Data Science PROJECT

Client: No-Churn Telecom | Category: Telecom – Churn Rate ML

Project Ref: PM-PR-0017

Business Case:

No-Churn Telecom is an established Telecom operator in Europe with more than a decade in Business. Due to new players in the market, telecom industry has become very competitive and retaining customers becoming a challenge.

In spite of No-Churn initiatives of reducing tariffs and promoting more offers, the churn rate (percentage of customers migrating to competitors) is well above 10%.

No-Churn wants to explore possibility of Machine Learning to help with following use cases to retain competitive edge in the industry.

Project Goal

Help No-Churn with their use cases with ML

1. Understanding the variables that are influencing the customers to migrate.

2. Creating Churn risk scores that can be indicative to drive retention campaigns.

3. Introduce new predicting variable “CHURN-FLAG” with values YES(1) or NO(0) so that email campaigns with lucrative offers can be targeted to Churn YES customers.

4. Exporting the trained model with prediction capability for CHURN-FLAG, which can be highlighted in service applications to serve the customer better.
help to identify possible CHURN-FLAG YES customers and provide more attention in customer touch point areas, including customer care support, request fulfilment, auto categorizing tickets as high priority for quick resolutions any questions they may have etc.,

Data

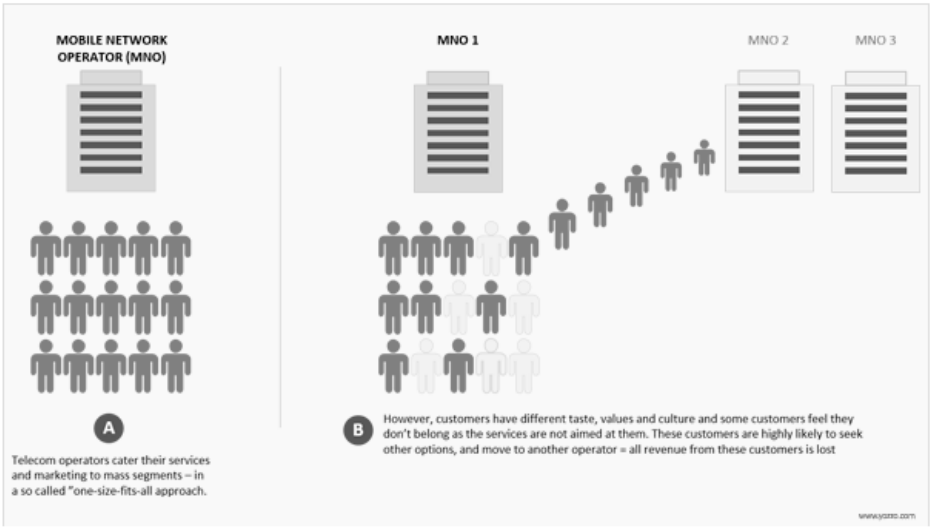
DataBase Details:
SQL database
DB Name: project_telecom
Table Name: telecom_churn_data
Host: 18.136.56.185
Username: dm_team3
Password: dm_team15119#

Meta Info of Data
State : Name of the state
Account Length : How long the account of customer has been active
Area Code : std code of a particular area
Phone : Telephone number of the customer
International Plan : International roaming pack activated by the customer
VMail Plan : Voice Mail plan activated by the customer
VMail Message : Total number of voice mail messages
Day Mins : Total minutes spent on the call during the day time
Day Calls : Total calls done in the day time
Day Charge : Total charge in day
Eve Mins : Total minutes spent on the call during the evening time
Eve Calls : Total calls done in the evening time
Eve Charge : Total charge in evening
Night Mins : Total minutes spent on the call during the night
Night Calls : Total calls done in the night time
Night Charge : Total charge in night
International Mins : Total minutes spent on the international call
International calls : Total calls done to a people at international location
International Charge : Total international charge
CustServ Calls : Number of customer service calls made
Churn : the annual percentage rate at which customers stop subscribing to a service

We're going to go through a machine learning project with the goal of predicting how many customers churned in telecom industry.

1. Telecom Churn Description

Churn rate or attrition rate is a measure of the number of individuals or items moving out of a collective group over a specific period. It is one of two primary factors that determine the steady-state level of customers a business will support. Churn rate is an input into customer lifetime value modelling and part of a simulator used to measure return on marketing investment using marketing mix modelling https://en.wikipedia.org/wiki/Churn_rate



2. Features

The features of the datasets were provided by DataMites company.

3. Assumptions

1. Dropped the column phone as it is not required for predicting churn
2. Created new fields like Per_Day_Call, Per_Day_Mins, Per_Day_Charge based on Day,Evening and Night calls, minutes and charge.
3. Used Churn as a target variable.
4. Created a new variable called Churn Flag based on Churn: True(1) , False(0)
5. Removed outliers for the fields like

   'VMail_Message','Day_Calls','Day_Mins','Day_Charge',
   'Eve_Calls','Eve_Mins','Eve_Charge','Area_Code','Acount_Length',Night_Calls','
   Night_Mins','Night_Charge','International_Calls','International_Mins','Internatio
   nal_Charge','CustServ_Calls'

Steps:

1. **Import the necessary packages**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
%matplotlib inline
from collections import Counter
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder,scale
from sklearn.metrics import
accuracy_score,precision_score,confusion_matrix,classification_report,f1_score,recall_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")
from matplotlib import pyplot
```

2. **Load the dataset**

```
data=pd.read_excel("telecom_churn_data.xlsx")
data
```

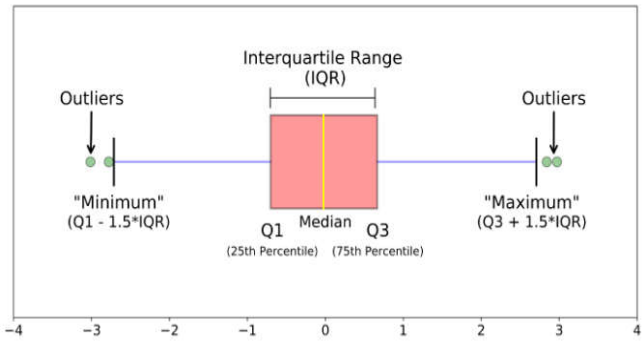
3. **Perform Exploratory Data Analysis (EDA)**

```
data.index
data.columns
data.drop(labels='Unnamed: 0',axis=1,inplace=True)
data.head()

Rename the columns
data.rename(columns={'columns1':'State',
        'columns2':'Account_Length',
        'columns3':'Area_Code',
        'columns4':'Phone',
        'columns5':'International_Plan',
        'columns6':'VMail_Plan',
        'columns7':'VMail_Message',
        'columns8':'Day_Mins',
        'columns9':'Day_Calls',
        'columns10':'Day_Charge',
        'columns11':'Eve_Mins',
```

```
        'columns12':'Eve_Calls',
        'columns13':'Eve_Charge',
        'columns14':'Night_Mins',
        'columns15':'Night_Calls',
        'columns16':'Night_Charge',
        'columns17':'International_Mins',
        'columns18':'International_Calls',
        'columns19':'International_Charge',
        'columns20':'CustServ_Calls',
        'columns21':'Churn'},inplace=True)

data.head()
data.dtypes
data.info()
data.describe()
data.shape
data.duplicated().sum()
pd.get_dummies(data.State,drop_first=False)
pd.get_dummies(data.International_Plan,drop_first=False)
pd.get_dummies(data.VMail_Plan,drop_first=False)
pd.get_dummies(data.Churn,drop_first=False)
Counter(data.State)
Counter(data.Account_Length)
Counter(data.Area_Code)
Counter(data.International_Plan)
Counter(data.VMail_Plan)
Counter(data.VMail_Message)
Counter(data.Day_Mins)
Counter(data.Day_Calls)
Counter(data.Day_Charge)
Counter(data.Eve_Mins)
Counter(data.Eve_Calls)
Counter(data.Eve_Charge)
Counter(data.Night_Mins)
Counter(data.Night_Calls)
Counter(data.Night_Charge)
Counter(data.International_Mins)
Counter(data.International_Calls)
Counter(data.International_Charge)
Counter(data.Churn)
```

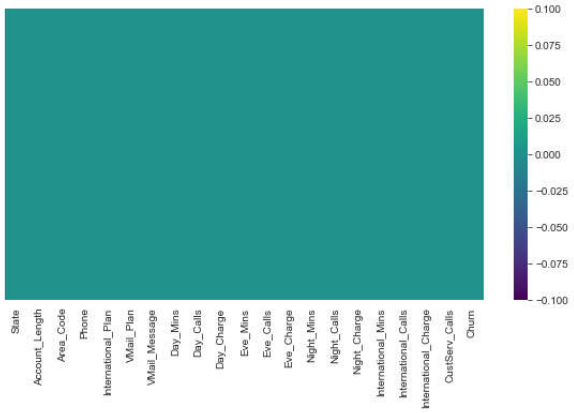
4. Checking for the outliers

An outlier is an object that deviates significantly from the rest of the objects. They can be caused by measurement or execution error. The analysis of outlier data is referred to as outlier analysis or outlier mining.

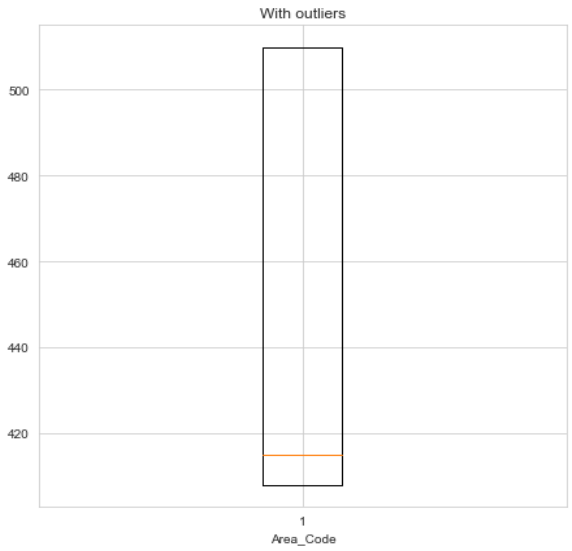


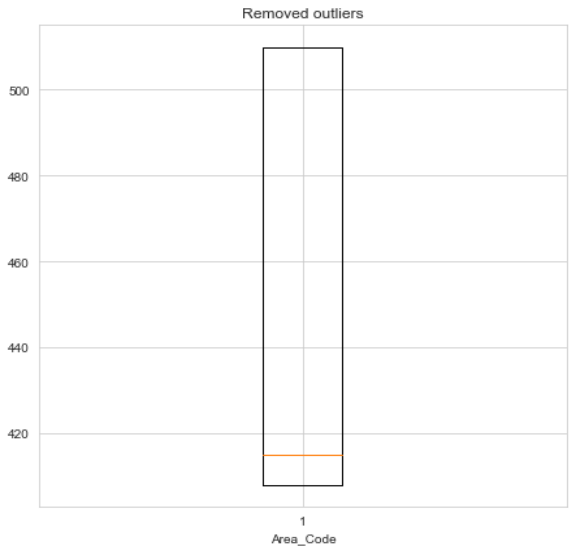
Boxplots are a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum").

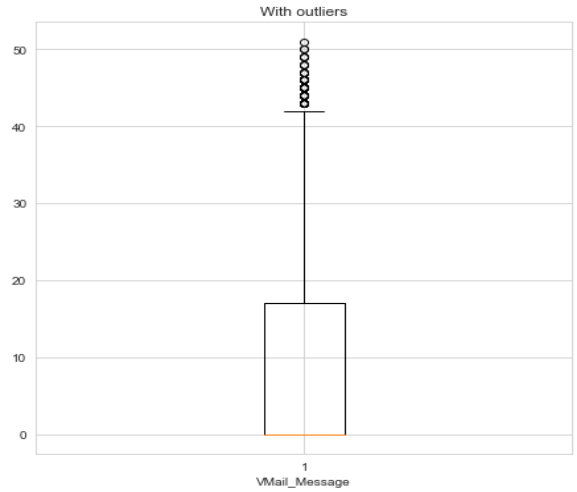
Heatmap representing the dataset.

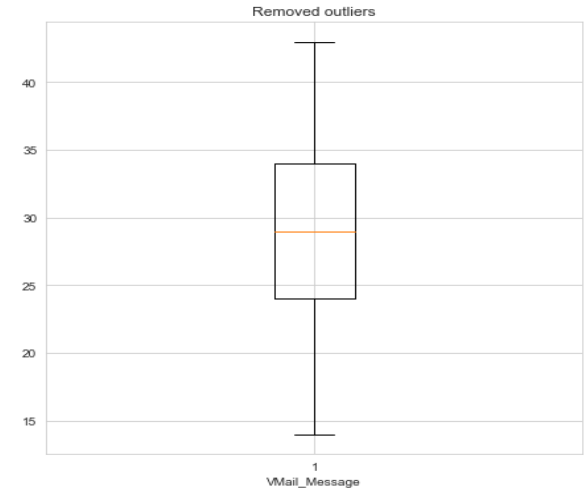


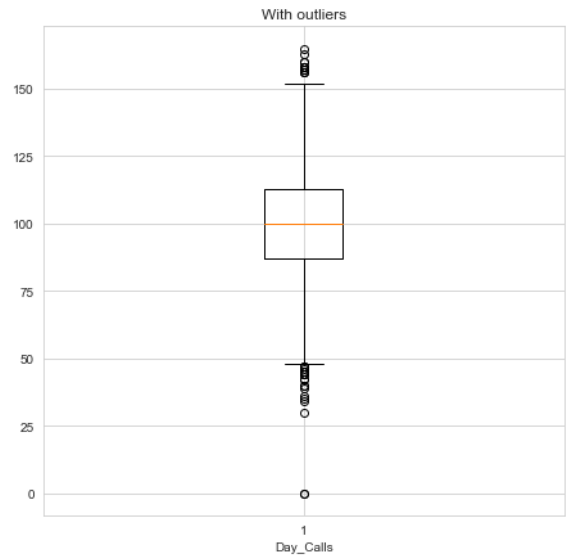
Boxplots to detect and remove the outliers

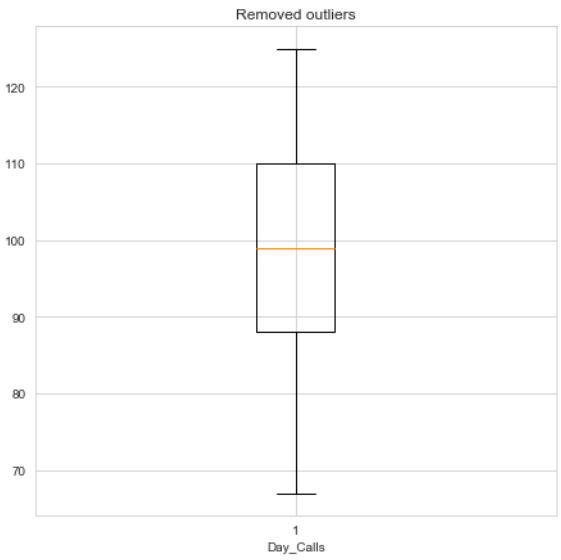

With outliers
500
480
460
440
420
1
Area_Code

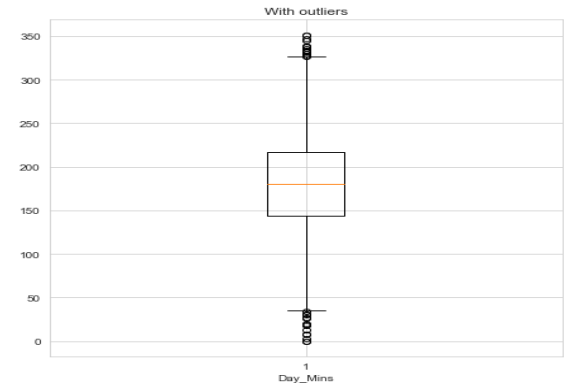

Removed outliers
500
480
460
440
420
1
Area_Code


With outliers
50
40
30
20
10
0
1
VMail_Message

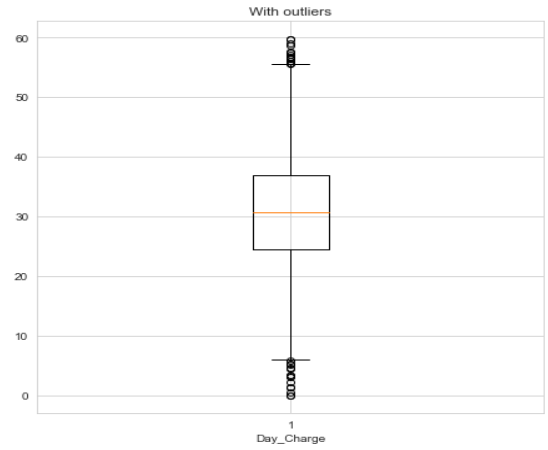

Removed outliers
40
35
30
25
20
15
1
VMail_Message

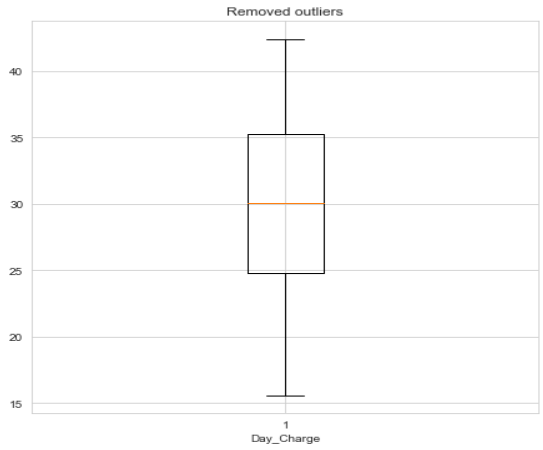

With outliers
150
125
100
75
50
25
0
1
Day_Calls

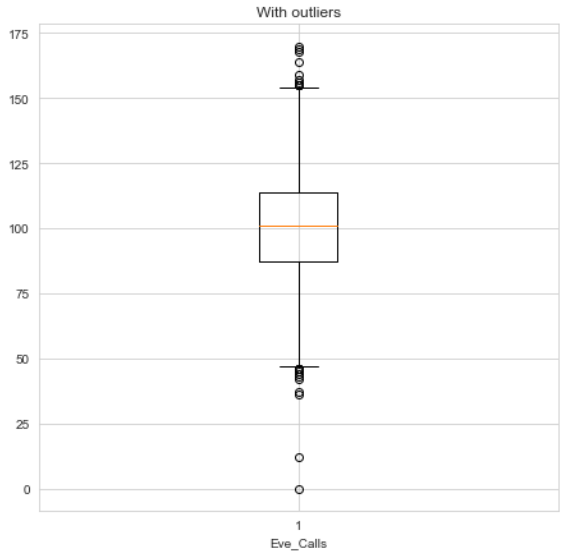

Removed outliers
120
110
100
90
80
70
1
Day_Calls

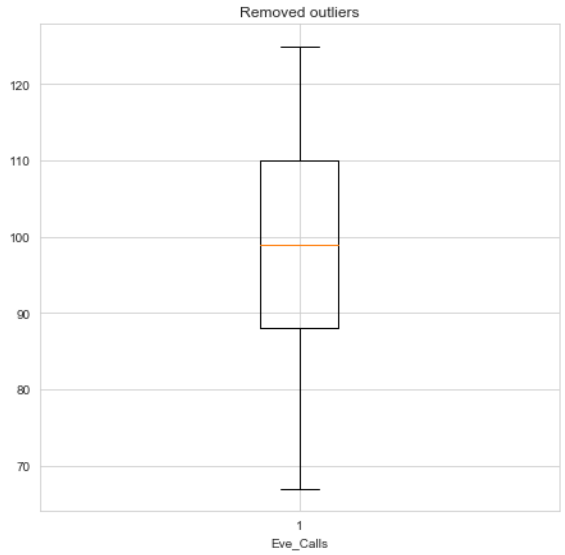

With outliers
350
300
250
200
150
100
50
0
1
Day_Mins

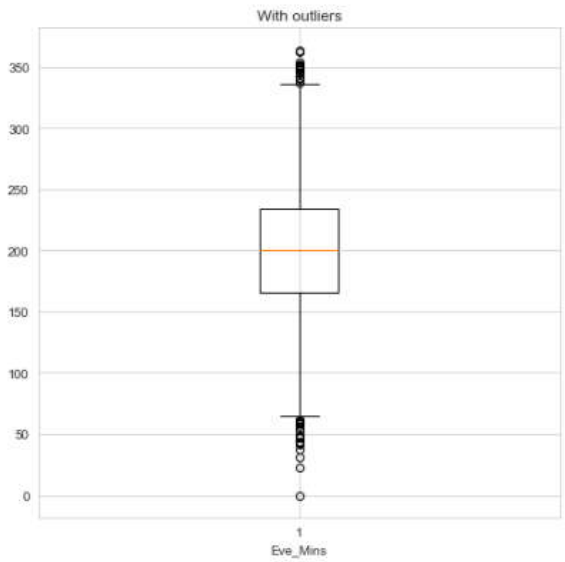

Removed outliers
240
220
200
180
160
140
120
100
1
Day_Mins

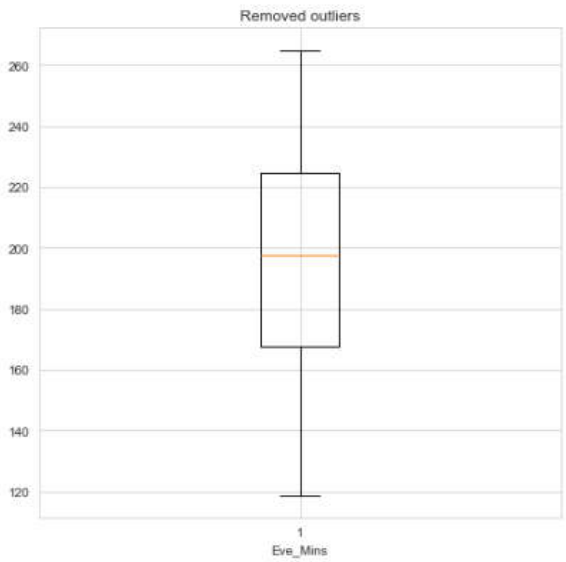

With outliers
60
50
40
30
20
10
0
1
Day_Charge

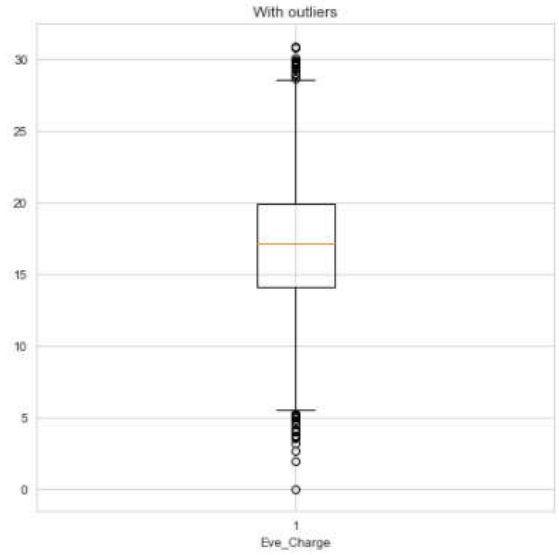

Removed outliers
40
35
30
25
20
15
1
Day_Charge

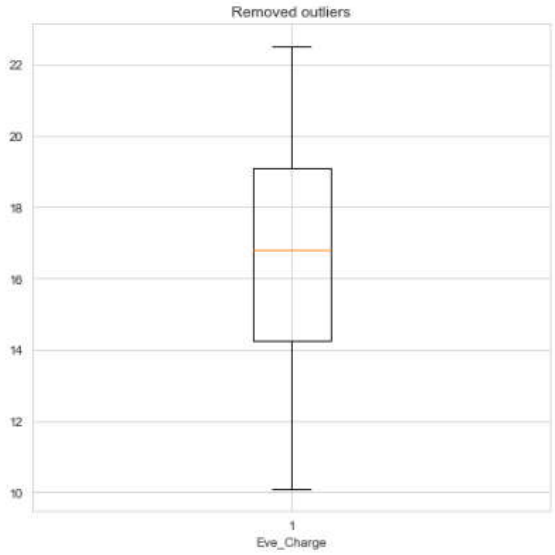

With outliers
175
150
125
100
75
50
25
0
1
Eve_Calls

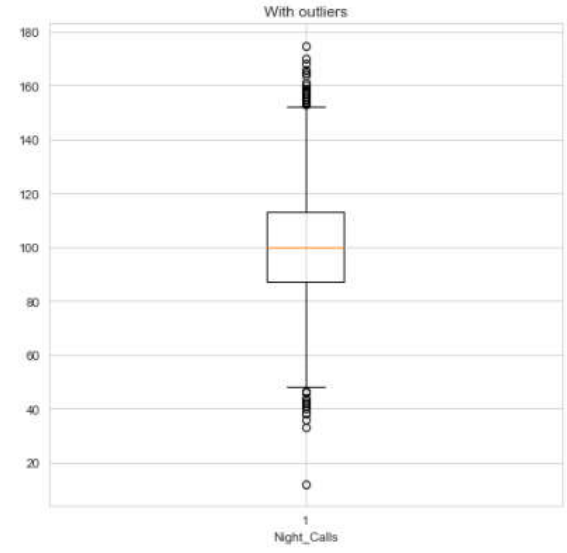

Removed outliers
120
110
100
90
80
70
1
Eve_Calls

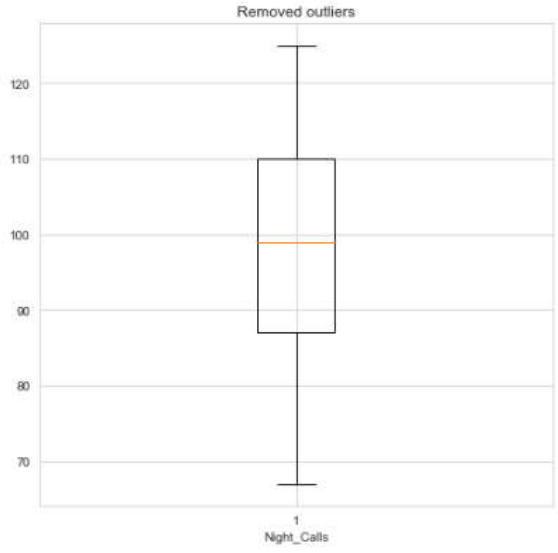

With outliers
350
300
250
200
150
100
50
0
1
Eve_Mins

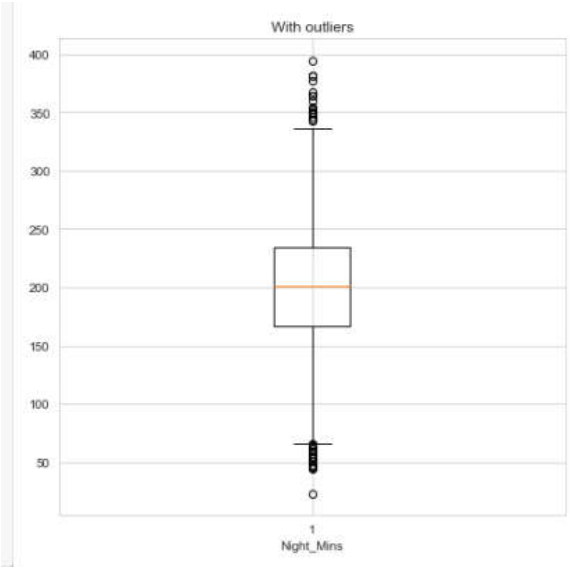

Removed outliers
260
240
220
200
180
160
140
120
1
Eve_Mins

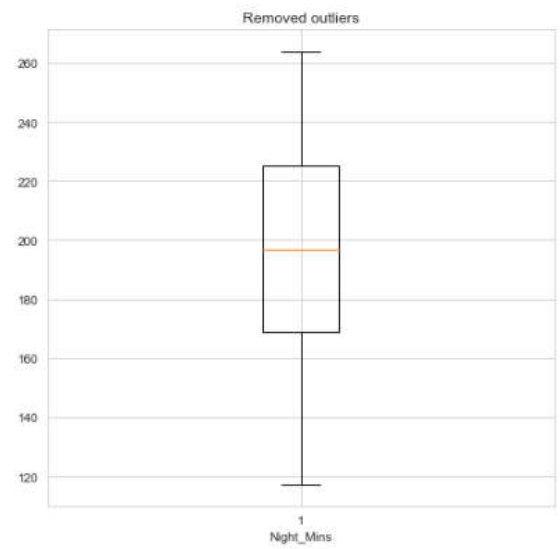

With outliers
30
25
20
15
10
5
0
1
Eve_Charge

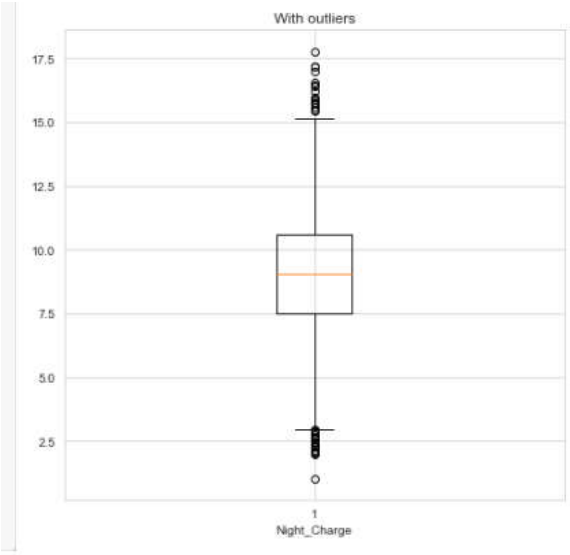

Removed outliers
22
20
18
16
14
12
10
1
Eve_Charge

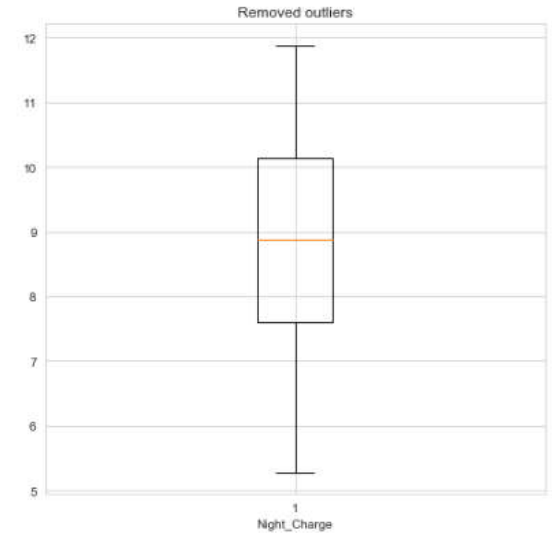

With outliers
180
160
140
120
100
80
60
40
20
1
Night_Calls

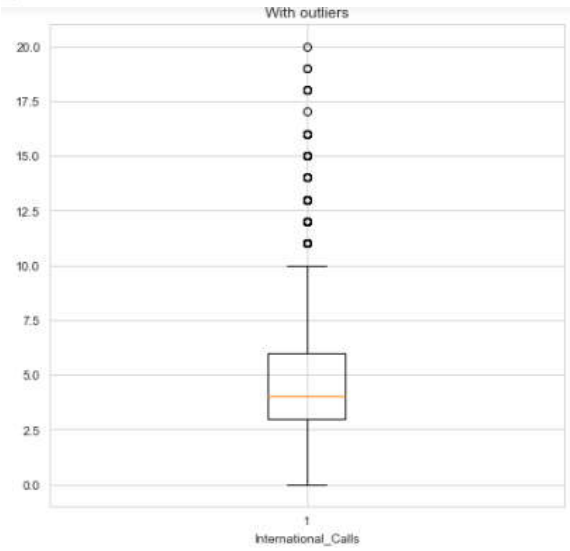

Removed outliers
120
110
100
90
80
70
1
Night_Calls

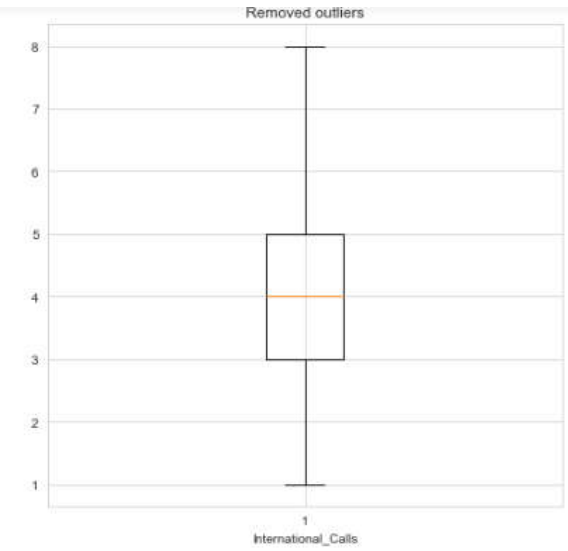

With outliers
400
350
300
250
200
150
100
50
1
Night_Mins

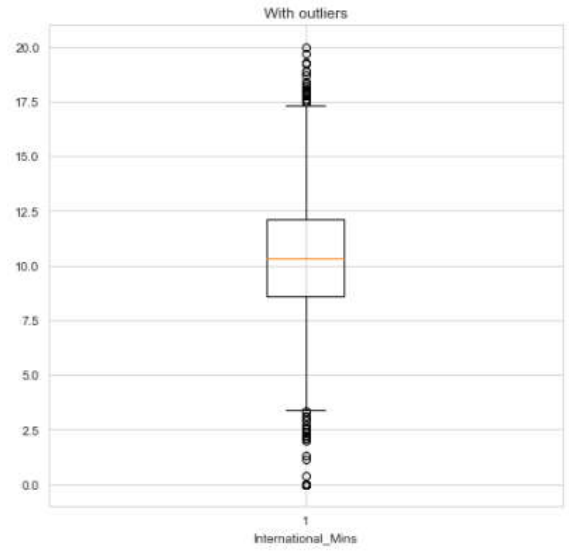

Removed outliers
260
240
220
200
180
160
140
120
1
Night_Mins

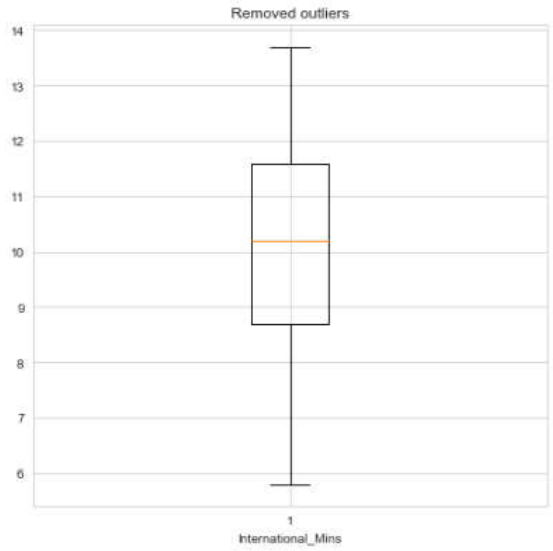

With outliers
17.5
15.0
12.5
10.0
7.5
5.0
2.5
1
Night_Charge

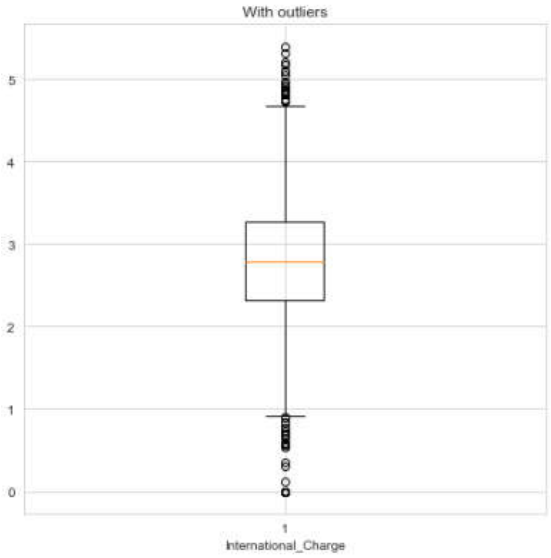

Removed outliers
12
11
10
9
8
7
6
5
1
Night_Charge

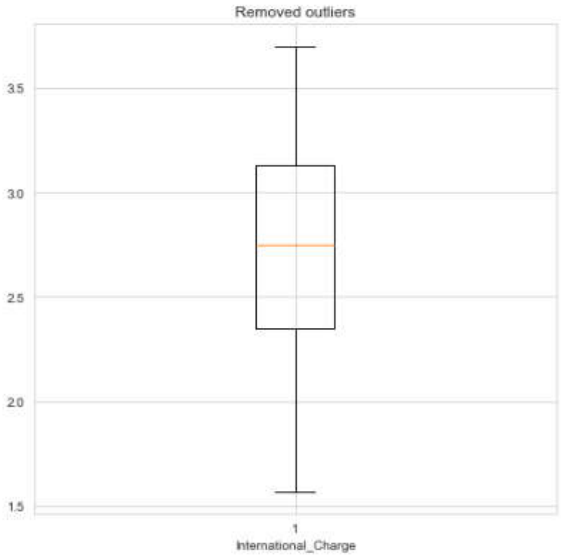

With outliers
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1
International_Calls

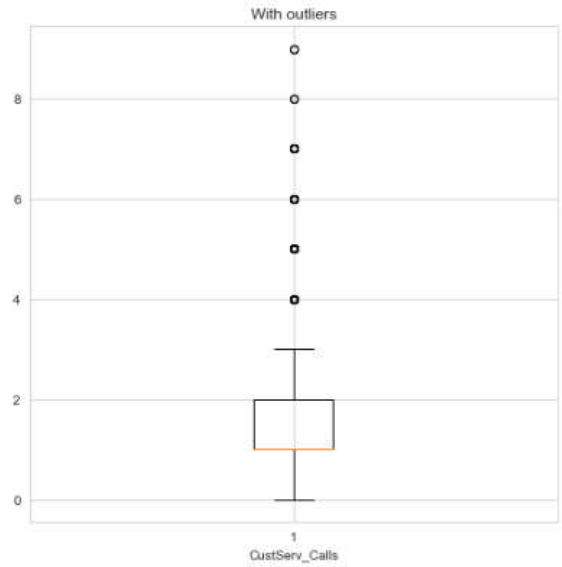

Removed outliers
8
7
6
5
4
3
2
1
1
International_Calls

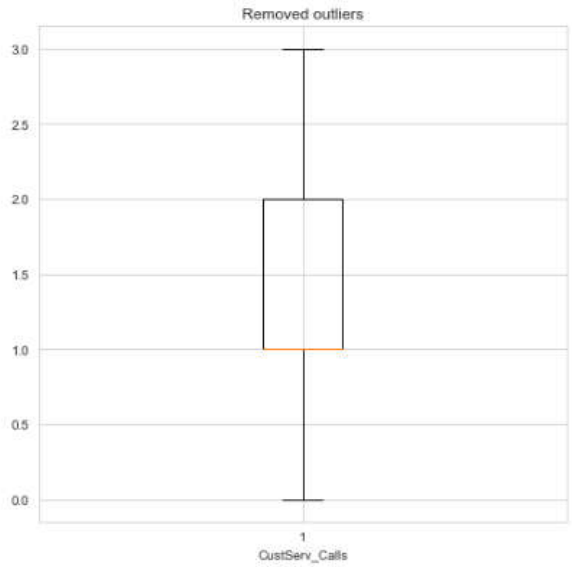

With outliers
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1
International_Mins


Removed outliers
14
13
12
11
10
9
8
7
6
1
International_Mins


With outliers
5
4
3
2
1
0
1
International_Charge


Removed outliers
3.5
3.0
2.5
2.0
1.5
1
International_Charge


With outliers
8
6
4
2
0
1
CustServ_Calls


Removed outliers
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
CustServ_Calls

5. Using Label Encoding

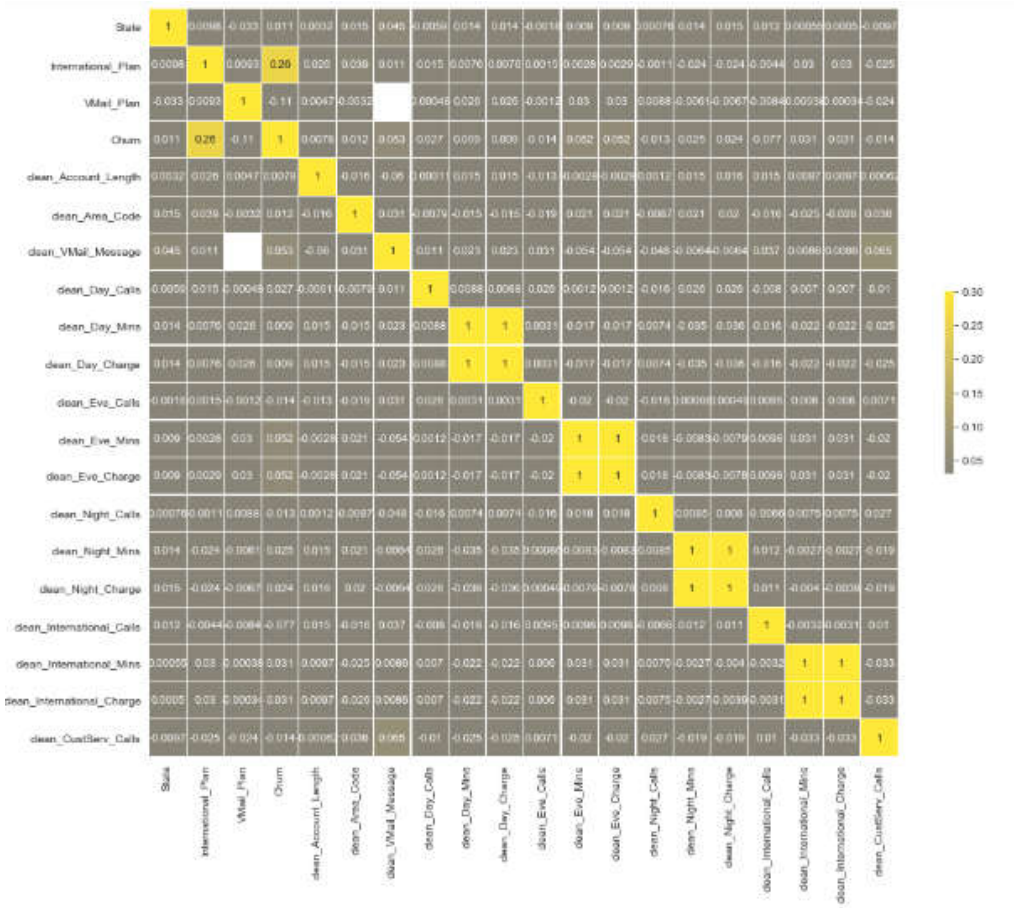
Perform label encoding for the following fields:
List=['International_Plan','State','VMail_Plan','Churn']
for i in List:
  data[[i]]=enc.fit_transform(data[[i]])

Drop Phone field
data.drop('Phone',inplace=True,axis=1)
data.info()

6. Drop the following fields

data.drop(['Account_Length','Area_Code'],inplace=True,axis=1)
data.drop(['VMail_Message','Day_Calls','Day_Mins','Day_Charge',
  'Eve_Calls','Eve_Mins','Eve_Charge',
  'Night_Calls','Night_Mins','Night_Charge',
  'International_Calls','International_Mins','International_Charge',
  'CustServ_Calls'],inplace=True,axis=1)

7. Correlation Matrix



Some of the results of feature selection

```
VMail_Plan                    -0.033323
clean_CustServ_Calls          -0.009709
clean_Day_Calls               -0.005927
clean_Eve_Calls               -0.001811
clean_International_Charge     0.000502
clean_International_Mins       0.000550
clean_Night_Calls              0.000756
clean_Account_Length           0.003198
clean_Eve_Charge               0.008984
clean_Eve_Mins                 0.008995
International_Plan             0.009789
Churn                          0.010979
clean_International_Calls      0.011528
clean_Night_Mins               0.014336
clean_Day_Charge               0.014496
clean_Day_Mins                 0.014497
clean_Night_Charge             0.015092
clean_Area_Code                0.015370
clean_VMail_Message            0.044921
State                          1.000000
Name: State, dtype: float64
```

```
clean_VMail_Message           -0.059679
clean_Area_Code               -0.016211
clean_Eve_Calls               -0.013305
clean_Eve_Mins                -0.002783
clean_Eve_Charge              -0.002771
clean_CustServ_Calls          -0.000624
clean_Day_Calls               -0.000107
clean_Night_Calls              0.001217
State                          0.003198
VMail_Plan                     0.004693
Churn                          0.007805
clean_International_Mins       0.009651
clean_International_Charge     0.009675
clean_International_Calls      0.014886
clean_Day_Mins                 0.015032
clean_Day_Charge               0.015033
clean_Night_Mins               0.015249
clean_Night_Charge             0.015958
International_Plan             0.025634
clean_Account_Length           1.000000
Name: clean_Account_Length, dtype: float64
```

```
clean_International_Charge    -0.025682
clean_International_Mins      -0.025489
clean_Eve_Calls               -0.018948
clean_Account_Length          -0.016211
clean_International_Calls     -0.015909
clean_Day_Charge              -0.015041
clean_Day_Mins                -0.015036
clean_Night_Calls             -0.008748
clean_Day_Calls               -0.007911
VMail_Plan                    -0.003150
Churn                          0.011995
State                          0.015370
clean_Night_Charge             0.020326
clean_Eve_Mins                 0.020696
clean_Eve_Charge               0.020725
clean_Night_Mins               0.020864
clean_VMail_Message            0.030756
clean_CustServ_Calls           0.036070
International_Plan             0.039263
clean_Area_Code                1.000000
Name: clean_Area_Code, dtype: float64
```

```
clean_CustServ_Calls          -0.025153
clean_Night_Charge            -0.024266
clean_Night_Mins              -0.023966
clean_International_Calls     -0.004410
clean_Night_Calls             -0.001136
clean_Eve_Calls                0.001465
clean_Eve_Mins                 0.002844
clean_Eve_Charge               0.002852
clean_Day_Charge               0.007637
clean_Day_Mins                 0.007638
VMail_Plan                     0.009278
State                          0.009789
clean_VMail_Message            0.011464
clean_Day_Calls                0.014840
clean_Account_Length           0.025634
clean_International_Charge     0.030349
clean_International_Mins       0.030384
clean_Area_Code                0.039263
Churn                          0.257524
International_Plan             1.000000
Name: International_Plan, dtype: float64
```

```
 Churn                         -0.110149
 State                         -0.033323
 clean_CustServ_Calls          -0.023809
 clean_International_Calls     -0.008385
 clean_Night_Charge            -0.006689
 clean_Night_Mins              -0.006104
 clean_Area_Code               -0.003150
 clean_Eve_Calls               -0.001244
 clean_Day_Calls               -0.000478
 clean_International_Mins      -0.000379
 clean_International_Charge    -0.000345
 clean_Account_Length           0.004693
 clean_Night_Calls              0.008817
 International_Plan             0.009278
 clean_Day_Mins                 0.026290
 clean_Day_Charge               0.026292
 clean_Eve_Charge               0.030346
 clean_Eve_Mins                 0.030346
 VMail_Plan                     1.000000
 clean_VMail_Message                 NaN
 Name: VMail_Plan, dtype: float64
```

```
clean_Account_Length          -0.059679
clean_Eve_Mins                -0.054271
clean_Eve_Charge              -0.054252
clean_Night_Calls             -0.048144
clean_Night_Mins              -0.006424
clean_Night_Charge            -0.006415
clean_International_Charge     0.008620
clean_International_Mins       0.008621
clean_Day_Calls                0.010764
International_Plan             0.011464
clean_Day_Charge               0.023079
clean_Day_Mins                 0.023090
clean_Area_Code                0.030756
clean_Eve_Calls                0.031306
clean_International_Calls      0.037088
State                          0.044921
Churn                          0.053259
clean_CustServ_Calls           0.065180
clean_VMail_Message            1.000000
VMail_Plan                          NaN
Name: clean_VMail_Message, dtype: float64
```

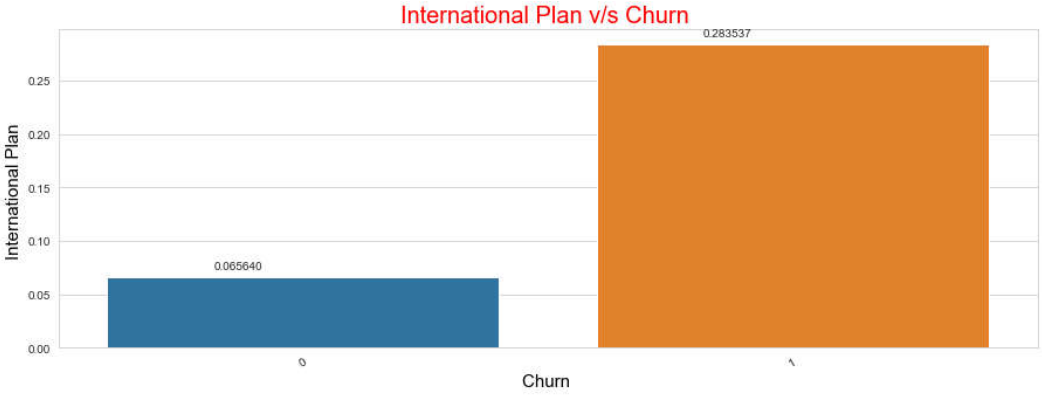
```
clean_International_Charge   -0.032910
clean_International_Mins     -0.032802
International_Plan           -0.025153
clean_Day_Charge             -0.025045
clean_Day_Mins               -0.025036
VMail_Plan                   -0.023809
clean_Eve_Charge             -0.019988
clean_Eve_Mins               -0.019966
clean_Night_Charge           -0.019476
clean_Night_Mins             -0.018637
Churn                        -0.014492
clean_Day_Calls              -0.010029
State                        -0.009709
clean_Account_Length         -0.000624
clean_Eve_Calls               0.007097
clean_International_Calls     0.010119
clean_Night_Calls             0.026711
clean_Area_Code               0.036070
clean_VMail_Message           0.065180
clean_CustServ_Calls          1.000000
Name: clean_CustServ_Calls, dtype: float64
```

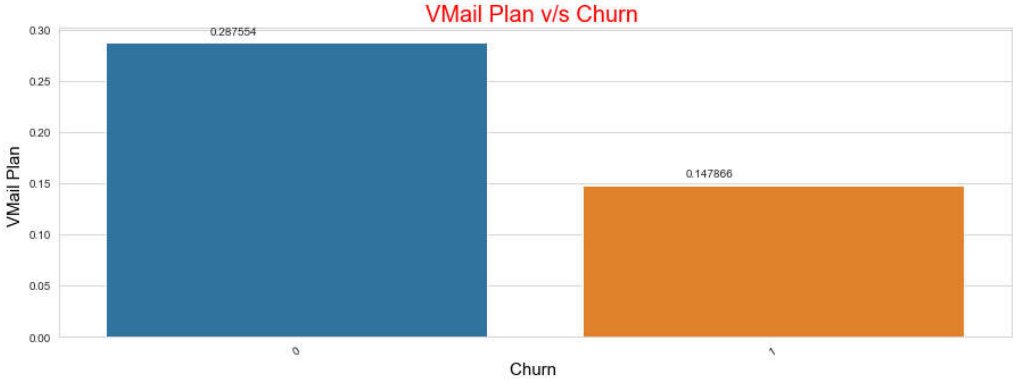
```
 VMail_Plan                  -0.110149
 clean_International_Calls   -0.076634
 clean_CustServ_Calls        -0.014492
 clean_Eve_Calls             -0.013830
 clean_Night_Calls           -0.012660
 clean_Account_Length         0.007805
 clean_Day_Charge             0.009013
 clean_Day_Mins               0.009026
 State                        0.010979
 clean_Area_Code              0.011995
 clean_Night_Charge           0.024099
 clean_Night_Mins             0.024500
 clean_Day_Calls              0.026988
 clean_International_Charge   0.031458
 clean_International_Mins     0.031475
 clean_Eve_Charge             0.051802
 clean_Eve_Mins               0.051814
 clean_VMail_Message          0.053259
 International_Plan           0.257524
 Churn                        1.000000
 Name: Churn, dtype: float64
```

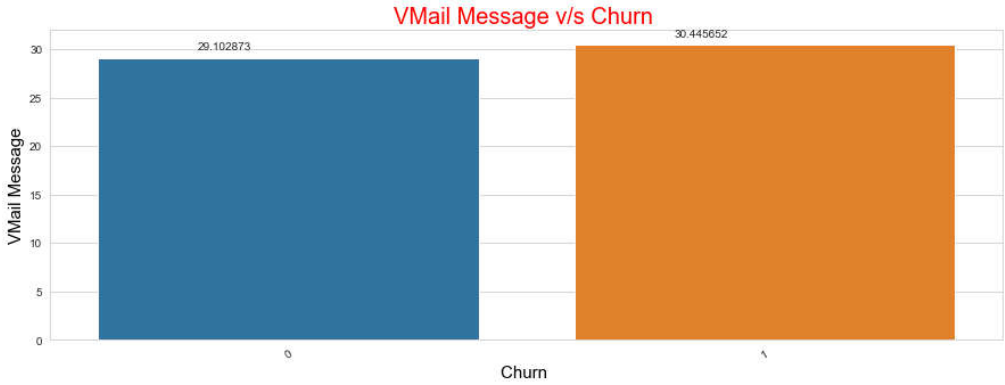
8. Data Exploration Insights

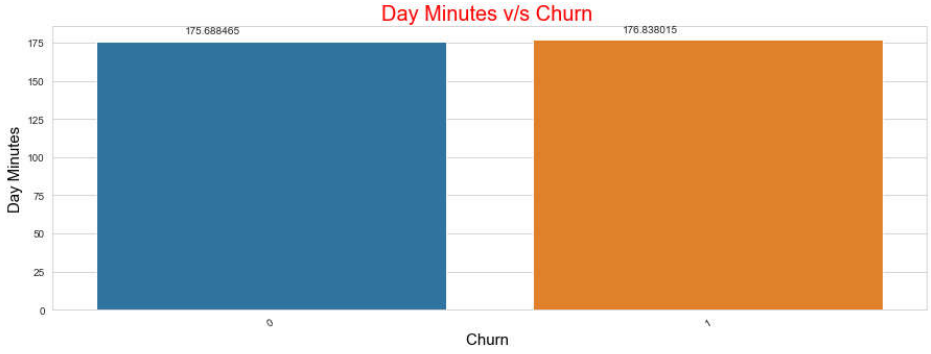
1. Understanding the variables that are influencing the customers to migrate.

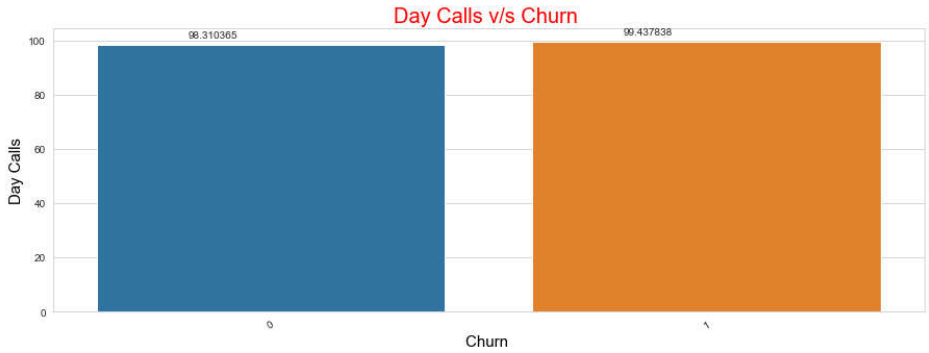

Area code v/s Churn
436.839939
438.292683
Area code count
Churn

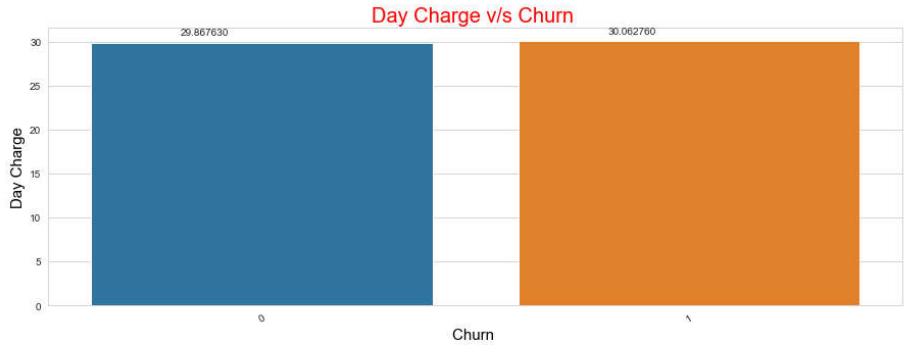

International Plan v/s Churn
0.283537
0.065640
International Plan
Churn

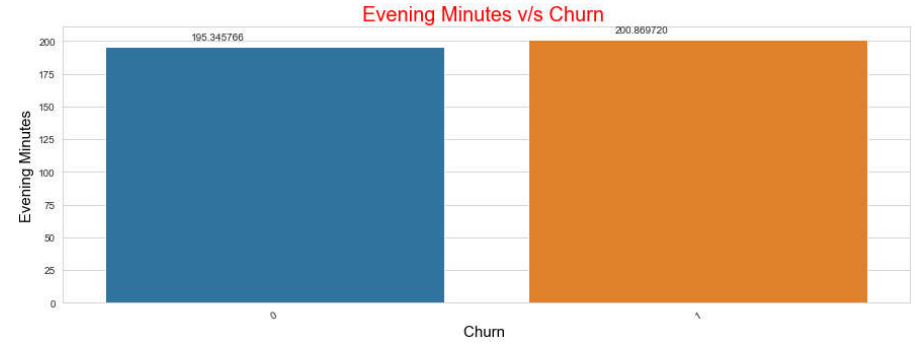

VMail Plan v/s Churn
0.287554
0.147866
VMail Plan
Churn

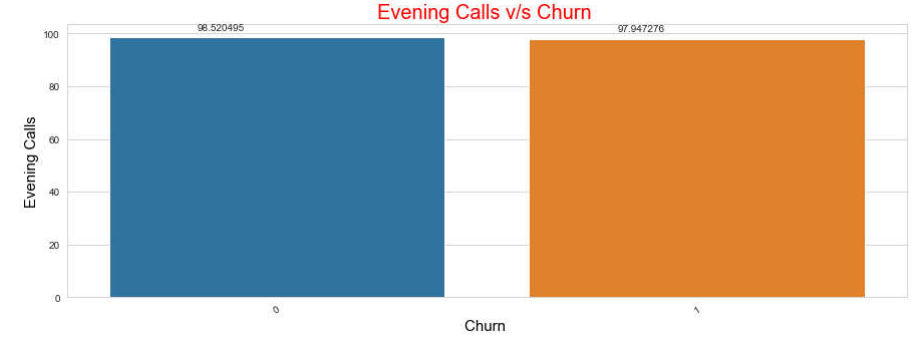

VMail Message v/s Churn
29.102873
30.445652
VMail Message
Churn

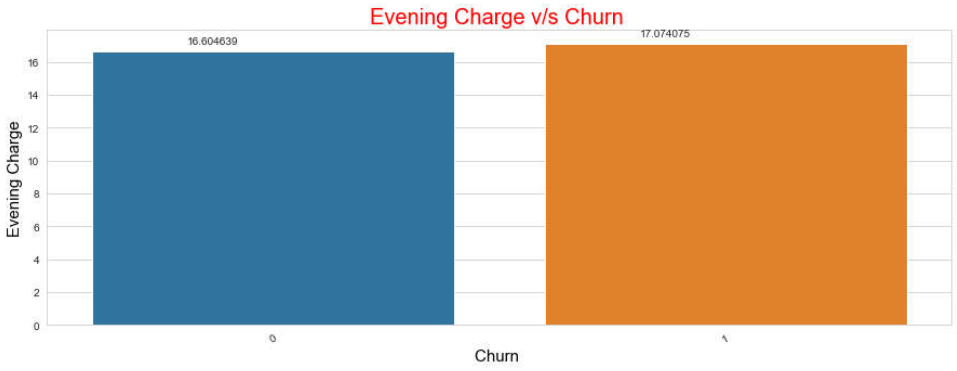

Day Minutes v/s Churn
175.688465
176.838015
Day Minutes
Churn

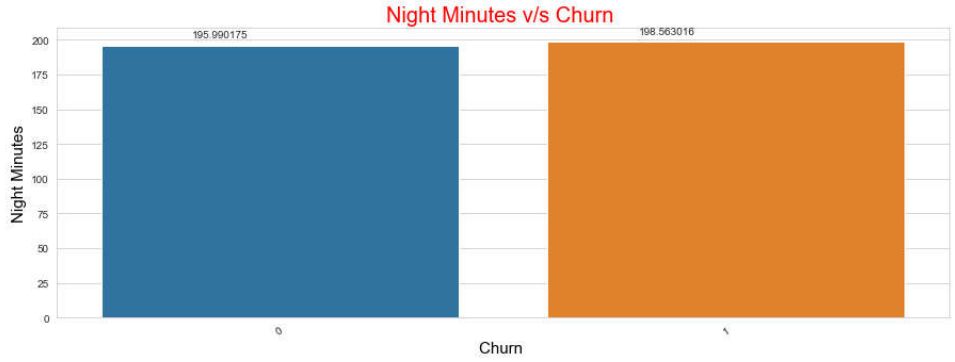

Day Calls v/s Churn
98.310365
99.437838
Day Calls
Churn

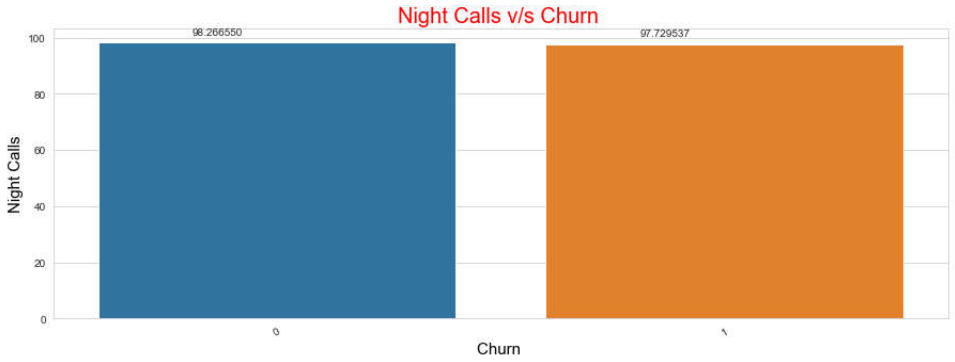

Day Charge v/s Churn
29.867630
30.062760
Day Charge
Churn

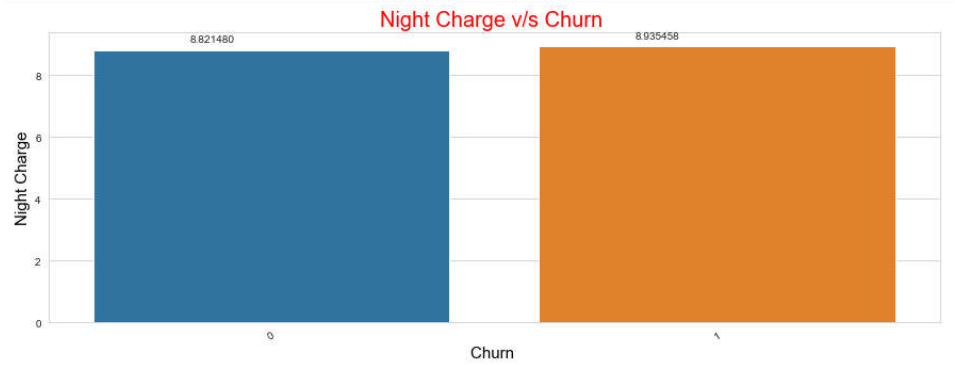

Evening Minutes v/s Churn
195.345766
200.869720
Evening Minutes
Churn

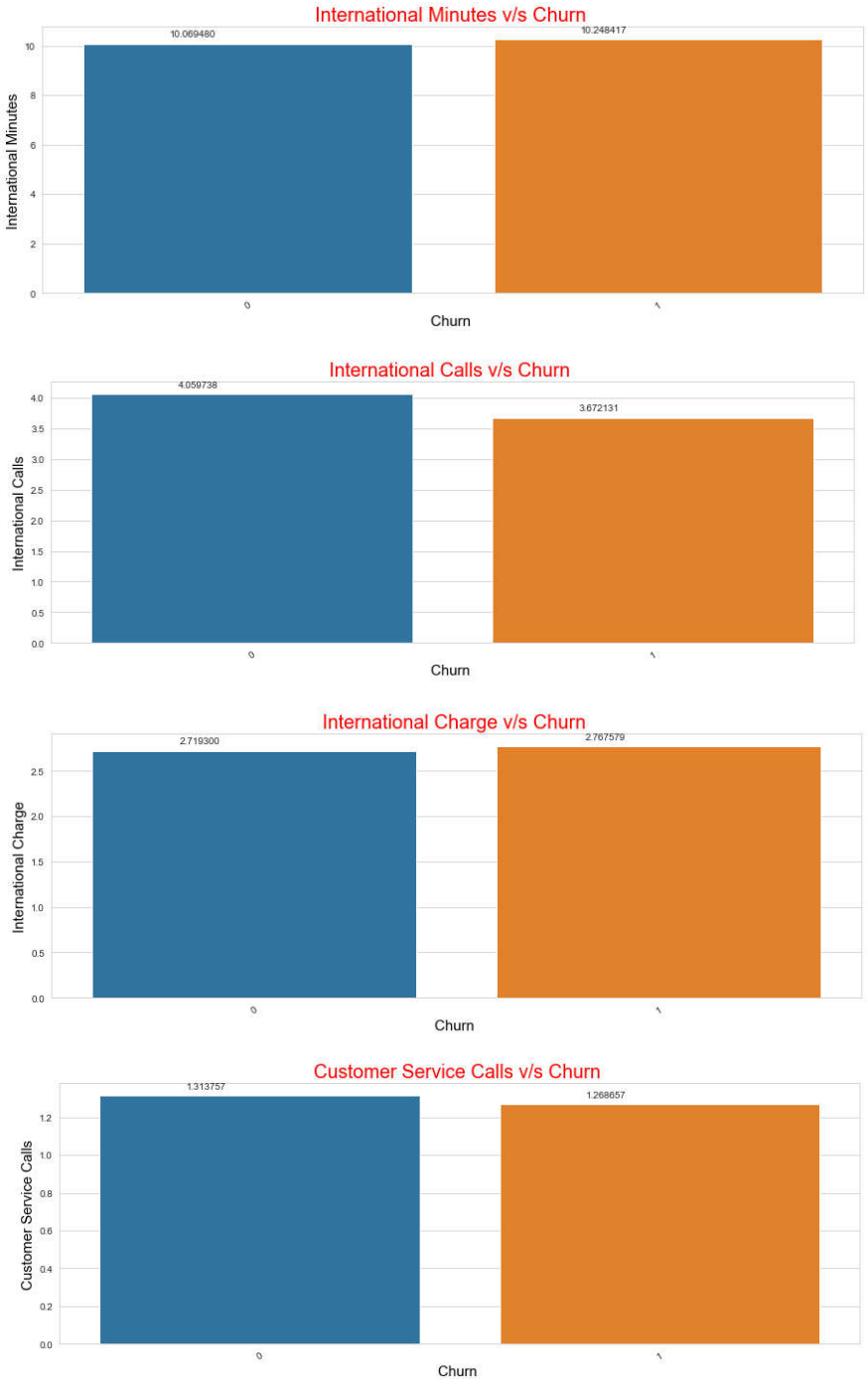

Evening Calls v/s Churn
98.520495
97.947276
Evening Calls
Churn


Evening Charge v/s Churn
16.604639
17.074075
Evening Charge
Churn


Night Minutes v/s Churn
195.990175
198.563016
Night Minutes
Churn

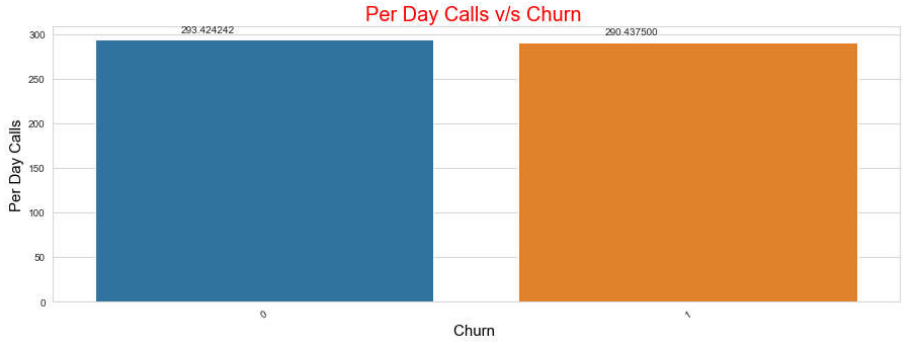

Night Calls v/s Churn
98.266550
97.729537
Night Calls
Churn

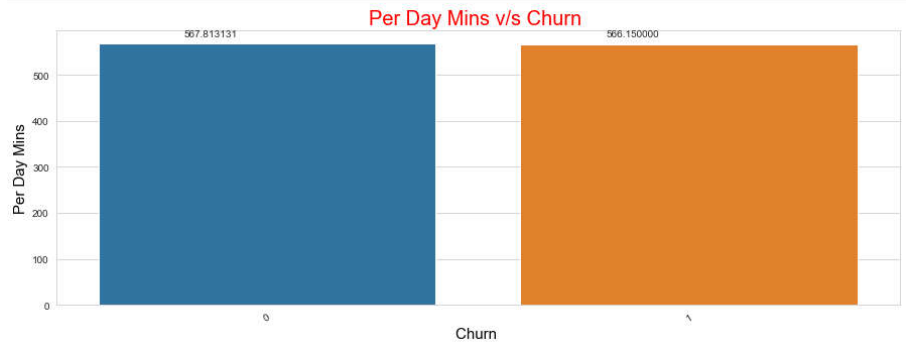

Night Charge v/s Churn
8.821480
8.935458
Night Charge
Churn

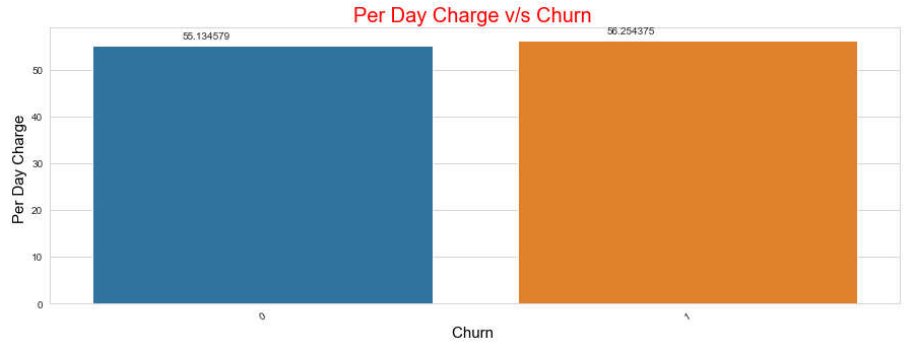


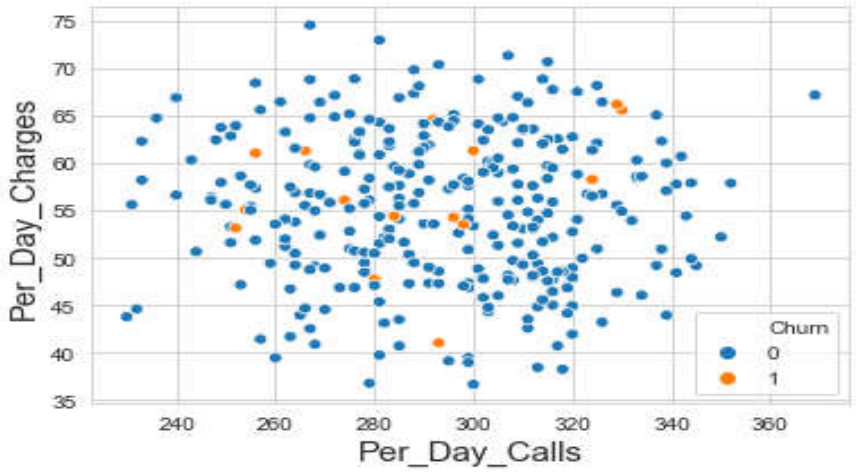
Generate new fields as follows:

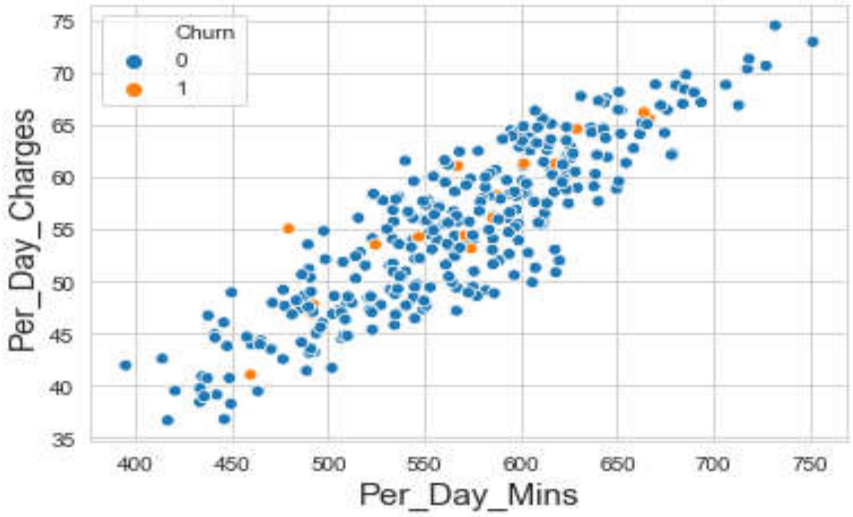
data['Per_Day_Calls']=data['clean_Day_Calls']+data['clean_Eve_Calls']+data['clean_Night_Calls']
data['Per_Day_Mins']=data['clean_Day_Mins']+data['clean_Eve_Mins']+data['clean_Night_Mins']
data['Per_Day_Charge']=data['clean_Day_Charge']+data['clean_Eve_Charge']+data['clean_Night_Charge']


Per Day Calls v/s Churn
293.424242
290.437500
Per Day Calls
Churn

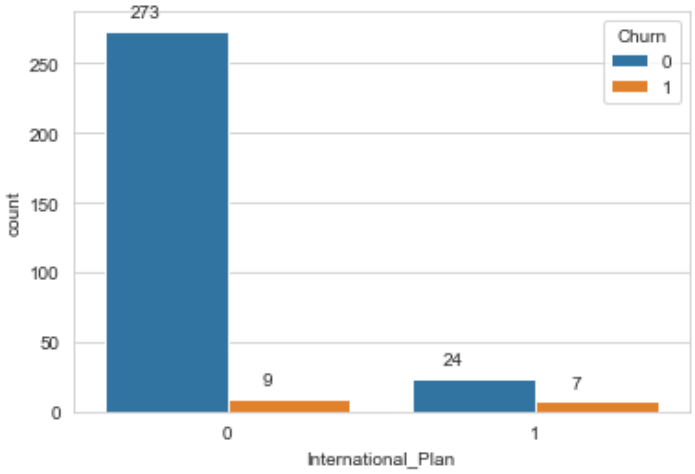

Per Day Mins v/s Churn
567.813131
566.150000
Per Day Mins
Churn

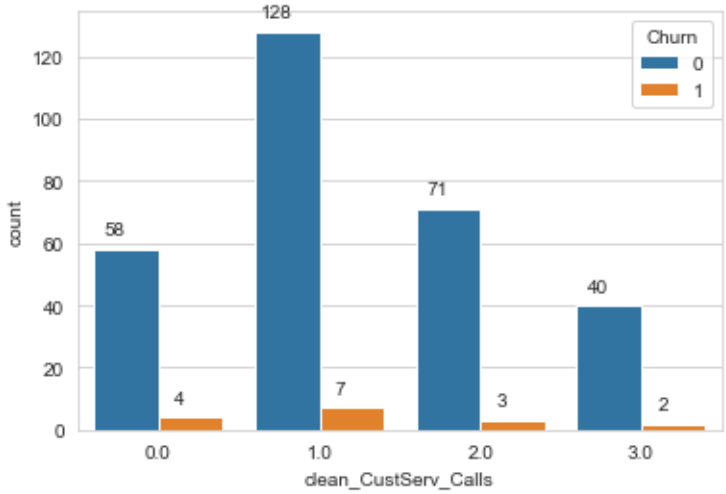

Per Day Charge v/s Churn
55.134579
56.254375
Per Day Charge
Churn

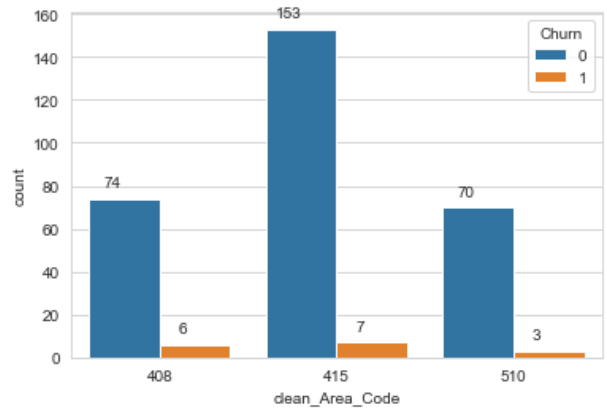

Per_Day_Charges
Per_Day_Calls
Churn
0
1

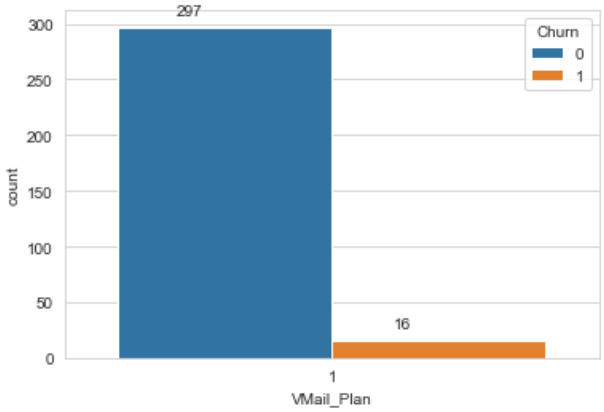


9. Categorizing the features with respect to Churn









Removing null values

data.isna().sum().to_frame().T

data.dropna(axis=0,inplace=True)

data.isna().sum().to_frame().T

2. Creating Churn risk scores that can be indicative to drive retention campaigns.

Define X and y variables

```
X=data[['International_Plan','VMail_Plan',
'clean_VMail_Message','Per_Day_Calls','Per_Day_Mins','Per_Day_Charge',
      'clean_International_Mins',    'clean_International_Calls',    'clean_International_Charge',
'clean_CustServ_Calls']]
y=data.Churn
```

Using train-test split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=10)
Results obtained using the following algorithms

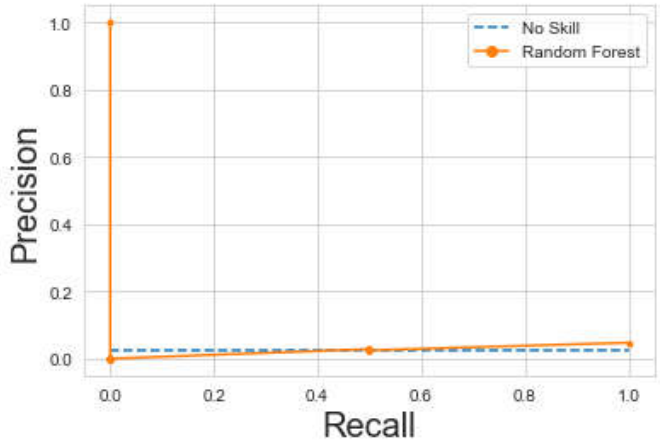
1) Random-Forest Classifier

Accuracy of Training = 96.15384615384616
Accuracy of Testing = 97.46835443037975
Precision score = 95.00080115366126
Recall score = 97.46835443037975
F1 score = 96.21876014281078

	precision	recall	f1-score	support
0	0.97	1.00	0.99	77
1	0.00	0.00	0.00	2
accuracy			0.97	79
macro avg	0.49	0.50	0.49	79
weighted avg	0.95	0.97	0.96	79

Churn Risk Score

ROC and Precision-Recall Curve



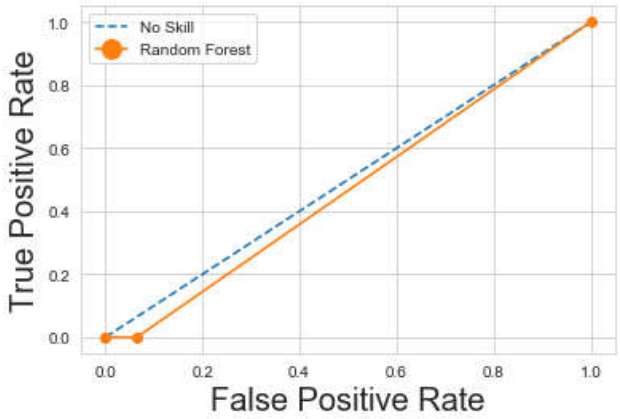
1.1) Using Synthetic Minority Over-sampling Technique (SMOTE) on Random-Forest Classifier

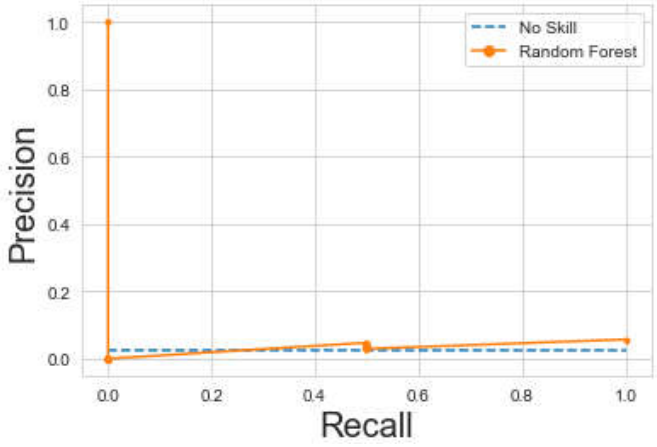
Accuracy of Training = 100.0
Accuracy of Testing = 91.13924050632912

Precision score = 94.83407458091003
Recall score = 91.13924050632912
F1 score = 92.94995389387208

	precision	recall	f1-score	support
0	0.97	0.94	0.95	77
1	0.00	0.00	0.00	2
accuracy			0.91	79
macro avg	0.49	0.47	0.48	79
weighted avg	0.95	0.91	0.93	79

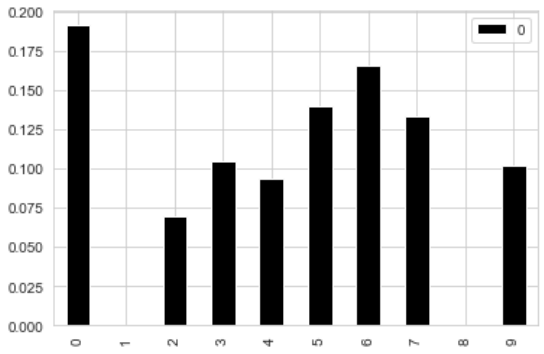
ROC and Precision-Recall Curve





2) eXtreme Gradient Boosting (XGBoost) Classifier

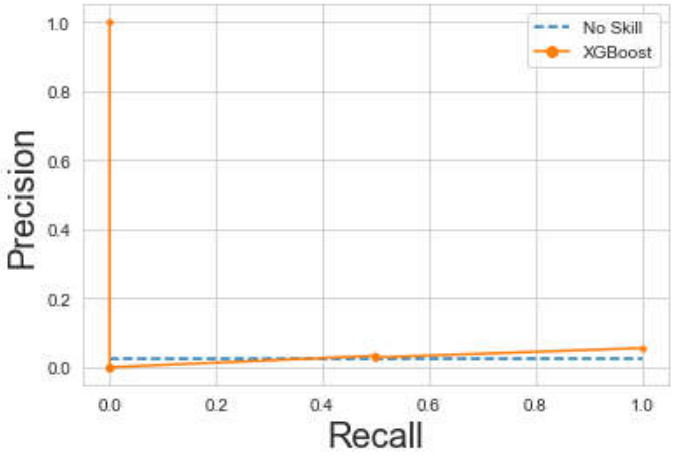
Feature Importance



Accuracy of Training = 97.00854700854701
Accuracy of Testing = 97.46835443037975
Precision score = 95.00080115366126
Recall score = 97.46835443037975
F1 score = 96.21876014281078

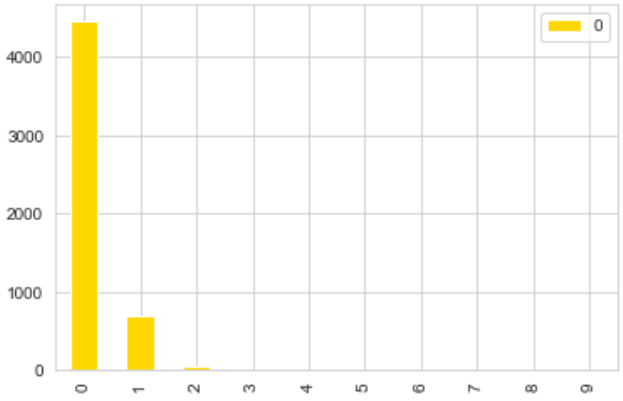
	precision	recall	f1-score	support
0	0.97	1.00	0.99	77
1	0.00	0.00	0.00	2
accuracy			0.97	79
macro avg	0.49	0.50	0.49	79
weighted avg	0.95	0.97	0.96	79

ROC and Precision-Recall Curve

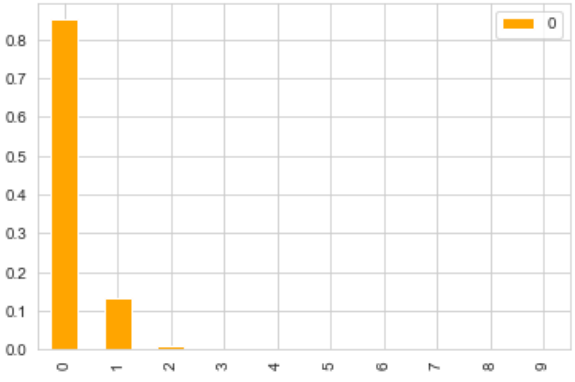


2.1) Using Principal Component Analysis (PCA) on eXtreme Gradient Boosting (XGBoost) Classifier

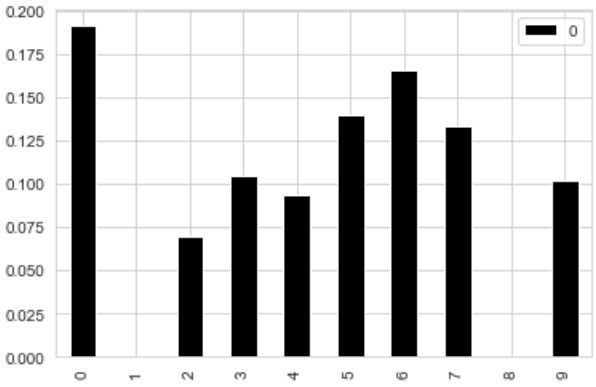
Explained variance



Explained variance ratio



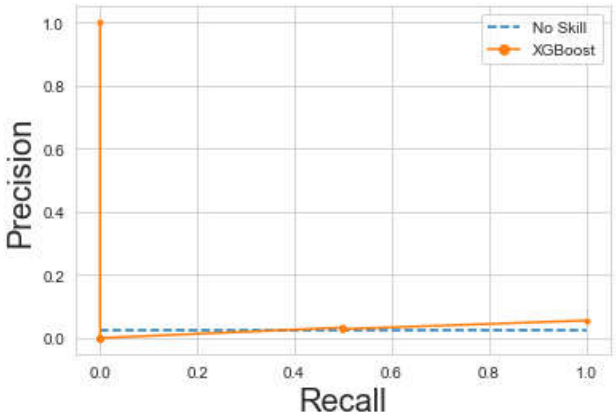
Feature Importance



Accuracy of Training = 97.00854700854701
Accuracy of Testing = 97.46835443037975
Precision score = 95.00080115366126
Recall score = 97.46835443037975
F1 score = 96.21876014281078

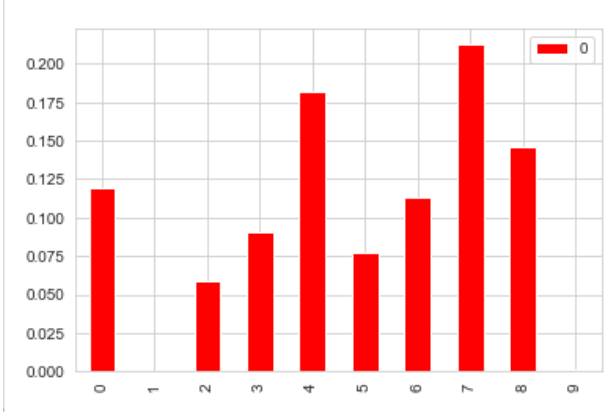
	precision	recall	f1-score	support
0	0.97	1.00	0.99	77
1	0.00	0.00	0.00	2
accuracy			0.97	79
macro avg	0.49	0.50	0.49	79
weighted avg	0.95	0.97	0.96	79

ROC and Precision-Recall Curve



3) Gradient Boosting Classifier

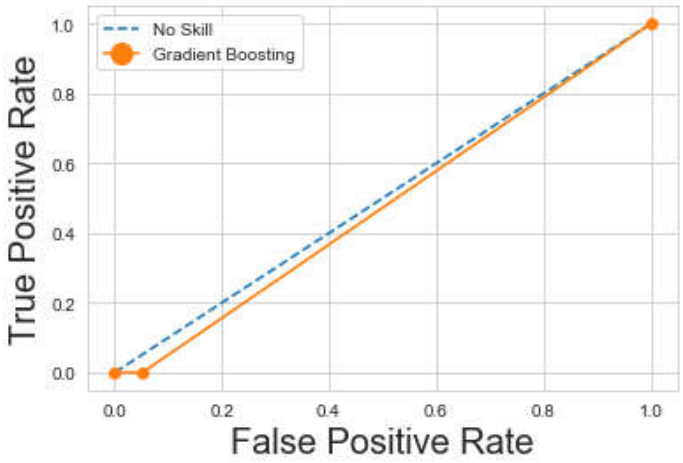
Feature Importance

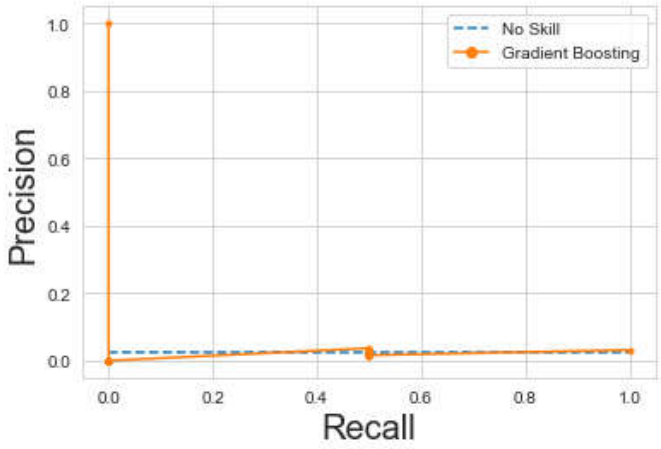


Accuracy of Training = 100.0
Accuracy of Testing = 92.40506329113924
Precision score = 94.86919831223629
Recall score = 92.40506329113924
F1 score = 93.62091938707529

	precision	recall	f1-score	support
0	0.97	0.95	0.96	77
1	0.00	0.00	0.00	2
accuracy			0.92	79
macro avg	0.49	0.47	0.48	79
weighted avg	0.95	0.92	0.94	79

ROC and Precision-Recall curve



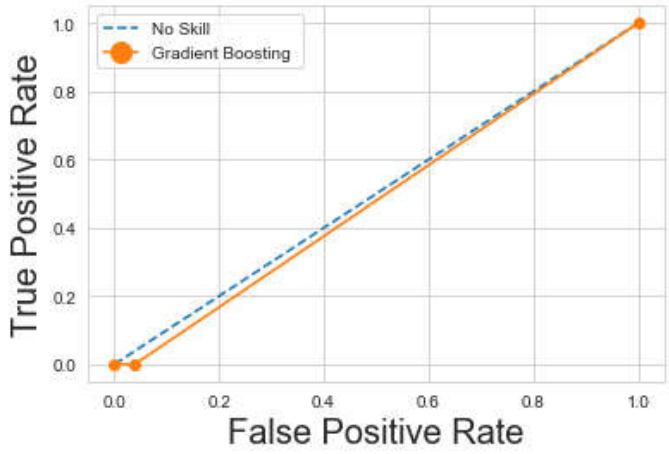


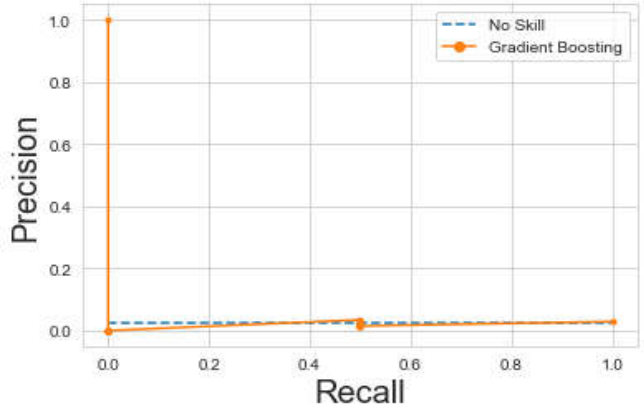
3.1) Using GridSearch Cross Validation on Gradient Boosting Classifier

Accuracy of Training = 97.00854700854701
Accuracy of Testing = 93.67088607594937
Precision score = 94.90339773484344
Recall score = 93.67088607594937
F1 score = 94.28311408951767

	precision	recall	f1-score	support
0	0.97	0.96	0.97	77
1	0.00	0.00	0.00	2
accuracy			0.94	79
macro avg	0.49	0.48	0.48	79
weighted avg	0.95	0.94	0.94	79

ROC and Precision-Recall curve



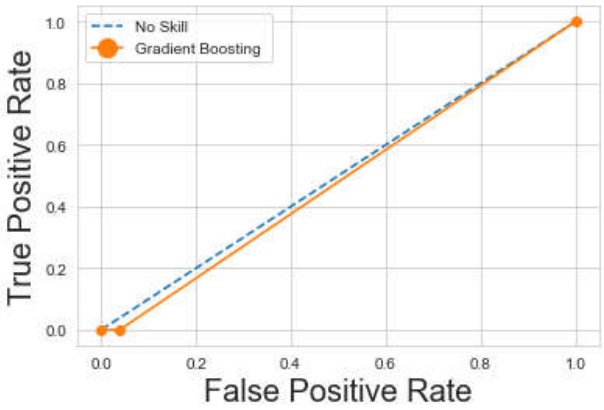


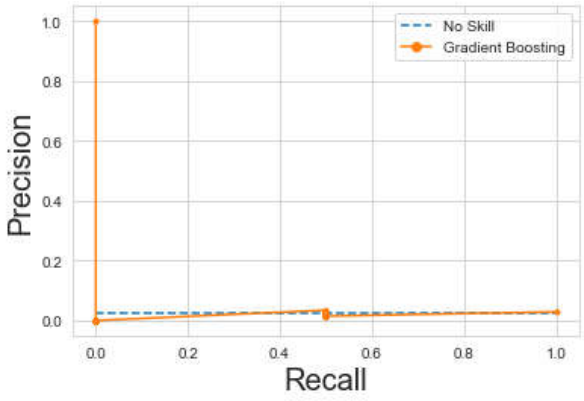
3.2) Using RandomizedSearch Cross Validation on Gradient Boosting Classifier

Accuracy of Training = 97.00854700854701
Accuracy of Testing = 93.67088607594937
Precision score = 94.90339773484344
Recall score = 93.67088607594937
F1 score = 94.28311408951767

	precision	recall	f1-score	support
0	0.97	0.96	0.97	77
1	0.00	0.00	0.00	2
accuracy			0.94	79
macro avg	0.49	0.48	0.48	79
weighted avg	0.95	0.94	0.94	79

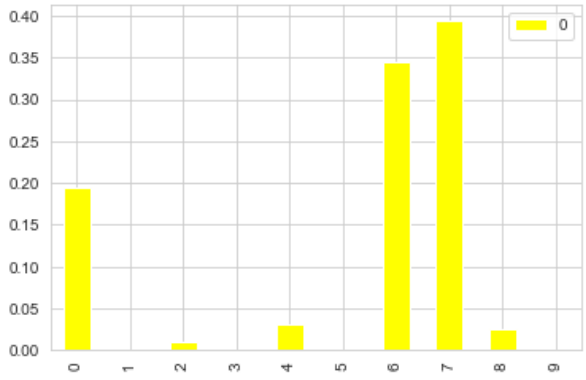
ROC and Precision-Recall Curve





4) Decision-Tree Classifier

Feature Importance

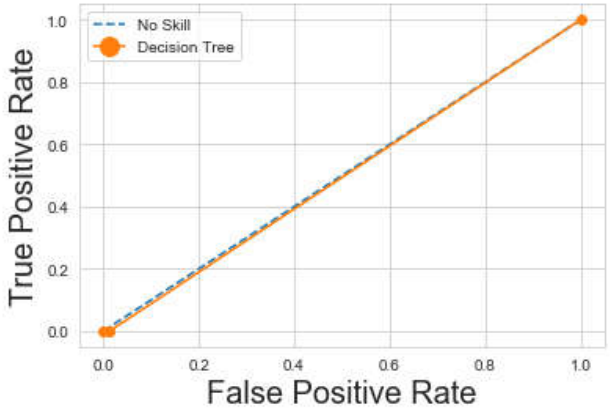


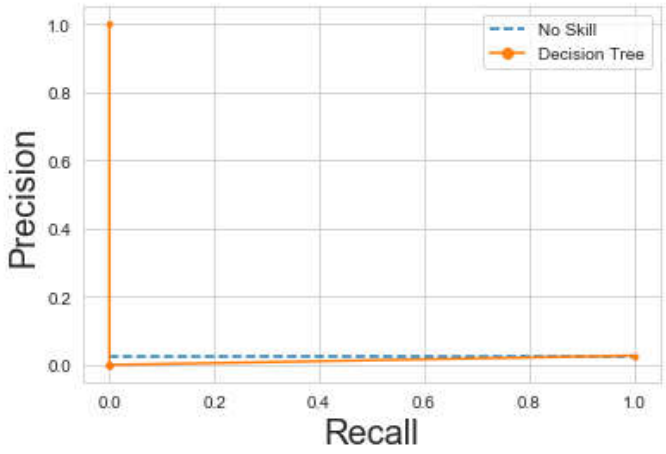
Accuracy of Training = 97.00854700854701
Accuracy of Testing = 96.20253164556962
Precision score = 94.9691658552418
Recall score = 96.20253164556962
F1 score = 95.58187015108206

	precision	recall	f1-score	support
0	0.97	0.99	0.98	77
1	0.00	0.00	0.00	2

accuracy			0.96	79
macro avg	0.49	0.49	0.49	79
weighted avg	0.95	0.96	0.96	79

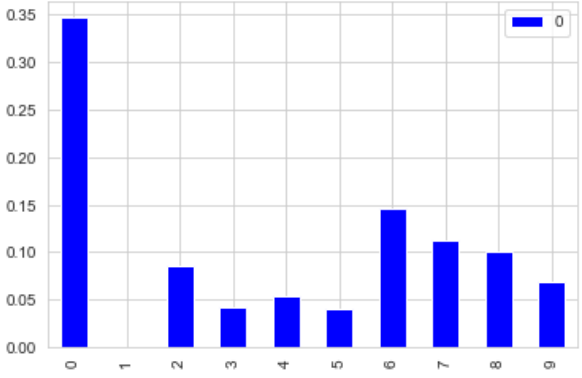
ROC and Precision-Recall Curve





5) Extra Trees Classifier

Feature Importance

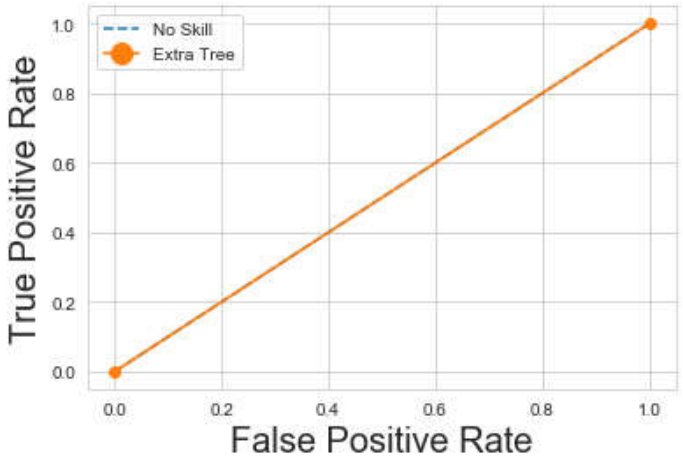


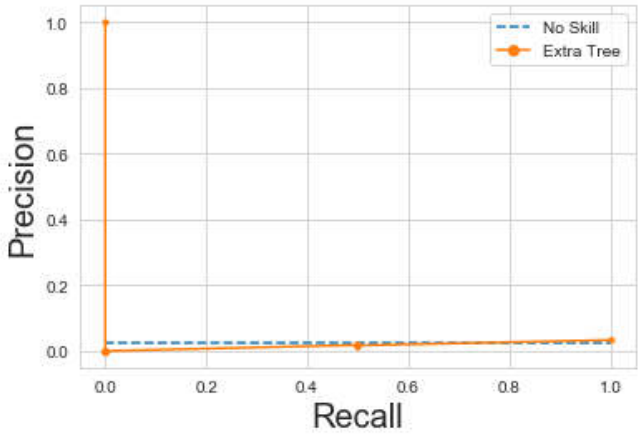
Accuracy score of Training = 94.44444444444444

Accuracy score of Testing = 97.46835443037975
Precision score = 95.00080115366126
Recall score = 97.46835443037975
F1 score = 96.21876014281078

	precision	recall	f1-score	support
0	0.97	1.00	0.99	77
1	0.00	0.00	0.00	2
accuracy			0.97	79
macro avg	0.49	0.50	0.49	79
weighted avg	0.95	0.97	0.96	79

ROC and Precision-Recall Curve





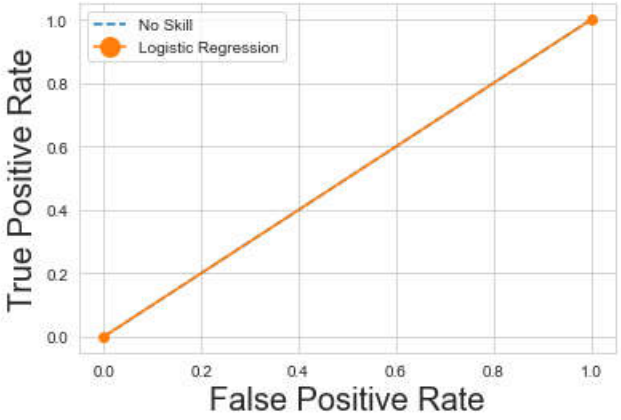
6) Logistic Regression

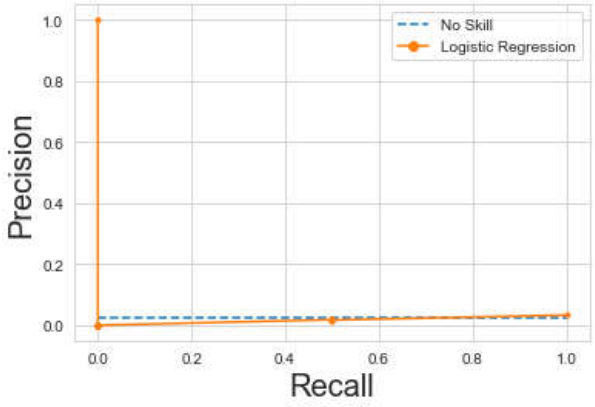
Accuracy score of Training = 94.44444444444444
Accuracy score of Testing = 97.46835443037975
Precision score = 95.00080115366126
Recall score = 97.46835443037975

F1 score = 96.21876014281078

	precision	recall	f1-score	support
0	0.97	1.00	0.99	77
1	0.00	0.00	0.00	2
accuracy			0.97	79
macro avg	0.49	0.50	0.49	79
weighted avg	0.95	0.97	0.96	79

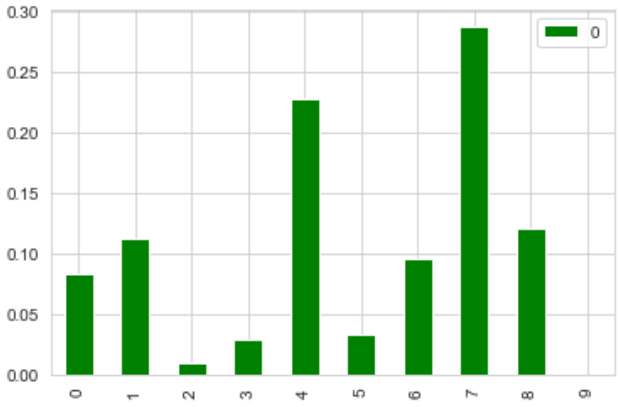
ROC and Precision-Recall Curve





Using K-folds cross validation on Decision Tree Classifier; we get mean score of 91.38.

Feature Importances



Using Stratified K-folds cross validation,
Accuracy of Training = 100.0
Accuracy of Testing = 90.32258064516128

2. Introduce new predicting variable "CHURN-FLAG" with values YES(1) or NO(0) so that email campaigns with lucrative offer can be targeted to Churn YES customers.

```
data['CHURN_FLAG'] = data['Churn']
data.drop(labels='Churn',axis=1,inplace=True)
data.columns
data.dtypes
```

Feature selection

```
clean_Area_Code               -0.185893
State                         -0.085644
clean_Day_Charge              -0.084094
clean_Day_Mins                -0.084089
International_Plan            -0.043125
Per_Day_Charge                -0.043064
clean_International_Calls     -0.040565
clean_VMail_Message           -0.034554
CHURN_FLAG                    -0.024923
clean_Account_Length          -0.008691
Per_Day_Mins                   0.002028
clean_International_Mins       0.019770
clean_International_Charge     0.019890
clean_Night_Charge             0.033733
clean_Night_Mins               0.033803
clean_Eve_Mins                 0.060363
clean_Eve_Charge               0.060403
clean_CustServ_Calls           0.083357
clean_Night_Calls              0.587284
clean_Day_Calls                0.608418
clean_Eve_Calls                0.633974
Per_Day_Calls                  1.000000
VMail_Plan                          NaN
Name: Per_Day_Calls, dtype: float64
```

```
International_Plan            -0.052170
clean_International_Mins      -0.039046
clean_International_Charge    -0.038744
State                         -0.030622
clean_CustServ_Calls          -0.029631
clean_VMail_Message           -0.015128
clean_Area_Code               -0.012432
clean_Night_Calls             -0.012319
CHURN_FLAG                    -0.005529
clean_Eve_Calls                0.000924
Per_Day_Calls                  0.002028
clean_International_Calls      0.003557
clean_Day_Calls                0.015482
clean_Account_Length           0.169658
clean_Night_Mins               0.551867
clean_Night_Charge             0.551883
clean_Eve_Mins                 0.562047
clean_Eve_Charge               0.562048
clean_Day_Charge               0.580288
clean_Day_Mins                 0.580289
Per_Day_Charge                 0.879670
Per_Day_Mins                   1.000000
VMail_Plan                          NaN
Name: Per_Day_Mins, dtype: float64
```

```
clean_CustServ_Calls          -0.074733
clean_Eve_Calls               -0.046877
International_Plan            -0.043600
Per_Day_Calls                 -0.043064
clean_International_Calls     -0.034870
clean_Night_Calls             -0.033155
State                         -0.026595
clean_VMail_Message           -0.025468
clean_Area_Code               -0.023026
clean_International_Mins      -0.016544
clean_International_Charge    -0.015991
clean_Day_Calls                0.001995
CHURN_FLAG                     0.031839
clean_Account_Length           0.177363
clean_Night_Mins               0.184429
clean_Night_Charge             0.184443
clean_Eve_Charge               0.396658
clean_Eve_Mins                 0.396660
Per_Day_Mins                   0.879670
clean_Day_Mins                 0.880943
clean_Day_Charge               0.880944
Per_Day_Charge                 1.000000
VMail_Plan                          NaN
Name: Per_Day_Charge, dtype: float64
```

Define X and y variables

X=data.loc[:,['Per_Day_Calls','Per_Day_Mins','Per_Day_Charge','clean_CustServ_Calls','clean_International_Mins','clean_International_Calls','clean_International_Charge']]

X.head()

y=data.CHURN_FLAG

Using train-test split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=10)

Using following algorithms we get the following results:

1) Random-Forest Classifier

Accuracy of Training = 95.2991452991453
Accuracy of Testing = 93.67088607594937
Precision score = 94.90339773484344
Recall score = 93.67088607594937
F1 score = 94.28311408951767

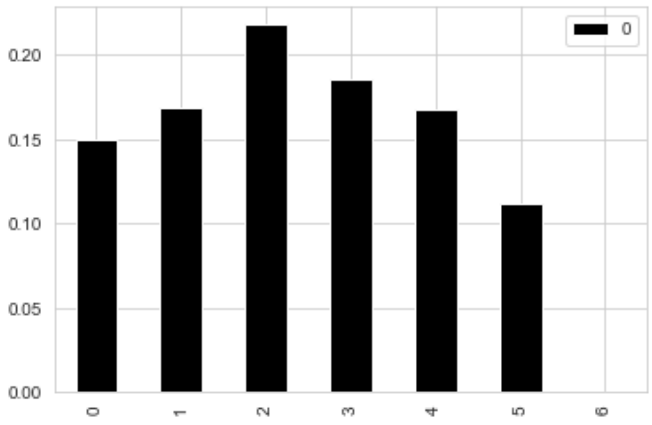
	precision	recall	f1-score	support
0	0.97	0.96	0.97	77
1	0.00	0.00	0.00	2
accuracy			0.94	79
macro avg	0.49	0.48	0.48	79
weighted avg	0.95	0.94	0.94	79

2) XGBoost Classifier

Accuracy of Training = 95.72649572649573
Accuracy of Testing = 93.67088607594937
Precision score = 94.90339773484344
Recall score = 93.67088607594937
F1 score = 94.28311408951767

	precision	recall	f1-score	support
0	0.97	0.96	0.97	77
1	0.00	0.00	0.00	2
accuracy			0.94	79
macro avg	0.49	0.48	0.48	79
weighted avg	0.95	0.94	0.94	79

Feature Importance



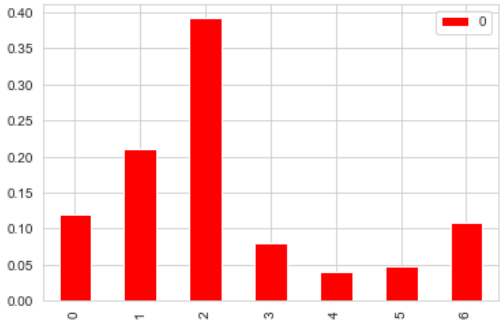
3) Gradient Boosting Classifier

Accuracy of Training = 100.0
Accuracy of Testing = 93.67088607594937

Precision score = 94.90339773484344
Recall score = 93.67088607594937
F1 score = 94.28311408951767

	precision	recall	f1-score	support
0	0.97	0.96	0.97	77
1	0.00	0.00	0.00	2
accuracy			0.94	79
macro avg	0.49	0.48	0.48	79
weighted avg	0.95	0.94	0.94	79

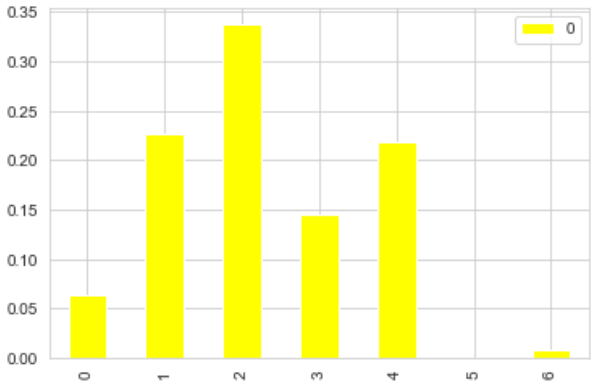
Feature Importance



4) Decision-Tree Classifier

Accuracy of Training = 95.72649572649573
Accuracy of Testing = 91.13924050632912
Precision score = 94.83407458091003
Recall score = 91.13924050632912
F1 score = 92.94995389387208

	precision	recall	f1-score	support
0	0.97	0.94	0.95	77
1	0.00	0.00	0.00	2
accuracy			0.91	79
macro avg	0.49	0.47	0.48	79
weighted avg	0.95	0.91	0.93	79

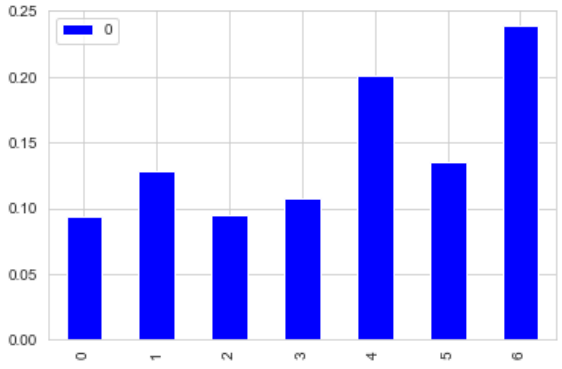


5) Extra Trees Classifier

Accuracy of Training = 94.01709401709401
Accuracy of Testing = 97.46835443037975
Precision score = 95.00080115366126
Recall score = 97.46835443037975
F1 score = 96.21876014281078

	precision	recall	f1-score	support
0	0.97	1.00	0.99	77
1	0.00	0.00	0.00	2
accuracy			0.97	79
macro avg	0.49	0.50	0.49	79
weighted avg	0.95	0.97	0.96	79

Feature Importance



6) Logistic Regression

Accuracy of Training = 94.01709401709401
Accuracy of Testing = 97.46835443037975
Precision score = 95.00080115366126
Recall score = 97.46835443037975
F1 score = 96.21876014281078

	precision	recall	f1-score	support

0	0.97	1.00	0.99	77
1	0.00	0.00	0.00	2
accuracy			0.97	79
macro avg	0.49	0.50	0.49	79
weighted avg	0.95	0.97	0.96	79

4. Exporting the trained model with prediction capability for CHURN-FLAG, which can be highlighted in service applications to serve the customer better.

1] Using joblib

Steps:

1. **Save the model in a file**

```
from sklearn.externals import joblib
joblib.dump(rf, 'Telecom_churn_rf.ml')
```

2. **Load the model from the file**

```
rf=joblib.load('Telecom_churn_rf.ml')
```

3. **Use the loaded model to make predictions**

```
rf.predict(X_test)
```

4. **Predict the value**

```
rf.predict([[305.0,497.0,46.00,2.0,8.5,5.0,2.30]])
```

Follow the similar procedure for the other algorithms.

2] Using pickle

Steps:

1. **Save the trained model as a pickle string**

```
saved_model=pickle.dumps(rf)
```

2. **Load the pickled model**

```
model_from_pickle=pickle.loads(saved_model)
```

3. **Use the loaded pickled model to make predictions**

```
model_from_pickle.predict(X_test)
```

Follow the similar procedure for the other algorithms.