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.

It helps 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

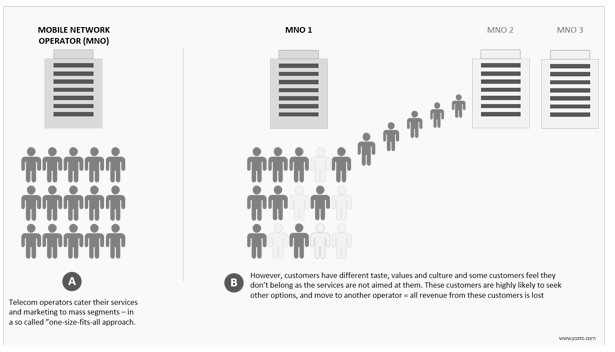
CustServ Calls : Number of customer service calls made

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>



1. **Features**

The features of the datasets were provided by DataMites company.

1. **Assumptions**
2. Dropped the column phone as it is not required for predicting churn
3. Created new fields like Per\_Day\_Call, Per\_Day\_Mins, Per\_Day\_Charge based on Day,Evening and Night calls, minutes and charge.
4. Used Churn as a target variable.
5. Created a new variable called Churn Flag based on Churn: True(1) , False(0)
6. 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','International\_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

1. **Load the dataset**

data=pd.read\_excel("telecom\_churn\_data.xlsx")

data

# 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)

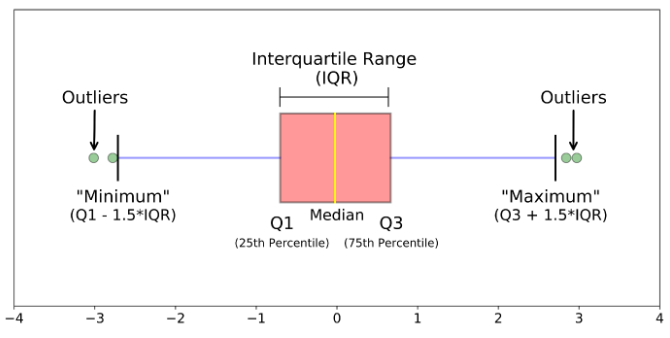
Counter(data.International\_Calls)

Counter(data.International\_Charge)

Counter(data.Churn)

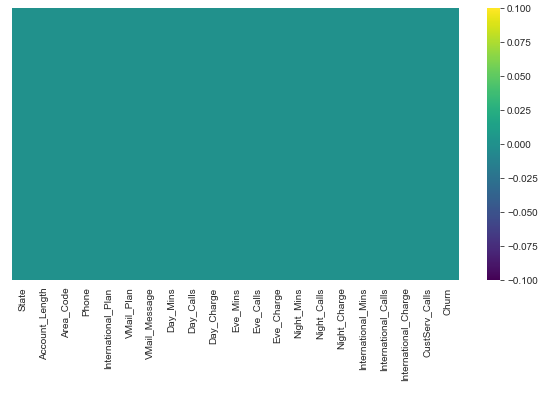
1. **Checking for the outliers**

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.

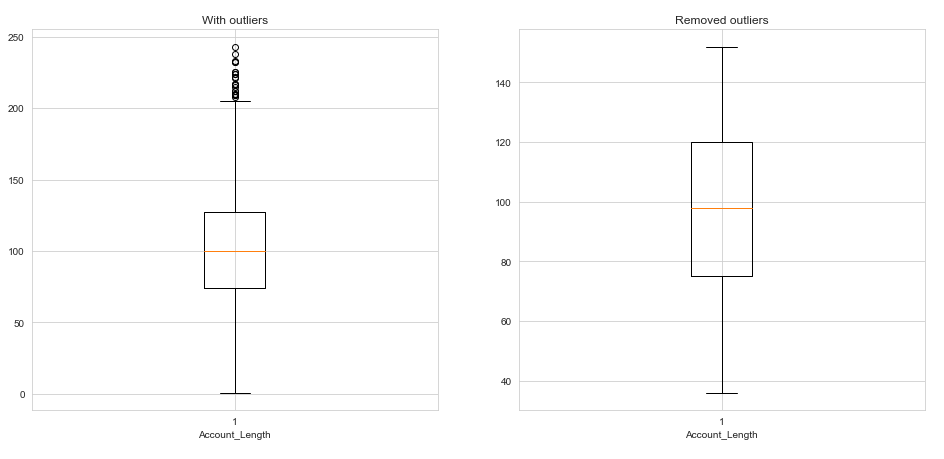


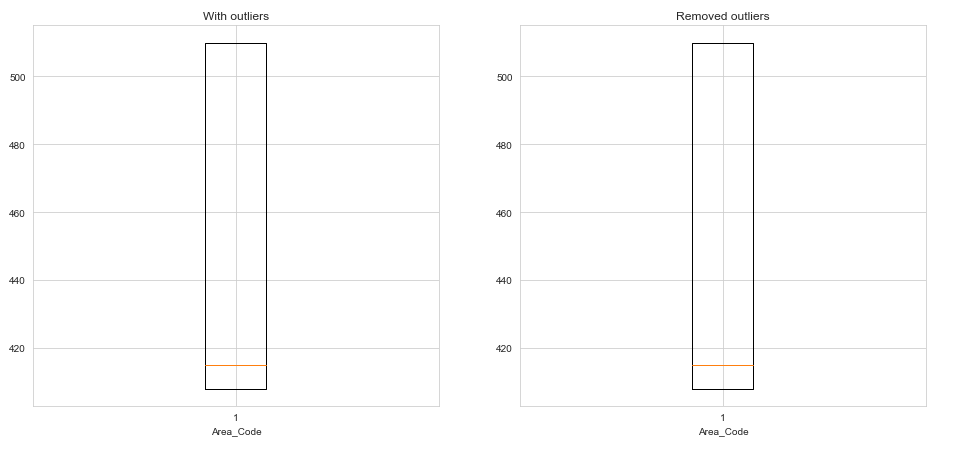
Boxplots are a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”).

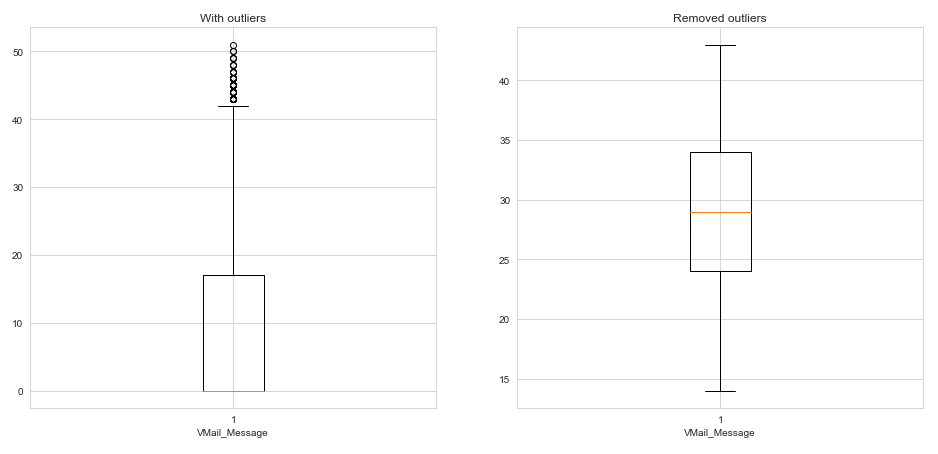
Heatmap representing the dataset.

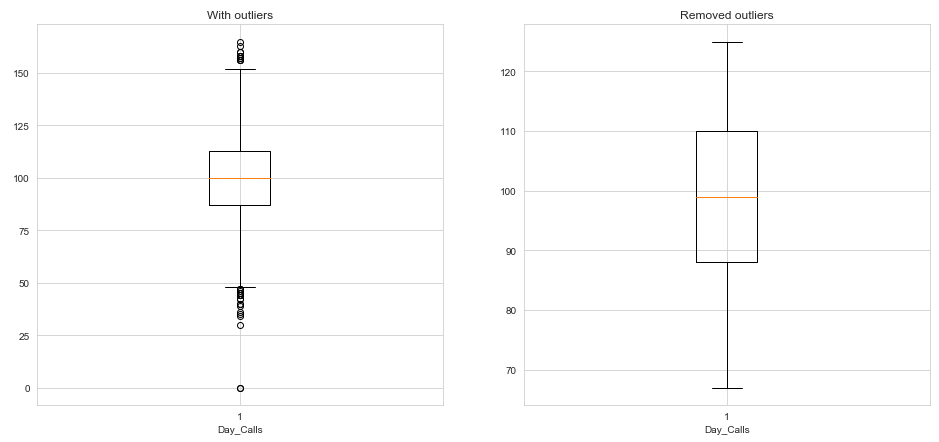


Boxplots to detect and remove the outliers

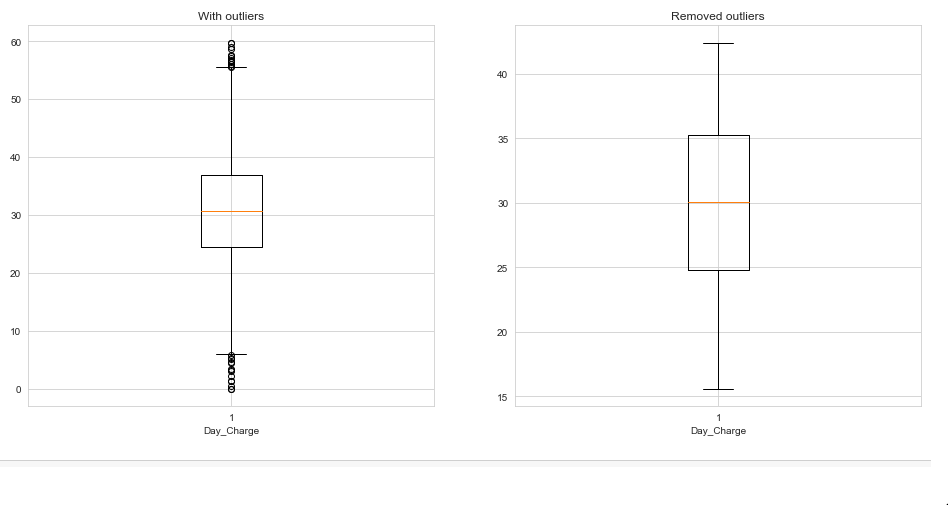




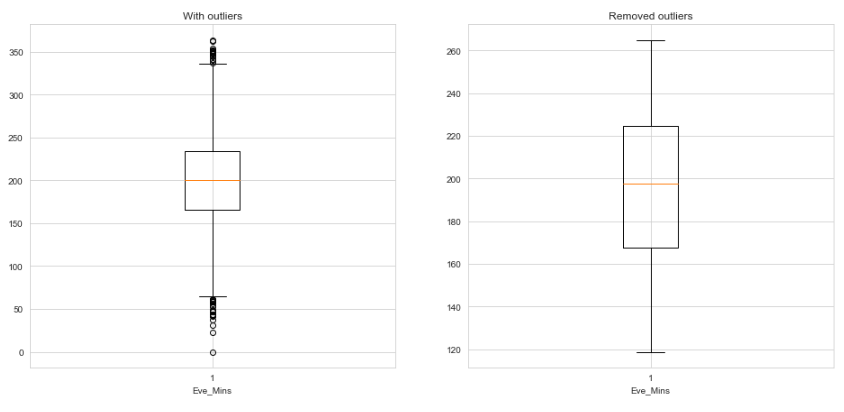


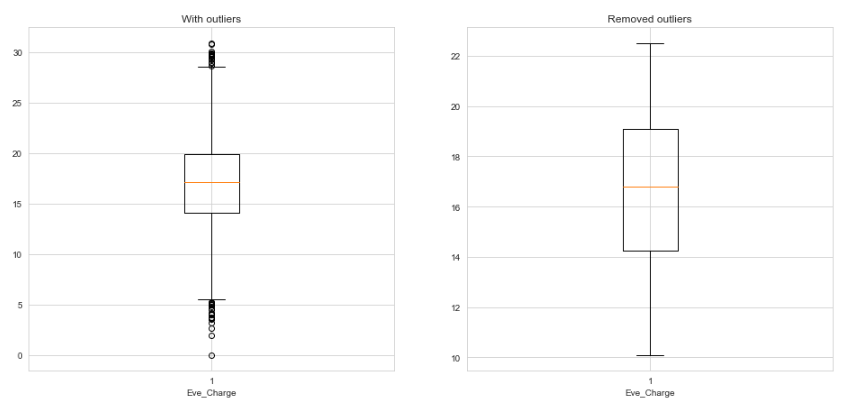


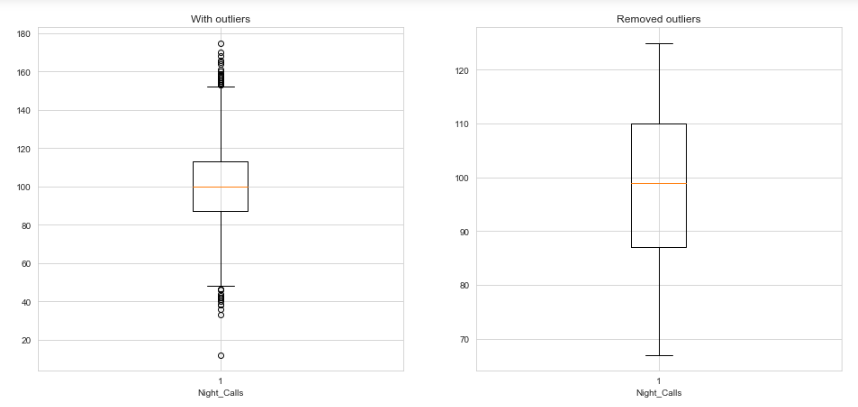


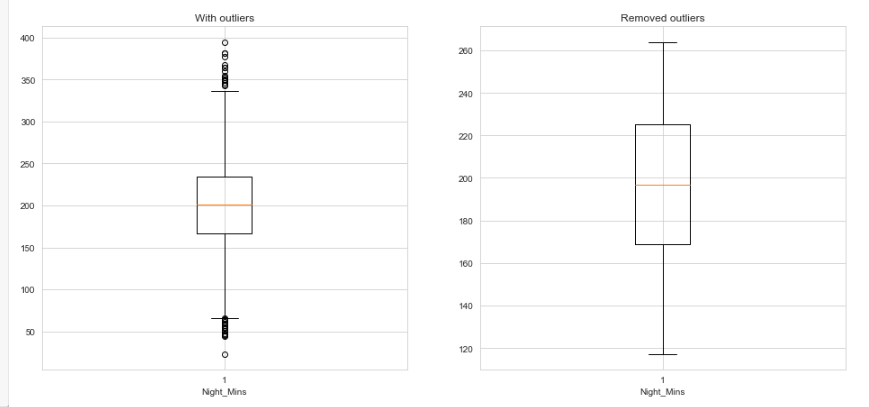


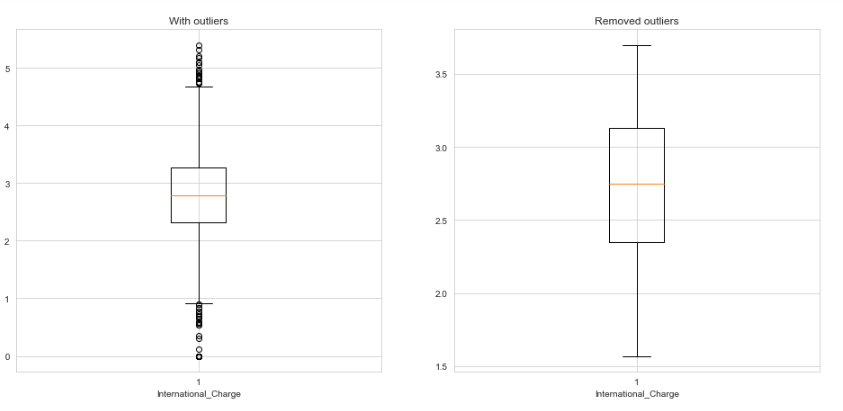
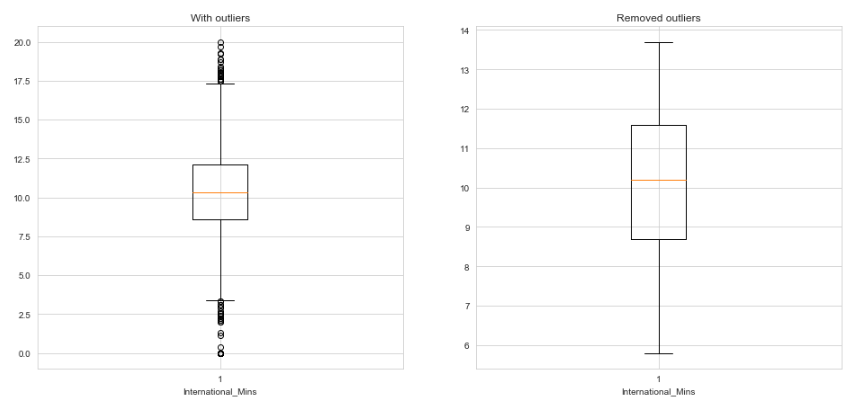
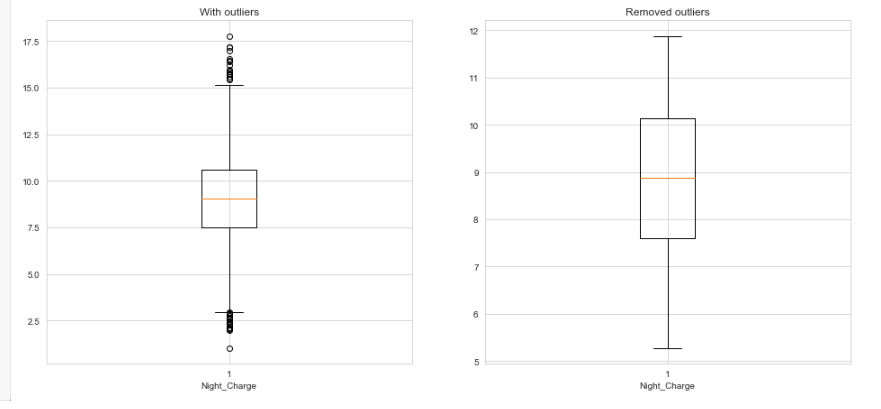


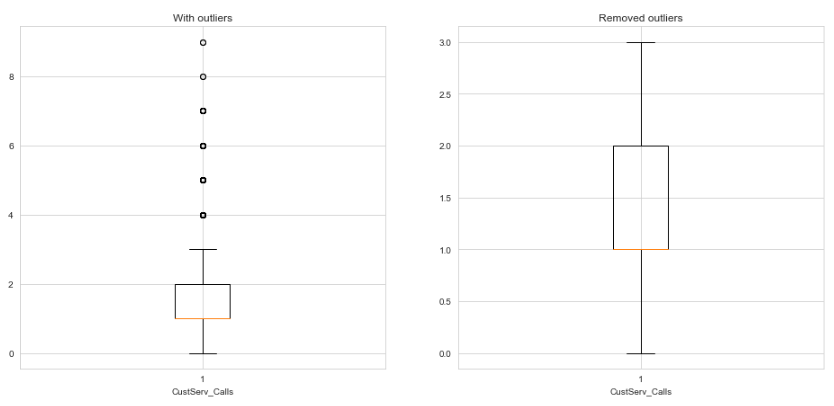












1. **Using Label Encoding**

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()

1. **Drop the following fields**

data.drop(['Account\_Length','Area\_Code'],inplace=True,axis=1)

data.drop(['VMail\_Message','Day\_Calls','Day\_Mins','Day\_Charge',

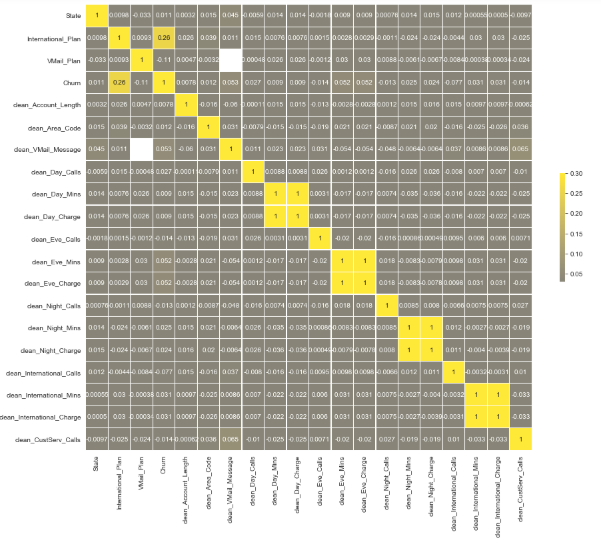
'Eve\_Calls','Eve\_Mins','Eve\_Charge',

'Night\_Calls','Night\_Mins','Night\_Charge',

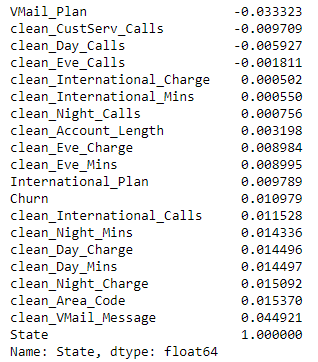
'International\_Calls','International\_Mins','International\_Charge',

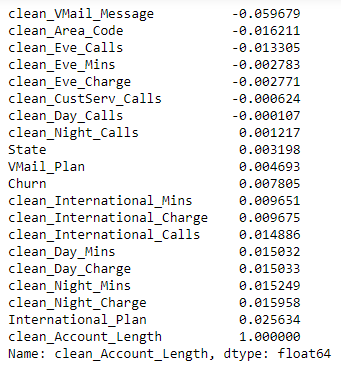
'CustServ\_Calls'],inplace=True,axis=1)

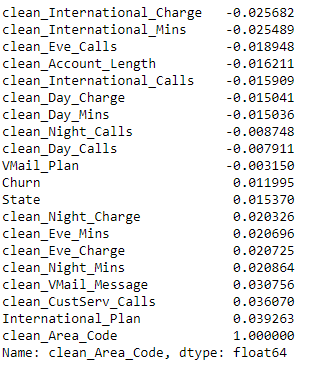
1. **Correlation Matrix**

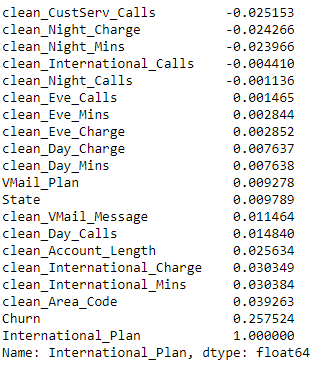
****

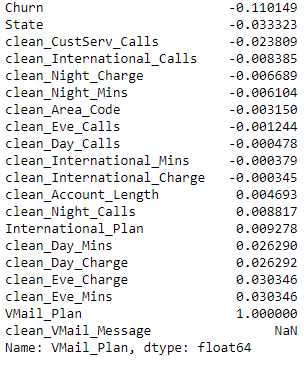
Some of the results of feature selection

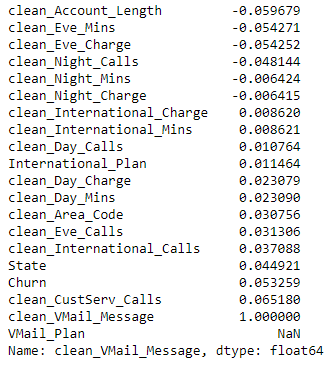


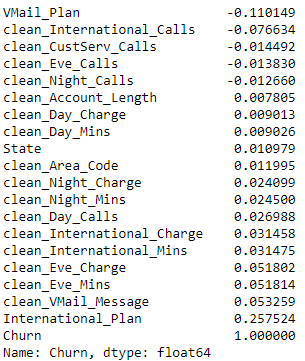
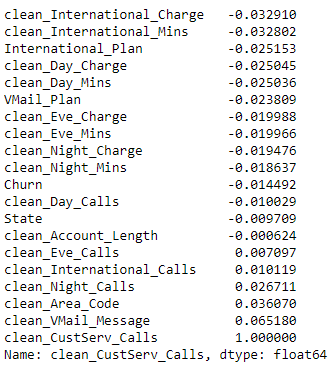












1. **Data Exploration Insights**

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

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 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']

# 

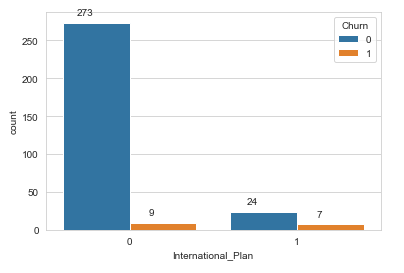
# 

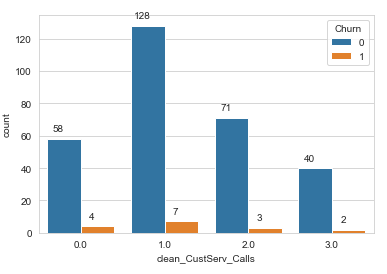
# 

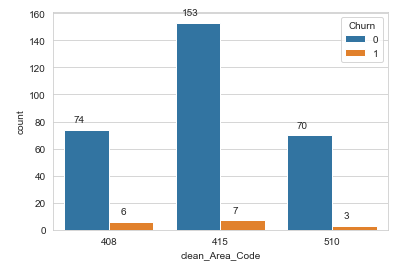
# 

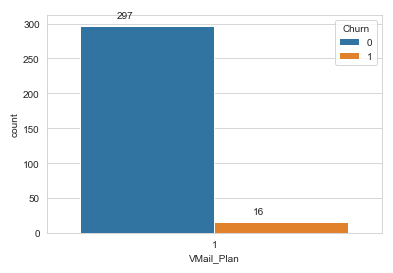
# 

### 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

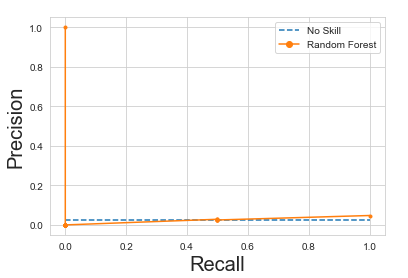
macro avg 0.49 0.50 0.49 79

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

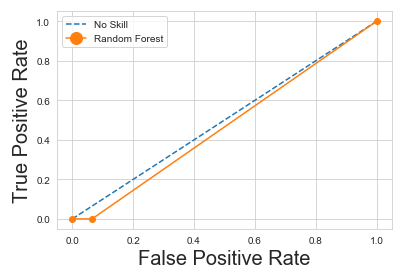
1 0.00 0.00 0.00 2

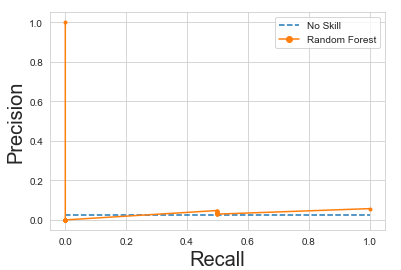
accuracy 0.91 79

macro avg 0.49 0.47 0.48 79

weighted avg 0.95 0.91 0.93 79

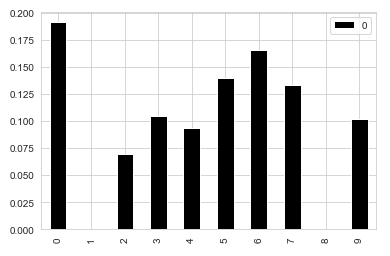
ROC and Precision-Recall Curve





# 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

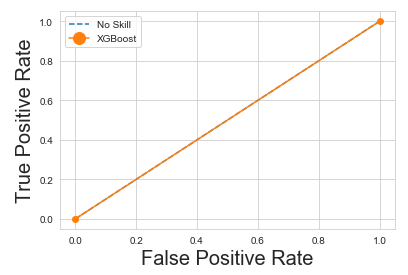
1 0.00 0.00 0.00 2

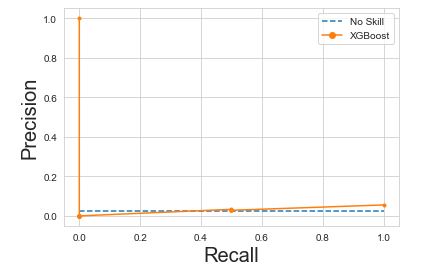
accuracy 0.97 79

macro avg 0.49 0.50 0.49 79

weighted avg 0.95 0.97 0.96 79

ROC and Precision-Recall Curve



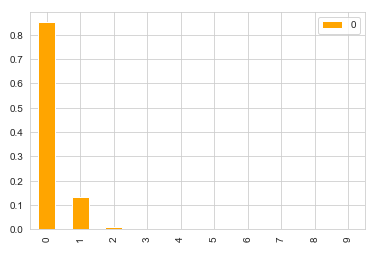


# 2.1) Using Principal Component Analysis (PCA) on eXtreme Gradient Boosting (XGBoost) Classifier

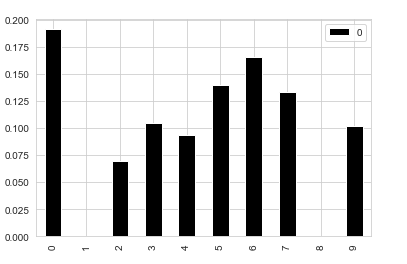
# Explained variance

# 

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

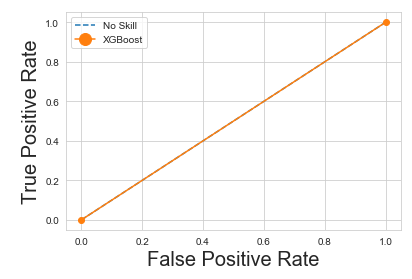
1 0.00 0.00 0.00 2

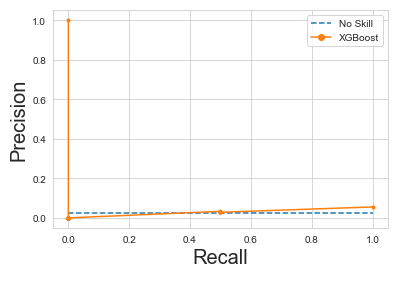
accuracy 0.97 79

macro avg 0.49 0.50 0.49 79

weighted avg 0.95 0.97 0.96 79

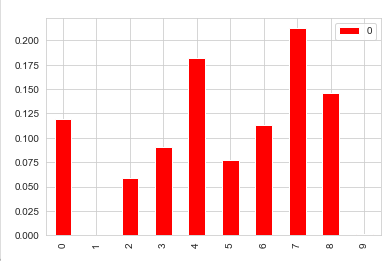
ROC and Precision-Recall Curve





# 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

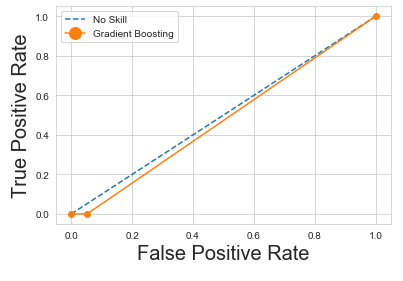
1 0.00 0.00 0.00 2

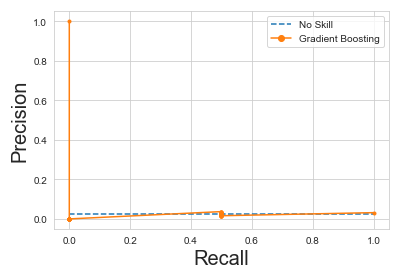
accuracy 0.92 79

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

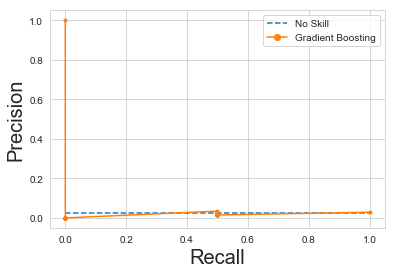
accuracy 0.94 79

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

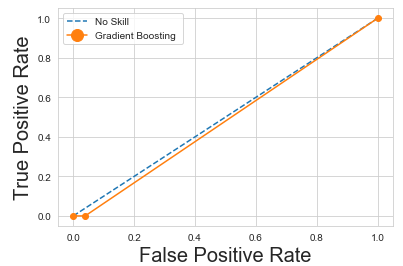
1 0.00 0.00 0.00 2

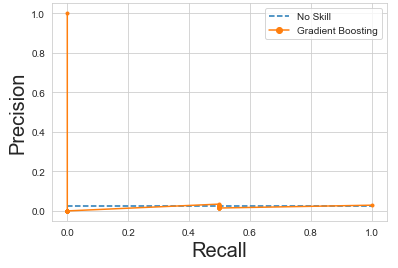
accuracy 0.94 79

macro avg 0.49 0.48 0.48 79

weighted avg 0.95 0.94 0.94 79

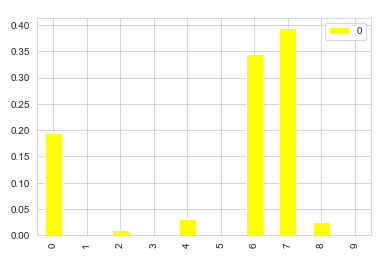
ROC and Precision-Recall Curve





* 1. **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

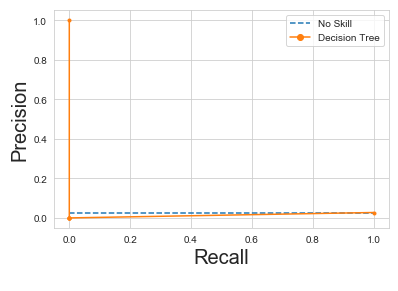
1 0.00 0.00 0.00 2

accuracy 0.96 79

macro avg 0.49 0.49 0.49 79

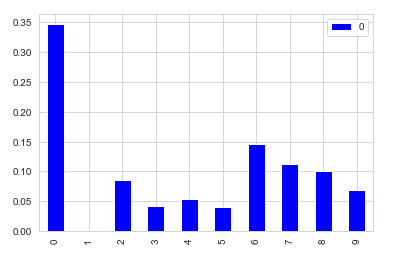
weighted avg 0.95 0.96 0.96 79

ROC and Precision-Recall Curve



* 1. **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

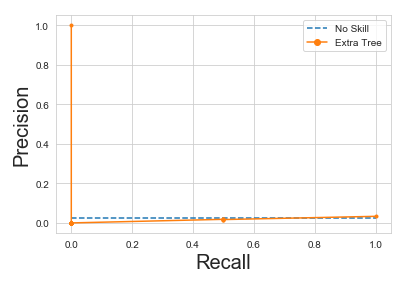
accuracy 0.97 79

macro avg 0.49 0.50 0.49 79

weighted avg 0.95 0.97 0.96 79

ROC and Precision-Recall Curve





* 1. **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

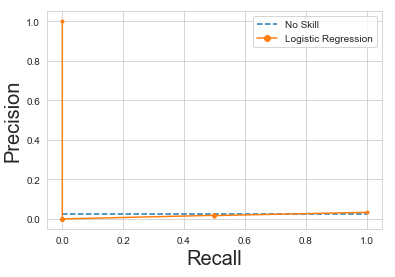
accuracy 0.97 79

macro avg 0.49 0.50 0.49 79

weighted avg 0.95 0.97 0.96 79

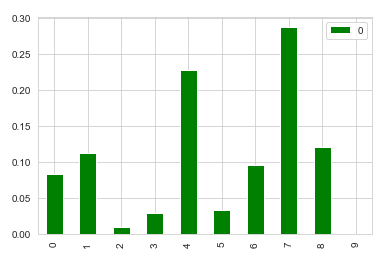
ROC and Precision-Recall Curve





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

# 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.

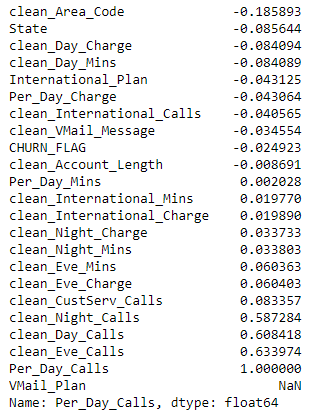
data['CHURN\_FLAG'] = data['Churn']

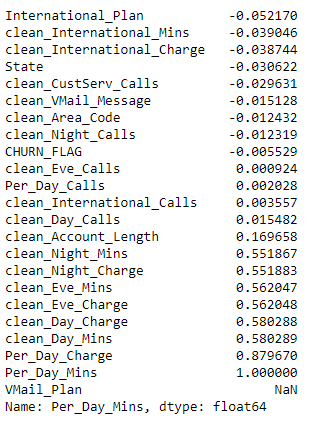
data.drop(labels='Churn',axis=1,inplace=True)

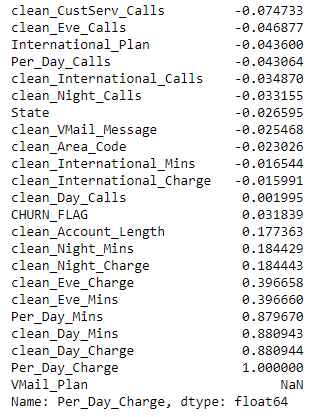
data.columns

data.dtypes

Feature selection







**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

* 1. **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

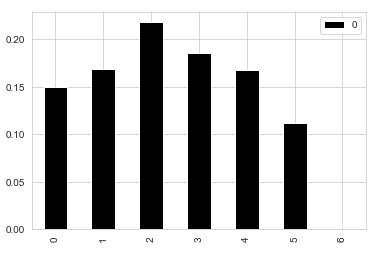
1 0.00 0.00 0.00 2

accuracy 0.94 79

macro avg 0.49 0.48 0.48 79

weighted avg 0.95 0.94 0.94 79

Feature Importance



* 1. **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

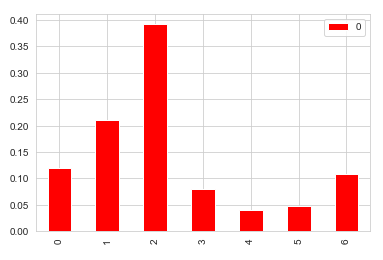
1 0.00 0.00 0.00 2

accuracy 0.94 79

macro avg 0.49 0.48 0.48 79

weighted avg 0.95 0.94 0.94 79

Feature Importance



* 1. **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

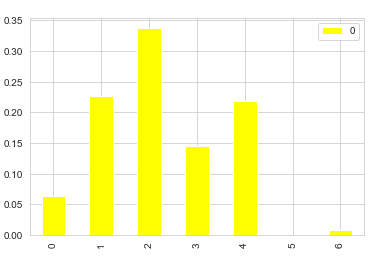
0 0.97 0.94 0.95 77

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



* 1. **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

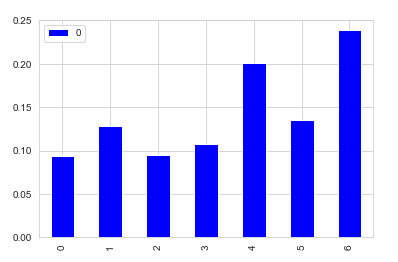
1 0.00 0.00 0.00 2

accuracy 0.97 79

macro avg 0.49 0.50 0.49 79

weighted avg 0.95 0.97 0.96 79

Feature Importance



* 1. **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:**

## Save the model in a file

from sklearn.externals import joblib

joblib.dump(rf, 'Telecom\_churn\_rf.ml')

## Load the model from the file

rf=joblib.load('Telecom\_churn\_rf.ml')

## Use the loaded model to make predictions

rf.predict(X\_test)

1. **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)

1. **Load the pickled model**

model\_from\_pickle=pickle.loads(saved\_model)

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

model\_from\_pickle.predict(X\_test)

Follow the similar procedure for the other algorithms.