Churn Reduction

Robin Redhu

24th May 2018

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 in R and Python

1.2 Dataset

Our task is to build a classification model which will classify the behaviour of customer whether he/she will remain with the company or churn out. Given below a sample data that I am using to train the model which contain 21 columns and 3333 observations.

	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 calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	n cu s
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7	91	11.01	10.0	3	2.70	
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4	103	11.45	13.7	3	3.70	
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2	5	3.29	
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86	6.6	7	1.78	
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1	3	2.73	

(Table 1.1 Following are the first 5 rows of Train dataset)

Now table 1.2 specifies test data. It contains 1667 instances of customer details and 21 variables.

	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 calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	n cus s
0	HI	101	510	354-8815	no	no	0	70.9	123	12.05	...	73	18.01	236.0	73	10.62	10.6	3	2.86	
1	MT	137	510	381-7211	no	no	0	223.6	86	38.01	...	139	20.81	94.2	81	4.24	9.5	7	2.57	
2	OH	103	408	411-9481	no	yes	29	294.7	95	50.10	...	105	20.17	300.3	127	13.51	13.7	6	3.70	
3	NM	99	415	418-9100	no	no	0	216.8	123	36.86	...	88	10.74	220.6	82	9.93	15.7	2	4.24	
4	SC	108	415	413-3643	no	no	0	197.4	78	33.56	...	101	10.54	204.5	107	9.20	7.7	4	2.08	

(Table 1.2 Following are the first 5 rows of test dataset)

Following is the list of 21 variables which will classify whether a customer will churn or not.

S.No	Attribute
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 minutes
9	total day calls
10	total day charges
11	total eve minutes
12	total eve calls
13	total eve charges
14	total night minutes

S.No	Attribute
15	total night calls
16	total night charges
17	total intl minutes
18	total intl calls
19	total intl charges
20	number customer service calls
21	Churn

On the basis on these parameters we predict the classify the customer behaviour.

Chapter 2

Methodology

2.1 Pre Processing

Any predictive modelling requires that we should look at the data first which we will be going to use to train the model because if the data used will be uncleaned and unstructured then it will not produce good results which may lead to our prediction failure. So, our first step is to do data pre processing here as we are given a set of customer details. First of all we need to list out all the steps that we will be going to do

- Check for missing values
- Feature Selection

2.1.1 Check for missing values

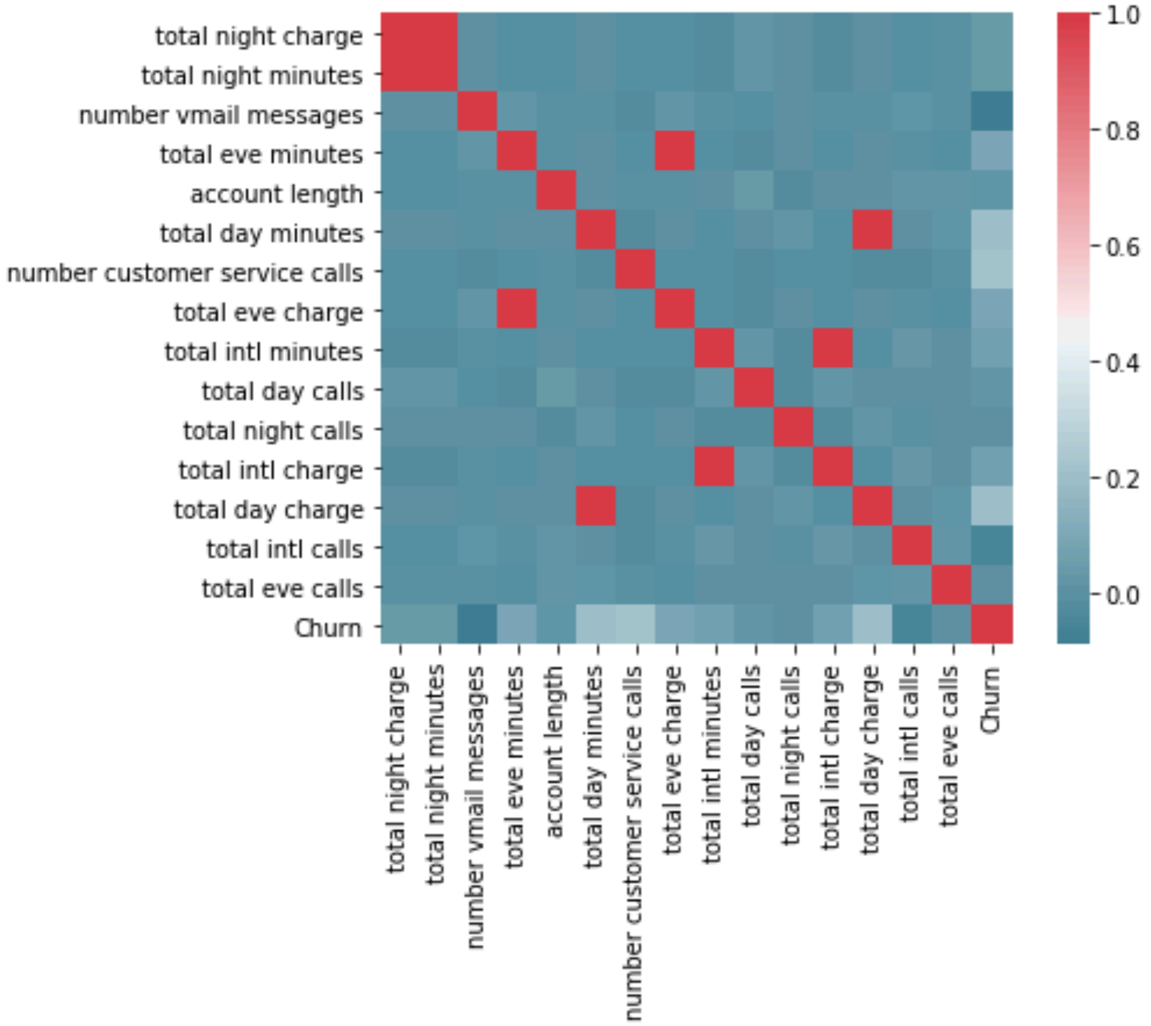
This is again a necessary step to be performed before we start our project as it also creates problem later. First of all check how much data is missing as in my case there are no fields missing in the dataset so we will go with next step of Feature Selection

2.1.2 Feature Selection

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for four reasons:

- Simplification of models to make easier to interpret by users
- Shorter the training time
- Enhance generalisation.

First of all I have separated the categorical and continuous dataset. As for categorical dataset we need to perform chi-square test and for continuous dataset I had performed co-relation analysis.



(Figure 2.1 shows the co-relation analysis of all continuous variables)

(Figure 2.2 shows the chi-square test performed on continuous variables)

```
state
0.00229622155201
area code
0.915055696024
phone number
0.491856084559
international plan
2.49310770332e-50
voice mail plan
5.15063965904e-09
```

Above results shows that area code, phone number, total intl minutes, total night minutes, total eve minutes, total day minutes have redundancy and are removed.

2.1.3 Outlier Analysis and Feature Scaling

I have tried to do these two steps also but in this project after doing these steps the accuracy and precision of the project became low so that's why I skipped that. Even in these types of project for example If we do outlier analysis on feature like no of international calls a customer had made in a month then many of the users will be having 0 or 1 but there will be very few cases when a customer will be having some relatives in foreign and he made calls then he he be taken as outlier and we can't ignore such customers. And also same case goes with number of customer service calls some customer have problems while others don't. That why I have not done outlier analysis. And for feature scaling also there were no such variables which have a great difference in their values like age is in range 10-70 which income is in range of lacks. So I haven't done scaling as well as it was also reducing my performance.

2.2 Modeling

2.2.1 Model Selection

During pre-processing we have came to understand that we have need to train our model in such a way that it should predict the target variable based on the customer details.
Now the dependent or target variables can be of following categories :

1. Nominal
2. Ordinal
3. Interval
4. Ratio

Now as we can see that in our case target variables lie under nominal category, as nominal represents classification and that's what we need. So as of now we get a clear idea that we need to go with classification methods like decision tree, random forest and logistic regression.

2.2.2 Logistic Regression

The logistic regression is a predictive analysis technique used for classification problems.
The fact is that linear regression works on a continuum of numeric estimates. In order to classify correctly, we need a more suitable measure, such as the probability of class ownership.
Thanks to the following formula, we can transform a linear regression numeric estimate into a probability that is more apt to describe how a class fits an observation:

probability of a *class = exp(r) / (1+exp(r))*

- r is the regression result (the sum of the variables weighted by the coefficients)
- exp is the exponential function.
- exp(r) corresponds to Euler's number e elevated to the power of r.
- A linear regression using such a formula (also called a link function) for transforming its results into probabilities is a logistic regression.

This is why I have used this algorithm for predicting values and then classified them accordingly.

2.2.3 Decision Tree

Decision Tree is another predictive classification algorithm which is used for classification as well as regression problem statements. But here I have used it for classification purpose. After data pre processing stage, with the help of independent variables I have first trained the model then used that to predict the target variable of test case summary of C5.0 algorithm model is :

```
Call:
C5.0.formula(formula = Churn ~ ., data = train_data, trials = 100, rules = T)

Rule-Based Model
Number of samples: 3333
Number of predictors: 14

Number of boosting iterations: 100
Average number of rules: 22.7
```

```
   (a)   (b)    <-classified as
  ----  ----
  2850          (a): class 1
    45   438    (b): class 2


Attribute usage:

100.00% state
100.00% account.length
100.00% international.plan
100.00% total.day.minutes
100.00% total.eve.minutes
100.00% total.eve.calls
100.00% total.night.minutes
100.00% total.intl.minutes
100.00% total.intl.calls
100.00% number.customer.service.calls
 99.97% total.night.calls
 99.34% total.day.calls
 99.01% voice.mail.plan
 97.75% number.vmail.messages
```

Figure 2.3 shows summary of C5.0 model in R

Some of the rules generated are in figure 2.4

```
Rule 98/10: (68.2/14.1, lift 1.3)
	voice.mail.plan = 2
	total.day.minutes > 263.4
	->  class 1  [0.785]

Rule 98/11: (237.4/59.6, lift 1.2)
	international.plan = 1
	total.day.minutes > 174.3
	total.day.minutes <= 263.4
	total.intl.minutes > 5
	number.customer.service.calls > 3
	->  class 1  [0.747]

Rule 98/12: (520.5/131.5, lift 1.2)
	international.plan = 1
	total.day.minutes <= 174.3
	total.eve.minutes > 190.7
	total.night.minutes > 171.6
	->  class 1  [0.746]

Rule 98/13: (1739/538.4, lift 1.1)
	total.day.minutes <= 263.4
	total.eve.minutes <= 324.8
	total.intl.minutes <= 13.1
	total.intl.calls > 2
	->  class 1  [0.690]
```

Figure 2.4 rules generated in decision tree.

2.2.4 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.[1][2] Random decision forests correct for decision trees' habit of overfitting to their training set.

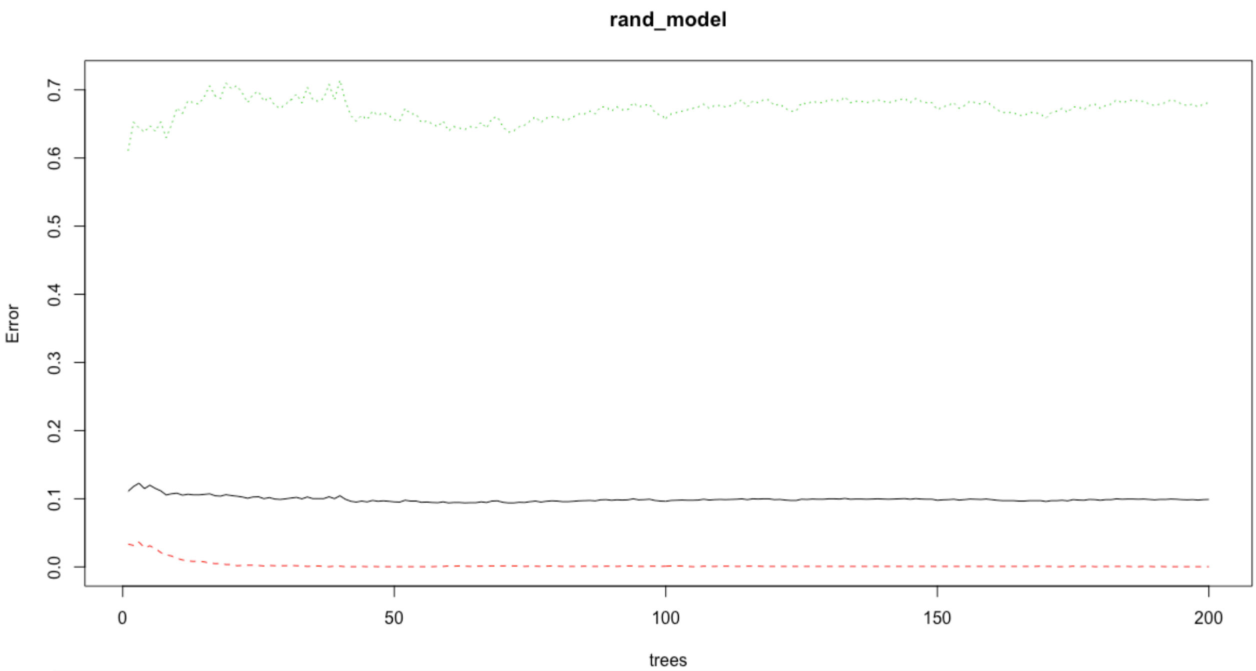
I have used this method also and get the best results. Figure 2.5 shows summary statistics of this algorithm in R.

```
Call:
 randomForest(formula = Churn ~ ., data = train_data, importance = T,      ntree = 200)
               Type of random forest: classification
                     Number of trees: 200
No. of variables tried at each split: 3

        OOB estimate of  error rate: 9.9%
Confusion matrix:
     1   2  class.error
1 2849   1 0.0003508772
2  329 154 0.6811594203
```

Figure 2.5 shows summary of random forest.

Figure 2.6 shows how many trees should be used to get efficient result. This figure shows the oob error.Out-of-bag (**OOB**) **error**, also called out-of-bag estimate, is a method of measuring the prediction **error** of **random forests**, boosted decision trees, and other machine



learning models utilizing bootstrap aggregating (bagging) to sub-sample data samples used for training.
Figure 2.6 shows OOB in random forest.

Some of the rules generated in R are :

```
[30,] "X[,2]<=142.5 & X[,6]<=159.3 & X[,13]>6.5 & X[,14]>3.5 & X[,14]<=4.5"
[31,] "X[,2]<=142.5 & X[,6]>159.3 & X[,12]<=7.15 & X[,13]>6.5 & X[,14]>3.5 & X[,14]<=4.5"
[32,] "X[,2]<=142.5 & X[,6]>159.3 & X[,12]>7.15 & X[,13]>6.5 & X[,14]>3.5 & X[,14]<=4.5"
[33,] "X[,1] %in% c('32') & X[,2]>142.5 & X[,2]<=169.5 & X[,14]>3.5 & X[,14]<=4.5"
[34,] "X[,1] %in% c('32') & X[,2]>142.5 & X[,2]>169.5 & X[,12]<=16.15 & X[,14]>3.5 & X[,14]<=4.5"
[35,] "X[,1] %in% c('32') & X[,2]>142.5 & X[,2]>169.5 & X[,12]>16.15 & X[,14]>3.5 & X[,14]<=4.5"
```

TO calculate accuracy, precision, F-1 score, Recall I have used train_data and test_data dataset in the given link below. Then for prediction of probability I have used train, test and sample dataset.

Chapter 3

Conclusion

3.1 Model Evaluation

3.1.1 Accuracy

Using sklearn library in python I have calculated the accuracy, precision score, recall score and F1 score of our model and got following results :
This result is calculated by using the training and test dataset. Training dataset is used to train the model and then target variable of test dataset is predicted with the help of the model and and then actual and predicted values are compared to see the result. The results provided below are performed on python.

Performance evaluation of Logistic Regression Result
Accuracy is 0.874625074985003
Precision is 0.7142857142857143
Recall score is 0.11160714285714286
F1 score is 0.5625413566876981

Performance evaluation of Decision Tree Classification
Accuracy is 0.9232153569286142
Precision is 0.7264150943396226
Recall score is 0.6875
F1 score is 0.8311268131770724

Performance evaluation of Random Forest Classification
Accuracy is 0.9568086382723455
Precision is 0.9691358024691358
Recall score is 0.7008928571428571
F1 score is 0.89452408236725

Here we can see that Random Forest has given best results so far.

Now we will look at the results obtained in R using above algorithms:

```
> #Logistic Regression Results
> result1$byClass[7]
       F1
0.9288564
> result1$byClass[5]
Precision
 0.968122
> result1$byClass[6]
   Recall
0.8926518
> #Decision Tree Results of C5.0
> result2$byClass[7]
      F1
0.976239
> result2$byClass[5]
Precision
 0.996535
> result2$byClass[6]
   Recall
0.9567532
> #Random Forest Results
> result3$byClass[7]
       F1
0.9501486
> result3$byClass[5]
Precision
 0.997228
> result3$byClass[6]
  Recall
0.907314
```

Figure 3.1 Results generated in R

From the figure 3.1 we can see that we get almost same results with C5.0 and random forest but C5.0 is a bit better in this case.

However we can see that logistic regression is not good for this problem as compared to Decision tree and random forest.

Appendix A - Python Code

In [37]:
```
#Import necessary libraries
import os
import pandas as pd
import numpy as np
from scipy.stats import chi2_contingency
import matplotlib as plt
import seaborn as sns
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
from sklearn.tree import export_graphviz
```

In [2]:
```
os.getcwd()
```

Out[2]: '/Users/robinredhu'

In [3]:
```
# Set the working directory
os.chdir('/Users/robinredhu/Downloads/')
```

In [8]:
```
test_data.describe()
```

Out[8]:

	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	tot
count	1667.000000	1667.000000	1667.000000	1667.000000	1667.000000	1667.000000	1667.000000	1667.000000	1667.000000	1667.000000	1667
mean	98.646671	436.369526	7.067786	181.316197	99.217157	30.824337	199.949190	100.344331	16.995903	199.431074	99.5
std	39.400755	41.890588	13.235274	52.732174	19.327148	8.964421	50.232869	19.637935	4.269758	50.437010	20.7
min	1.000000	408.000000	0.000000	6.600000	34.000000	1.120000	22.300000	38.000000	1.900000	0.000000	0.00
25%	72.000000	408.000000	0.000000	143.750000	86.000000	24.440000	165.900000	88.000000	14.100000	166.600000	86.0
50%	98.000000	415.000000	0.000000	181.000000	99.000000	30.770000	200.400000	100.000000	17.030000	199.400000	99.0
75%	126.000000	415.000000	0.000000	215.750000	112.000000	36.680000	232.300000	113.000000	19.745000	233.050000	113.
max	238.000000	510.000000	52.000000	351.500000	160.000000	59.760000	359.300000	169.000000	30.540000	381.600000	170.

2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2	5	3.
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86	6.6	7	1.
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1	3	2.

5 rows × 21 columns

In [6]:
```
test_data.head()
```

Out[6]:

	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 calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	cl
0	HI	101	510	354-8815	no	no	0	70.9	123	12.05	...	73	18.01	236.0	73	10.62	10.6	3	2.
1	MT	137	510	381-7211	no	no	0	223.6	86	38.01	...	139	20.81	94.2	81	4.24	9.5	7	2.
2	OH	103	408	411-9481	no	yes	29	294.7	95	50.10	...	105	20.17	300.3	127	13.51	13.7	6	3.
3	NM	99	415	418-9100	no	no	0	216.8	123	36.86	...	88	10.74	220.6	82	9.93	15.7	2	4.
4	SC	108	415	413-3643	no	no	0	197.4	78	33.56	...	101	10.54	204.5	107	9.20	7.7	4	2.

5 rows × 21 columns

In [9]:
```
train_data.dtypes
train_data['area code'] = train_data['area code'].astype(object)
test_data.loc[:,'area code'] = test_data.loc[:,'area code'].astype(object)
train_data.dtypes
```

Out[9]:
```
state                              object
account length                      int64
area code                          object
phone number                       object
international plan                 object
voice mail plan                    object
number vmail messages               int64
total day minutes                 float64
total day calls                     int64
total day charge                  float64
total eve minutes                 float64
total eve calls                     int64
total eve charge                  float64
total night minutes               float64
total night calls                   int64
total night charge                float64
total intl minutes                float64
total intl calls                    int64
total intl charge                 float64
number customer service calls       int64
Churn                              object
dtype: object
```

In [10]:
```
#Check for NA values for training data
missing_val_train = pd.DataFrame(train_data.isnull().sum())
missing_val_train = missing_val_train.reset_index()
#Check for NA values for testing data
missing_val_test = pd.DataFrame(test_data.isnull().sum())
missing_val_test = missing_val_test.reset_index()
```

In [11]:
```
missing_val_train
```

Out[11]:

	index	0
0	state	0
1	account length	0
2	area code	0
3	phone number	0
4	international plan	0
5	voice mail plan	0
6	number vmail messages	0
7	total day minutes	0
8	total day calls	0
9	total day charge	0
10	total eve minutes	0
11	total eve calls	0
12	total eve charge	0
13	total night minutes	0
14	total night calls	0
15	total night charge	0
16	total intl minutes	0
17	total intl calls	0
18	total intl charge	0
19	number customer service calls	0
20	Churn	0

In [12]:
```
missing_val_test
```

Out[12]:

	index	0
0	state	0
1	account length	0
2	area code	0
3	phone number	0
4	international plan	0
5	voice mail plan	0
6	number vmail messages	0
7	total day minutes	0
8	total day calls	0
9	total day charge	0
10	total eve minutes	0
11	total eve calls	0
12	total eve charge	0
13	total night minutes	0

16	total intl minutes	0
17	total intl calls	0
18	total intl charge	0
19	number customer service calls	0
20	Churn	0

In [13]:
```
#As we can see that there are no missing data next we will move to next step that is feature selection
#First of all we will didvide the dataset into categorical and numerical dataset then further
```

In [14]:
```
#This will convert the categorical variable to numbers
for i in train_data.columns:
    if(train_data[i].dtypes == 'object'):
        print(i)
        train_data.loc[:, i] = pd.Categorical(train_data.loc[:, i])
        train_data.loc[:, i] = train_data.loc[:, i].cat.codes

for i in test_data.columns:
    if(test_data[i].dtypes == 'object'):
        print(i)
        test_data.loc[:, i] = pd.Categorical(test_data.loc[:, i])
        test_data.loc[:, i] = test_data.loc[:, i].cat.codes
```

```
state
area code
phone number
international plan
voice mail plan
Churn
state
area code
phone number
international plan
voice mail plan
Churn
```

In [15]:
```
print(test_data.shape)
print(train_data.shape)
train_data.head()
test_data.head()
```

```
(1667, 21)
(3333, 21)
```

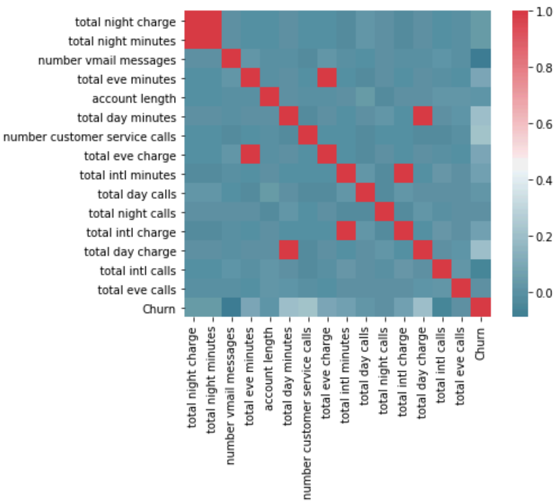
Out[15]:

	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 calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	ch
0	11	101	2	451	0	0	0	70.9	123	12.05	...	73	18.01	236.0	73	10.62	10.6	3	2.
1	26	137	2	905	0	0	0	223.6	86	38.01	...	139	20.81	94.2	81	4.24	9.5	7	2.
2	35	103	0	1467	0	1	29	294.7	95	50.10	...	105	20.17	300.3	127	13.51	13.7	6	3.
3	32	99	1	1601	0	0	0	216.8	123	36.86	...	88	10.74	220.6	82	9.93	15.7	2	4.
4	40	108	1	1501	0	0	0	197.4	78	33.56	...	101	10.54	204.5	107	9.20	7.7	4	2.

5 rows × 21 columns

In [16]:
```
#Now creating a variable to store categorical variables
cnames_cat = ['state', 'area code', 'phone number', 'international plan', 'voice mail plan']
```

In [17]:
```
#Now we will calculate the p value, degree of freedom using chi-square test.
for i in cnames_cat:
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(train_data['Churn'], train_data[i]))
    print(p)
```

In [23]:
```
cnames
```
Out[23]:
```
['total night charge',
 'number vmail messages',
 'account length',
 'international plan',
 'state',
 'number customer service calls',
 'total eve charge',
 'total day calls',
 'total night calls',
 'total intl charge',
 'total day charge',
 'total intl calls',
 'voice mail plan',
 'total eve calls',
 'Churn']
```
In [24]:
```
# Subsetting the dataset
train_data = train_data.loc[:, cnames]
test_data = test_data.loc[:, cnames]
cnames2 = list(set(list(train_data)) - set(['Churn']))
train_data.shape
```
Out[24]:
```
(3333, 15)
```
In [25]:
```
#Here we will train the classifier model
clf = tree.DecisionTreeClassifier(criterion='entropy').fit(train_data.loc[:, cnames2], train_data.loc[:,"Churn"])
```
In [26]:
```
#Here we will predict the target variable
pred = clf.predict(test_data.loc[:,cnames2])
```


In [22]:
```
# Now we can see that variables total intl minutes, total night minutes, total eve minutes, total day minutes have
# redundant information so we will remove them
cnames = list(set(list(train_data)) - set(['total intl minutes', 'total night minutes', 'total eve minutes',
                                           'total day minutes', 'area code', 'phone number']))
```
In [23]:
```
cnames
```
Out[23]:
```
['total night charge',
 'number vmail messages',
 'account length',
 'international plan',
 'state',
 'number customer service calls',
 'total eve charge',
 'total day calls',
 'total night calls',
 'total intl charge',
 'total day charge',
 'total intl calls',
 'voice mail plan',
 'total eve calls',
 'Churn']
```
In [24]:
```
# Subsetting the dataset
train_data = train_data.loc[:, cnames]
test_data = test_data.loc[:, cnames]
cnames2 = list(set(list(train_data)) - set(['Churn']))
train_data.shape
```
Out[24]:
```
(3333, 15)
```
In [25]:
```
#Here we will train the classifier model
clf = tree.DecisionTreeClassifier(criterion='entropy').fit(train_data.loc[:, cnames2], train_data.loc[:,"Churn"])
```
In [26]:
```
#Here we will predict the target variable
pred = clf.predict(test_data.loc[:,cnames2])
```

In [27]:
```
#Now we will check the results
print('Performance evaluation of {}'.format("Decision Tree Classification"))
print('Accuracy is {}'.format(accuracy_score(test_data.loc[:, "Churn"],pred)))
print('Precision is {}'.format(precision_score(test_data.loc[:, "Churn"],pred)))
print('Recall score is {}'.format(recall_score(test_data.loc[:, "Churn"],pred)))
print('F1 score is {}'.format(f1_score(test_data.loc[:, "Churn"],pred, average='macro')))
```

```
Performance evaluation of Decision Tree Classification
Accuracy is 0.9232153569286142
Precision is 0.7264150943396226
Recall score is 0.6875
F1 score is 0.8311268131770724
```

In [28]:
```
# Here we will generate the tree
dotfile = open('DeciTree.dot', 'w')
df = tree.export_graphviz(clf, out_file=dotfile, feature_names=cnames2)
```

In [29]:
```
# Now we will use random forest technique to generate the results
Rand_Model = RandomForestClassifier(n_estimators=500).fit(train_data.loc[:, cnames2], train_data.loc[:,"Churn"])
```

In [30]:
```
#Here we will predict the target variable
Rand_Prediction = Rand_Model.predict(test_data.loc[:,cnames2])
```

In [31]:
```
#Now we will check the results of random forest
print('Performance evaluation of {}'.format("Random Forest Classification"))
print('Accuracy is {}'.format(accuracy_score(test_data.loc[:, "Churn"],Rand_Prediction)))
print('Precision is {}'.format(precision_score(test_data.loc[:, "Churn"],Rand_Prediction)))
print('Recall score is {}'.format(recall_score(test_data.loc[:, "Churn"],Rand_Prediction)))
print('F1 score is {}'.format(f1_score(test_data.loc[:, "Churn"],Rand_Prediction, average='macro')))
```

```
Performance evaluation of Random Forest Classification
Accuracy is 0.9568086382723455
Precision is 0.9691358024691358
Recall score is 0.7008928571428571
F1 score is 0.89452408236725
```

In [32]:
```
# Now we will build Logistic regression model
LnModel = sm.OLS(train_data.loc[:, "Churn"], train_data.loc[:, cnames2]).fit()
```

In [33]:
```
# Now we will summarise the model
LnModel.summary()
```

Out[33]:

OLS Regression Results

Dep. Variable:	Churn	**R-squared:**	12.857
Model:	OLS	**Adj. R-squared:**	12.907
Method:	Least Squares	**F-statistic:**	-257.1
Date:	Thu, 24 May 2018	**Prob (F-statistic):**	1.00
Time:	16:21:54	**Log-Likelihood:**	-943.96
No. Observations:	3333	**AIC:**	1916.
Df Residuals:	3319	**BIC:**	2001.
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
total night charge	0.0013	0.002	0.558	0.577	-0.003	0.006
number vmail messages	0.0019	0.001	1.354	0.176	-0.001	0.005
international plan	0.3025	0.019	15.992	0.000	0.265	0.340
state	-0.0001	0.000	-0.310	0.756	-0.001	0.001
number customer service calls	0.0542	0.004	12.876	0.000	0.046	0.062
total eve charge	0.0044	0.001	3.654	0.000	0.002	0.007
total day calls	-0.0005	0.000	-1.915	0.056	-0.001	1.16e-05
total night calls	-0.0008	0.000	-3.108	0.002	-0.001	-0.000
total intl charge	0.0134	0.007	1.897	0.058	-0.000	0.027
total day charge	0.0063	0.001	10.828	0.000	0.005	0.007
total intl calls	-0.0103	0.002	-4.558	0.000	-0.015	-0.006
voice mail plan	-0.1393	0.043	-3.241	0.001	-0.224	-0.055
total eve calls	-0.0007	0.000	-2.845	0.004	-0.001	-0.000
account length	-8.968e-05	0.000	-0.651	0.515	-0.000	0.000

Omnibus:	869.471	**Durbin-Watson:**	1.966
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1769.136
Skew:	1.567	**Prob(JB):**	0.00
Kurtosis:	4.708	**Cond. No.**	1.60e+03

In [34]:
```
# Now we will predict the values
prediction = LnModel.predict(test_data.loc[:,cnames2])
```

In [35]:
```
# Here we will round off the values
prediction = np.round(prediction)
```

In [36]:
```
#Now we will check the results of Linear regression model
print('Performance evaluation of {}'.format("Logistic Regression Result"))
print('Accuracy is {}'.format(accuracy_score(test_data.loc[:, "Churn"],prediction)))
print('Precision is {}'.format(precision_score(test_data.loc[:, "Churn"],prediction)))
print('Recall score is {}'.format(recall_score(test_data.loc[:, "Churn"],prediction)))
print('F1 score is {}'.format(f1_score(test_data.loc[:, "Churn"],prediction, average='macro')))
```

```
Performance evaluation of Logistic Regression Result
Accuracy is 0.874625074985003
Precision is 0.7142857142857143
Recall score is 0.11160714285714286
F1 score is 0.5625413566876981
```

Appendix B - R Code

```
#Project 2
#Import the necessary libraries
library(caret)
library(inTrees)
library(C50)
library(randomForest)
library(dplyr)
library(corrgram)
library(MASS)
library(party)
library(DMwR)

#Import the dataset
train_data = read.csv('../robinredhu/Downloads/
Train_data.csv')
test_data = read.csv('../robinredhu/Downloads/
Test_data.csv')

#Categorising data
View(train_data)
summary(train_data)
glimpse(train_data)
View(test_data)
summary(test_data)
glimpse(test_data)
#Converting area code to factor variable
train_data$area.code = as.factor(train_data$area.code)
test_data$area.code = as.factor(test_data$area.code)

#Missing value analysis
sum(is.na(train_data))
sum(is.na(test_data))

#Feature Selection
#Now for this first of all we need to seperate out
numerical and categorical variables
```

```
num_index = which(unlist(sapply(train_data, function(x)
(class(x) == 'numeric'||class(x) == 'integer'))))
train_num = train_data[,num_index]
train_cat = train_data[,-num_index]
#Correlation matrix will help to find out and remove
multi collinearity
corrgram(train_num, order = F, upper.panel = panel.pie,
text.panel = panel.txt, main = 'Corelation Matrix')
# From the results obtained we will remove
total.day.charge, total.eve.charge, total.night.charge,
total.intl.charge variables

#Chi-Sqr test will help to remove unnecessary categorical
variables
for(i in 1:ncol(train_cat)){
  print(names(train_cat[i]))
  print(chisq.test(table(train_cat$Churn,
train_cat[,i])))
}
# From the results obtained we will remove  area.code,
phone.number variables
# c('area.code', 'phone.number', 'total.day.charge',
'total.eve.charge', 'total.night.charge',
'total.intl.charge')
train_data = train_data[,-c(3,4,10,13,16,19)]
test_data = test_data[,-c(3,4,10,13,16,19)]

#Convert all string variables to numeric
for(i in 1:ncol(train_data)){
  if(class(train_data[,i]) == 'factor'){
    train_data[,i] = factor(train_data[,i], labels =
1:length(levels(factor(train_data[,i]))))
  }
}
for(i in 1:ncol(test_data)){
  if(class(test_data[,i]) == 'factor'){
    test_data[,i] = factor(test_data[,i], labels =
1:length(levels(factor(test_data[,i]))))
  }
}
```

```
#After pre processing It's time to train the model
#Train model usign logistic regression
logistic_model = glm(Churn~., data = train_data, family =
'binomial')
summary(logistic_model)

prediction = predict(logistic_model, test_data[,-15],
type ='response')
prediction = ifelse(prediction>=0.5,1,0)

#Confusion matrix
table(test_data$Churn, prediction)
accuracy_log = ((1397+56)/nrow(test_data))*100
result1 = confusionMatrix(table(test_data$Churn,
prediction))
#Logistic Regression Results
#To calculate F1 score
result1$byClass[7]
#Precision score
result1$byClass[5]
#Recall score
result1$byClass[6]

#Using Decision Tree
tree2 = C5.0(Churn~., train_data, trials = 100, rules =T)
c5_prediction = predict(tree2, test_data[,-15], type =
'class')
summary(tree2)
#Confusion matrix
table(test_data$Churn, c5_prediction)
accuracy_c5.0 = ((1438+159)/nrow(test_data))*100
result2 = confusionMatrix(table(test_data$Churn,
c5_prediction))
#Decision Tree Results of C5.0
#To calculate F1 score
result2$byClass[7]
#Precision score
result2$byClass[5]
#Recall score
result2$byClass[6]

#Now we will try for random forest
```

```
rand_model = randomForest(Churn~., train_data, importance
= T, ntree =200)
rand_model
#Extract rules from trees
treeList = RF2List(rand_model)
rules = extractRules(treeList, X = test_data[, -15])
rules
rf_prediction = predict(rand_model, test_data[, -15])
#Confusion matrix
table(test_data$Churn, rf_prediction)
accuracy_rf = ((1436+67)/nrow(test_data))*100
plot(rand_model)
result3 =confusionMatrix(table(test_data$Churn,
rf_prediction))
#Random Forest Results
#To calculate F1 score
result3$byClass[7]
#Precision score
result3$byClass[5]
#Recall score
result3$byClass[6]
```

Github Link For Project :

https://github.com/robinredhu/Churn_Prediction.git