.numeric) #selecting only numeric

numeric_data = data1[,numeric_index]
cnames=colnames(numeric_data)
#visual plot of correlation matrix
ggcorr(data1[cnames],label=TRUE,label_alpha = TRUE)

cormatrix=cor(data1[cnames])
cormatrix[!lower.tri(cormatrix)]=0
#abc.new <- data[,!apply(cormatrix,2,function(x) any(abs(x) > 0.95))]
cor_var=c()
for(i in cnames){
for(j in cnames){
if(abs(cormatrix[j,i])>0.95){
cor_var=append(cor_var,j)
}}}
#remove correlated variables from data
data1=data1[, !colnames(data1) %in% cor_var]

##chi-square test
cat_var=list("state","area.code","internatiional.plan","voice.mail.plan")
factor_index = sapply(data1,is.factor)
factor_data = data1[,factor_index]
```

```
for (i in 1:dim(factor_data)[2])
{
print(names(factor_data)[i])
print(chisq.test(table(factor_data$Churn,factor_data[,i])))
}
#drop the categorical variable for which p-value> 0.05
#Null hypo, H0: predictor and target variable are independent
#Reject H0 when p-value <0.05 (alpha value), hence select (drop) those variables for which p-value<0.05
#Drop phone number as it is an irrelevant variable for churn prediction
drop_var=c("phone.number","area.code")
data1=data1[, !colnames(data1) %in% drop_var]
#drop 'state' as it has too many levels
data1=subset(data1,select=-c(state))

datacopy=data1
data1=datacopy


#**************SOLVING TARGET CLASS IMBALANCE PROBLEM*****************
##divide data into train and test sets and perform Resampling
#load original data
data1=datacopy


#1. Random Over Sampling
#applied only on train data
data1=datacopy
train=subset(data1,isTrain==TRUE)
test=subset(data1,isTrain==FALSE)
table(train$Churn)
train_over=ovun.sample(Churn~. , data=train, method = "over" , N=2850*2)$data
table(train_over$Churn)
#combine to generate complete data
data1=rbind(train_over,test)
```

```
#2. Random under Sampling
#applied on whole data
data1=datacopy
table(data1$Churn)
data1=ovun.sample(Churn~. , data=data1, method = "under" , N=707*2)$data
table(data1$Churn)



# 3. Combining under and over sampling
#applied on train data
data1=datacopy
train=subset(data1,isTrain==TRUE)
test=subset(data1,isTrain==FALSE)
table(train$Churn)
train_both=ovun.sample(Churn~. , data=train, method = "over" , p=0.5)$data
data1=rbind(train_both,test)



# 4. Generate synthetic data using SMOTE oversampling
data1=datacopy
train=subset(data1,isTrain==TRUE)
test=subset(data1,isTrain==FALSE)
table(train$Churn)
train_smote=ubBalance(X=train[,!colnames(train)=="Churn"],Y=train$Churn,
positive=2, type = "ubSMOTE", verbose=TRUE)
train_smote_balanced=cbind(train_smote$X,train_smote$Y)
colnames(train_smote_balanced)[which(names(train_smote_balanced)          ==
"train_smote$Y")] <- "Churn"
train_smote_balanced$isTrain=TRUE
table(train_smote_balanced$Churn)
data1=rbind(train_smote_balanced,test)
#or use SmoteClassif
```

```
#5. Under sampling using TOMEK links
#applied on whole data
data1=datacopy
table(data1$Churn)
#data_tomek=ubBalance(X=data1[,!colnames(data1)=="Churn"],
Y=data1$Churn, positive = 2, type="ubTomek", verbose = TRUE)
library(UBL)
tomek=TomekClassif(Churn~., data1, dist = "HEOM", rem = "maj")
class(tomek)
tomek1=as.data.frame(tomek[[1]])
data1=tomek1
table(data1$Churn)


#**********check numeric variable normality**********
#a.account.length
hist(data1$account.length)
#b.number.vmail.messages
hist(data1$number.vmail.messages)
#c.total.day.minutes
hist(data1$total.day.minutes)
#d.total.day.calls
hist(data1$total.day.calls)
#e.total.eve.minutes
hist(data1$total.eve.minutes)
#f.total.eve.calls
hist(data1$total.eve.calls)
#g.total.night.minutes
hist(data1$total.night.minutes)
#h.total.night.calls
hist(data1$total.night.calls)
#i.total.intl.minutes
hist(data1$total.intl.minutes)
#j.total.intl.calls
hist(data1$total.intl.calls)
#k.number.customer.service.calls
hist(data1$number.customer.service.calls)
```

```
#******** OR VIEW HISTOGRAMS IN SINGLE PANE**********************
data1 %>%
 keep(is.numeric) %>%
 gather() %>%
 ggplot(aes(value)) +
 facet_wrap(~ key, scales = "free") +
 geom_histogram()

#**********************FEATURE SCALING*****************************
#apply normalization on data
numeric_index = sapply(data1,is.numeric) #selecting only numeric
numeric_data = data1[,numeric_index]
cnames=colnames(numeric_data)
for(i in cnames){
 print(i)
 data1[,i] = (data1[,i] - min(data1[,i]))/
   (max(data1[,i] - min(data1[,i])))
}

##********************Apply Classification algorithms************************

errorfunction <- function(cm){
 TN=cm$table[1,1]
 FN=cm$table[1,2]
 FP=cm$table[2,1]
 TP=cm$table[2,2]
 FNR=((FN*100)/(FN+TP))
 acc=(((TP+TN)*100)/(TP+TN+FP+FN))
 sens=(TP*100/(TP+FN))
 spec=(TN*100/(TN+FP))
 prec=(TP*100/(TP+FP))
 cat(sprintf("FALSE NEGATIVE RATE :%.2f %%\nACCURACY :%.2f
%%\nSENSTIVITY :%.2f %%\nSPECIFICITY :%.2f %%\nPRECISION
:%.2f %%",FNR,acc,sens,spec,prec,rec))
}

train=subset(data1,isTrain==TRUE)
train=subset(train,select=-(isTrain))
test=subset(data1,isTrain==FALSE)
test=subset(test,select=-(isTrain))
```

```
#1.DECISION TREE CLASSIFIER
#Develop Model on training data
DT_model = C5.0(Churn ~., train, trials = 100, rules = TRUE)
#Summary of DT model
summary(DT_model)
#write rules into disk
write(capture.output(summary(DT_model)), "DT_Rules.txt")
#Lets predict for test cases
DT_Predictions = predict(DT_model, test[,!colnames(test)=="Churn"], type = "class")
##Evaluate the performance of classification model
ConfMatrix_DT = table(predictions=DT_Predictions,actual=test$Churn)
cm1=confusionMatrix(ConfMatrix_DT, positive='2')
print("DECISION TREE ERROR METRICS")
errorfunction(cm1)
roc.curve(test$Churn,DT_Predictions)

#2.RANDOM FOREST CLASSIFIER

RF_model = randomForest(Churn ~ ., train, importance = TRUE, ntree = 500)
#Extract rules fromn random forest
#transform rf object to an inTrees' format
treeList = RF2List(RF_model)
# #Extract rules
exec = extractRules(treeList, train[,!colnames(test)=="Churn"])  # R-executable conditions
# #Visualize some rules
exec[1:2,]
# #Make rules more readable:
readableRules = presentRules(exec, colnames(train))
readableRules[1:2,]
#Predict test data using random forest model
RF_Predictions = predict(RF_model, test[,!colnames(test)=="Churn"])
##Evaluate the performance of classification model
ConfMatrix_RF = table(predictions=RF_Predictions,actual=test$Churn)
cm2=confusionMatrix(ConfMatrix_RF, positive='2')
print("RANDOM FOREST ERROR METRICS")
errorfunction(cm2)
#ROC-AUC
roc.curve(test$Churn,RF_Predictions)
```

```
#3.Logistic Regression
logit_model = glm(Churn ~ ., data = train, family = "binomial")
#summary of the model
summary(logit_model)
#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test, type = "response")
#convert prob
logit_Predictions = ifelse(logit_Predictions > 0.3, 2, 1)
##Evaluate the performance of classification model
ConfMatrix_RF = table(predictions=logit_Predictions,actual=test$Churn)
cm3=confusionMatrix(ConfMatrix_RF, positive='2')
print("LOGISTIC REGRESSION ERROR METRICS")
errorfunction(cm3)

#ROC-AUC
roc.curve(test$Churn,logit_Predictions)


#4. k-nearest neighbors Classifier
library(class)
#Predict test data
#enter the number of neighbors
k=13
KNN_Predictions = knn(train[,!colnames(test)=="Churn"],
test[,!colnames(test)=="Churn"], train$Churn, k = k)
#Confusion matrix
Conf_matrix = table(KNN_Predictions, test$Churn)
cm4=confusionMatrix(Conf_matrix, positive='2')
sprintf("KNN classifier ERROR METRICS for k= %d",k)
errorfunction(cm4)
roc.curve(test$Churn,KNN_Predictions)
```

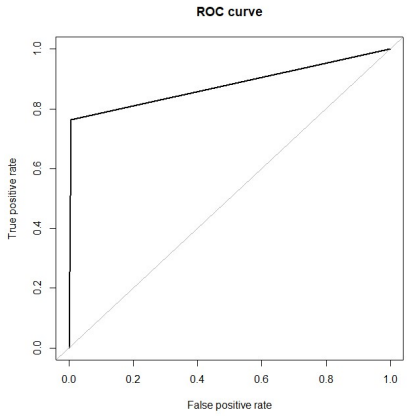
```
#5. Naive Bayes
#Develop model
NB_model = naiveBayes(Churn ~ ., data = train)
#predict on test cases #raw
NB_Predictions = predict(NB_model, test[,!colnames(test)=="Churn"], type = 'class')
#Look at confusion matrix
Conf_matrix = table(predicted = NB_Predictions, actual = test$Churn)
cm5=confusionMatrix(Conf_matrix, positive='2')
print("NAIVE BAYES ERROR METRICS")
errorfunction(cm5)
roc.curve(test$Churn,NB_Predictions)
```

OUTPUT:

Classifiers applied over data obtained from random over sampling:

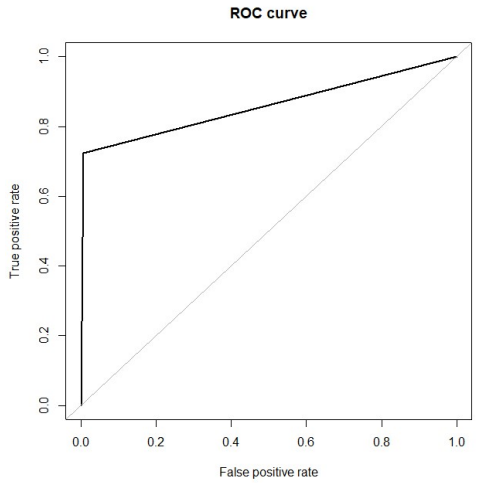
1) Decision Tree

```
[1] "DECISION TREE ERROR METRICS"
> errorfunction(cm1)
FALSE NEGATIVE RATE  :23.66 %
ACCURACY             :96.28 %
SENSTIVITY           :76.34 %
SPECIFICITY          :99.38 %
PRECISION            :95.00 %
> roc.curve(test$Churn,DT_Predictions)
Area under the curve (AUC): 0.879
```



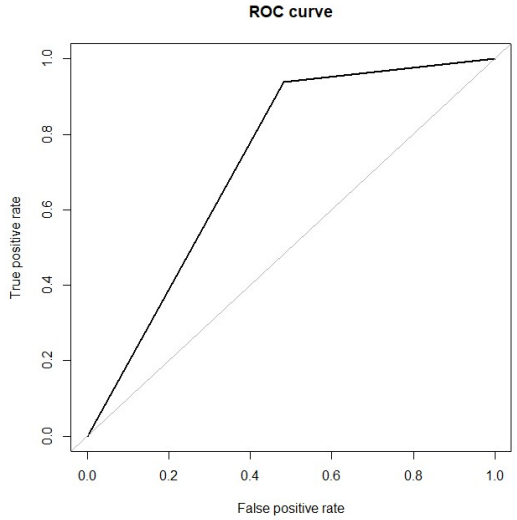
2) Random Forest

```
[1] "RANDOM FOREST ERROR METRICS"
> errorfunction(cm2)
FALSE NEGATIVE RATE  :27.68 %
ACCURACY             :95.68 %
SENSTIVITY           :72.32 %
SPECIFICITY          :99.31 %
PRECISION            :94.19 %> #ROC-AUC
> roc.curve(test$Churn,RF_Predictions)
Area under the curve (AUC): 0.858
```



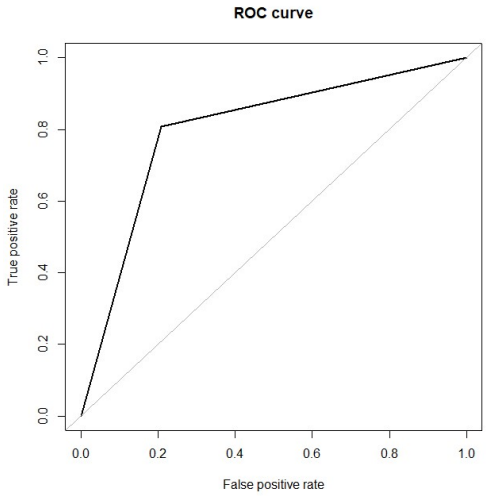
3) Logistic Regression

```
[1] "LOGISTIC REGRESSION ERROR METRICS"
> errorfunction(cm3)
FALSE NEGATIVE RATE  :6.25 %
ACCURACY             :57.41 %
SENSTIVITY           :93.75 %
SPECIFICITY          :51.77 %
PRECISION            :23.18 %
> #ROC-AUC
> roc.curve(test$Churn,logit_Predictions)
Area under the curve (AUC): 0.728
```



4) KNN (with k=13)

```
[1] "KNN classifier ERROR METRICS for k= 13"
> errorfunction(cm4)
FALSE NEGATIVE RATE  :19.20 %
ACCURACY             :79.36 %
SENSTIVITY           :80.80 %
SPECIFICITY          :79.14 %
PRECISION            :37.55 %
> roc.curve(test$Churn,KNN_Predictions)
Area under the curve (AUC): 0.800
```



5) Naïve Bayes

```
[1] "NAIVE BAYES ERROR METRICS"
> errorfunction(cm5)
FALSE NEGATIVE RATE  :15.18 %
ACCURACY             :78.52 %
SENSTIVITY           :84.82 %
SPECIFICITY          :77.55 %
PRECISION            :36.96 %
> roc.curve(test$Churn,NB_Predictions)
Area under the curve (AUC): 0.812
```