CUSTOMER CHURN ANALYSIS PROJECT

SUBMITTED BY : SHAHLA M
Data Trained Institute

Article on Customer Churn Analysis

Intoduction :

Customer churn is the process by which a company's clients discontinue doing business with it. Because it is much less expensive to retain an existing customer than to acquire a new one, businesses are particularly interested in gauging churn. Working leads through a sales funnel and leveraging marketing and sales budgets to acquire new clients are key components of new business. Existing consumers frequently use more services and are more likely to recommend businesses to others.

But why does customer churn affect firms so much in the first place?

The quick answer is that it would be too expensive for customers to refuse to do business with you. Knowing what performs for them is crucial, but it's equally crucial—possibly even more crucial—to know what doesn't work and makes them churn. By examining trends, data, and other indicators, it reveals the proportion of customers who won't buy from you again or use your product in the future.

Let's look after some reasons of customer churn

The wrong types of customers are drawn to us.

- Our consumers are not succeeding in getting the results they want.
- Our customer service department needs improvement.
- Our clients believe that our rivals are capable of performing this task more effectively.
- Customers complain that our product has faults that we can't fix.
- The value of your product is no longer appreciated by our clients.
- Our clients believe your goods is overpriced (or too cheap).

We now understand that maintaining existing consumers is less expensive than acquiring new ones. Because, the cost of acquiring a customer, also known as customer acquisition cost, includes all sales and marketing expenses. Customers that have been around longer are more inclined to make larger purchases. They are using the product for a reason, and their on boarding experience has already helped them form a bond with the brand. Existing consumers are simpler to sell to because they are familiar with your business. Existing customers who value your product are more willing to upgrade features if it means an improved user experience. Concentrate on upselling to increase sales. In an effort to generate a more lucrative transaction, provide more features or upgrades.

An easy formula can be used to determine the customer churn rate. Divide the number of customers you lose over a certain time period by the total number of customers you had at the start of that time period to determine your customer churn rate. From there, multiply the result to get the percentage

Problem Definition :

IBM Customer Churn analysis & Performance Dataset:

In this problem, the enormous amounts of consumer data amassed can be used to create churn prediction algorithms. A corporation can focus its marketing efforts on the segment of its customer base that is most likely to defect by identifying that group. Given the low barriers to switching providers, preventing client churn is crucial for the telecommunications industry.

In order to develop and evaluate several customer churn prediction models, we will investigate customer data from IBM Sample Data Sets. You can also download dataset from the GitHub link here.

This Dataset has 7043 rows and 21 columns describing customer details which helps to predict customer retention. As target variable is categorical in nature, this case study falls into classification machine learning problem. We have two objectives here:

1. Which key factors result in customer churn?

2. Building ML Model for predicting churn. Data Analysis: Data Preparation: Load, Clean and Format:

Data Analysis

Data Preparation: Load, Clean and Format:

Let ‘s begin with importing necessary libraries fpr EDA and dataset itself.

```
]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from scipy.stats import zscore
from sklearn.preprocessing  import power_transform, StandardScaler, LabelEncoder
from sklearn.feature_selection import VarianceThreshold, SelectKBest, f_classif
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split, GridSearchCV,cross_val_score
from sklearn.linear_model import LogisticRegression, SGDRegressor,Ridge, Lasso
from sklearn.metrics import roc_curve, auc, roc_auc_score, accuracy_score,classification_report, confusion_matrix, plot_roc_curve
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
import pickle
import warnings
warnings.filterwarnings('ignore')
```

Reading the head of the data

```
churn.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupp
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

5 rows × 21 columns

```
churn.shape
```

```
(7043, 21)
```

Checking the data types.

```
In [6]: churn.dtypes

Out[6]: customerID          object
        gender              object
        SeniorCitizen        int64
        Partner             object
        Dependents          object
        tenure               int64
        PhoneService        object
        MultipleLines       object
        InternetService     object
        OnlineSecurity      object
        OnlineBackup        object
        DeviceProtection    object
        TechSupport         object
        StreamingTV         object
        StreamingMovies     object
        Contract            object
        PaperlessBilling    object
        PaymentMethod       object
        MonthlyCharges     float64
        TotalCharges        object
        Churn               object
        dtype: object
```

The data set contains object , integer and float datatype.

```
In [9]: churn.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

Among 21 columns, there are 1 float values, 18 are object types and 2 are int datatype. There is one target variable, Churn. Here customerID has different value for every different entries. Later drop this column. "SeniorCitizen" is a categorical variable as it has two different value, 0 and 1. Let's convert it into object datatype.

Cheking for the null values.

```
]: churn.isnull().sum()

]: customerID          0
   gender              0
   SeniorCitizen       0
   Partner             0
   Dependents          0
   tenure              0
   PhoneService        0
   MultipleLines       0
   InternetService     0
   OnlineSecurity      0
   OnlineBackup        0
   DeviceProtection    0
   TechSupport         0
   StreamingTV         0
   StreamingMovies     0
   Contract            0
   PaperlessBilling    0
   PaymentMethod       0
   MonthlyCharges      0
   TotalCharges        0
   Churn               0
   dtype: int64
```


```
: customerID          0
  gender              0
  SeniorCitizen       0
  Partner             0
  Dependents          0
  tenure              0
  PhoneService        0
  MultipleLines       0
  InternetService     0
  OnlineSecurity      0
  OnlineBackup        0
  DeviceProtection    0
  TechSupport         0
  StreamingTV         0
  StreamingMovies     0
  Contract            0
  PaperlessBilling    0
  PaymentMethod       0
  MonthlyCharges      0
  TotalCharges        0
  Churn               0
  dtype: int64
```


Checking whether the dataset contains any space

```
churn[churn['Churn'] == '']
```

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSuppo

0 rows × 21 columns

So we can see there are no spaces present in the dataset.

Checking for unique values

```
churn.nunique()
```

```
customerID          7043
gender                 2
SeniorCitizen          2
Partner                2
Dependents             2
tenure                73
PhoneService           2
MultipleLines          3
InternetService        3
OnlineSecurity         3
OnlineBackup           3
DeviceProtection       3
TechSupport            3
StreamingTV            3
StreamingMovies        3
Contract               3
PaperlessBilling       2
PaymentMethod          4
MonthlyCharges      1585
TotalCharges        6531
Churn                  2
dtype: int64
```

Descriptive Statistics

```
# Description of Dataset : works only on continuous column
churn.describe()
```

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

we can see that 3 column is containing continuous data and rest 18 column contains categorical data.

Data Cleaning and Pre processing

```
: churn['SeniorCitizen'].value_counts()
```

```
: 0    5901
  1    1142
  Name: SeniorCitizen, dtype: int64
```

```
: churn['SeniorCitizen'].unique()
```

```
: array([0, 1], dtype=int64)
```

```
: churn["SeniorCitizen"]= churn["SeniorCitizen"].map({0: "No", 1: "Yes"})
```

```
: churn['SeniorCitizen'].unique()
```

```
: array(['No', 'Yes'], dtype=object)
```

```
: #Filling Null Values of "TotalCharges" column in dataset by mean value
  churn["TotalCharges"].fillna(churn["TotalCharges"].mean(), inplace=True)
```

```
: churn[churn['tenure'] == 0].index
```

```
: Int64Index([488, 753, 936, 1082, 1340, 3331, 3826, 4380, 5218, 6670, 6754], dtype='int64')
```

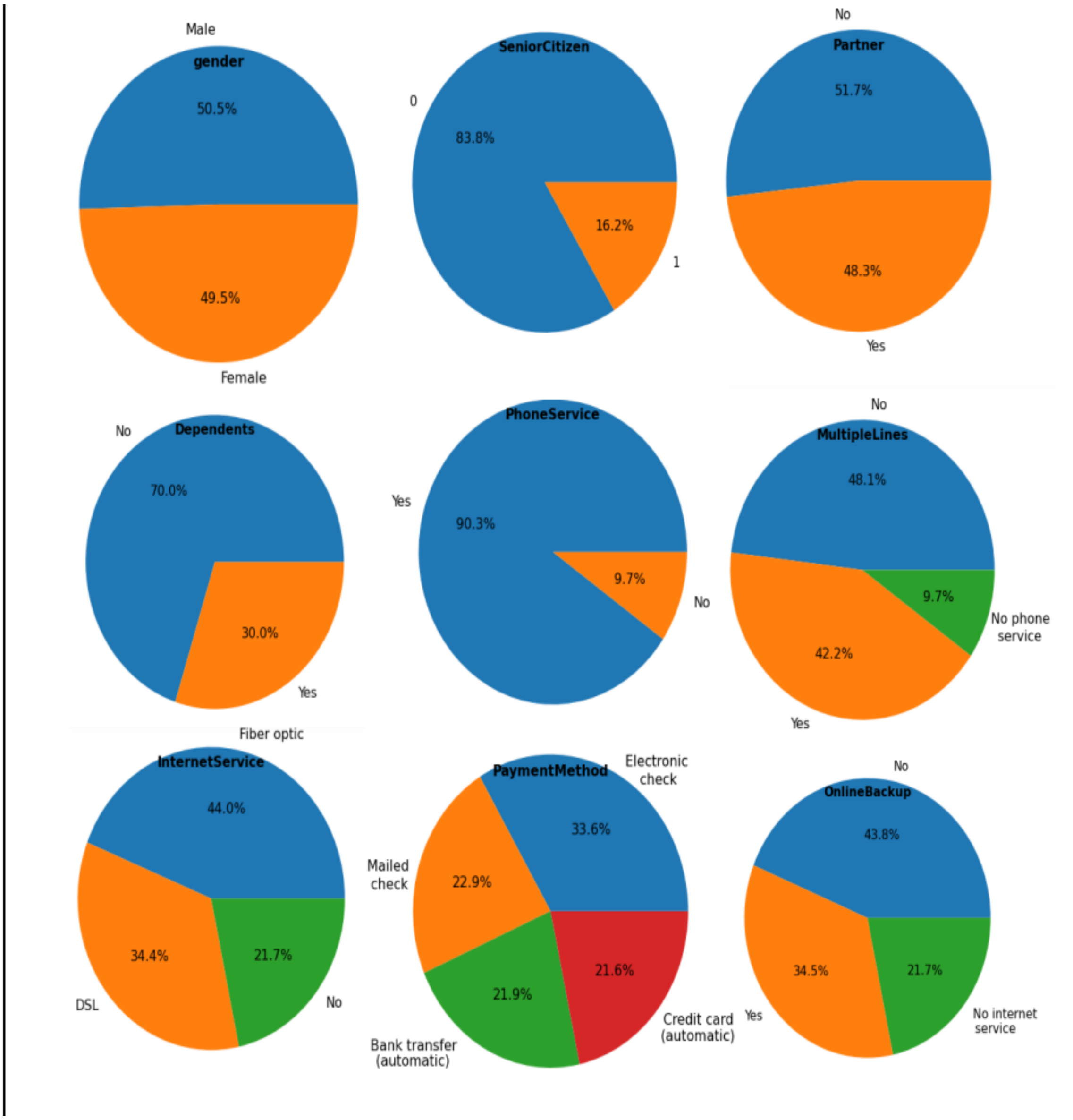
```
: #Droping/Deleting the rows with missing values in Tenure columns since there are only 11 rows and deleting them will not affect t

  churn.drop(labels=churn[churn['tenure'] == 0].index, axis=0, inplace=True)
  churn[churn['tenure'] == 0].index
```

```
: Int64Index([], dtype='int64')
```

Exploratory Data Analysis

Exploratory data analysis is the crucial procedure of doing first investigations on data in order to find patterns, uncover anomalies, test hypotheses, and double-check assumptions with the use of summary statistics and graphical representations.

Let’s begin data exploration of Categorical Data Analysis.



Observations from above plot:

1. Around 16% customer are Senior citizen

2. Around 50% customer are having partners.

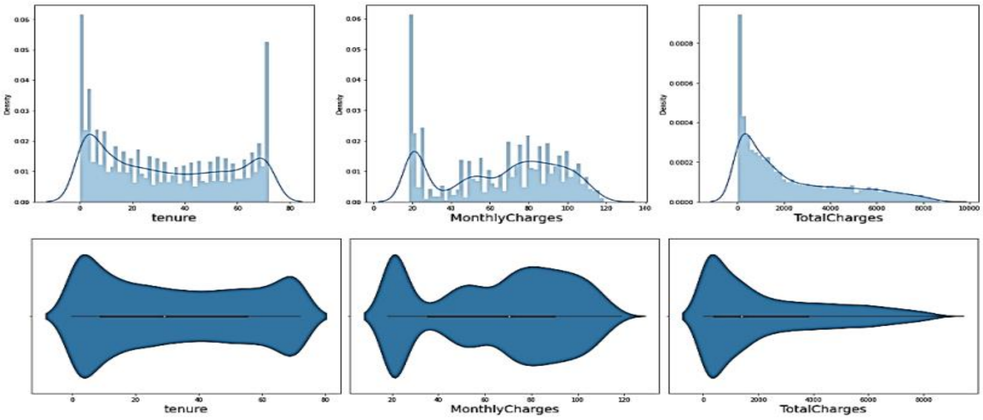
3. Around 30% customer have dependents on them

4. Almost 55% customer prefer month to month contract compare to other.

5. 60% Customer prefer paperless billing.

6. Most used payment method is electronic check.

Let's do analysis of Numerical Data variable:



Observations :

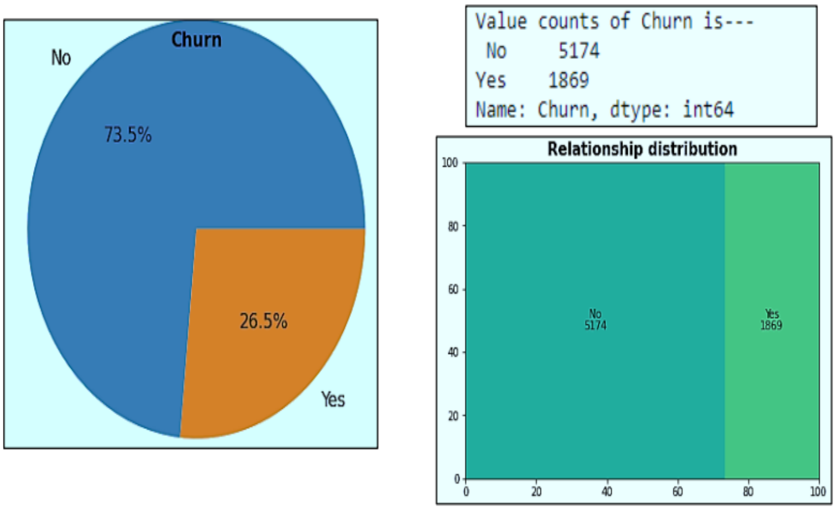
1. Average range of age is 0-70.

2. Monthly charges range is 20-120.

3. 0 value is present in TotalCharges column.

4. All the data have right skewness

*Let's analyse the **target variable**.*



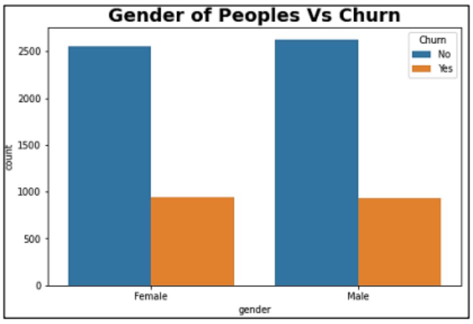
Observations from above plot :

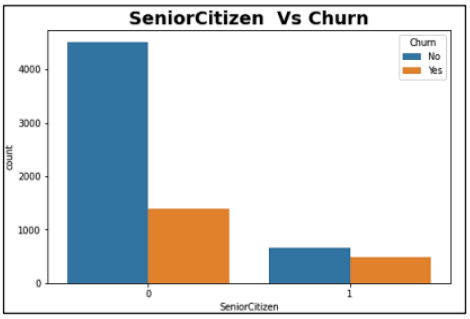
1. 73.5 % customers are not choose to Churn the service in last month.

2. 26.5 % customers are choose to Churn the service in last month.

3. The distribution of target variable is quite imbalance as there is a 75:25 relationship between NO:YES of tendency of churn.

Let's analyse the impact of different features on target variable.



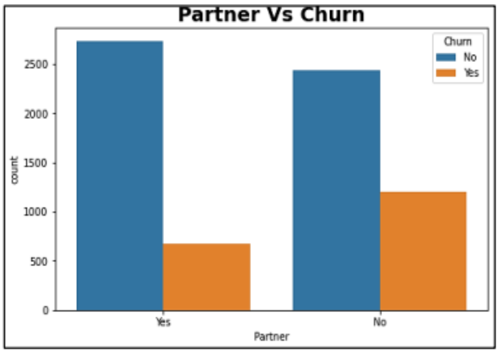


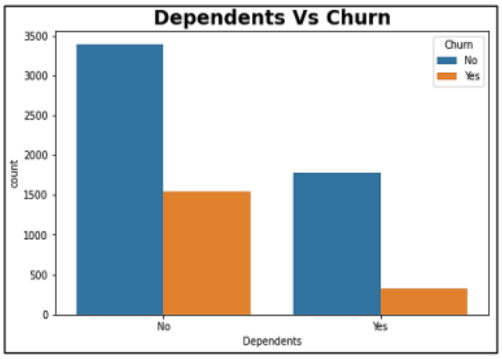
Observations from above plot:

1. In terms of gender, the distribution of Churn is in same proportion with minor difference.

2. For Male, YES: NO= 26:74 and for Female, YES: NO= 27:73

3. Senior citizen have more tendency to churn with respect to others.





Observations from above plot:

1. Customer having Partner have less tendency to Churn.

2. The customer not having partner have more tendency to Churn with respect to the customer who have their partner.

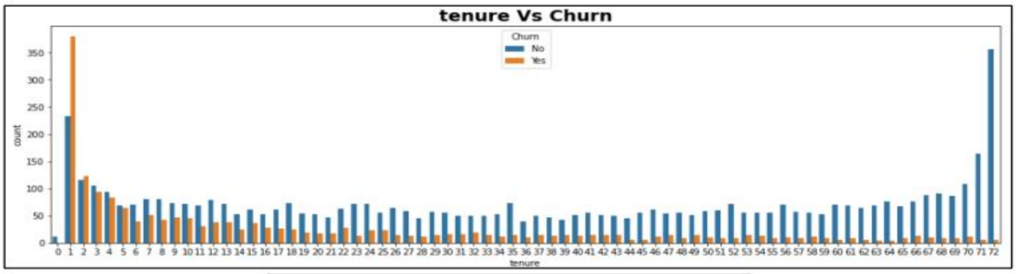
3. Only around 30% customers who have no dependents are tendency to Churn.

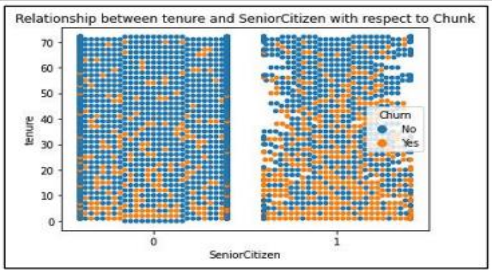
4. For all dependent customers around 85 % customers are more tendency to Churn

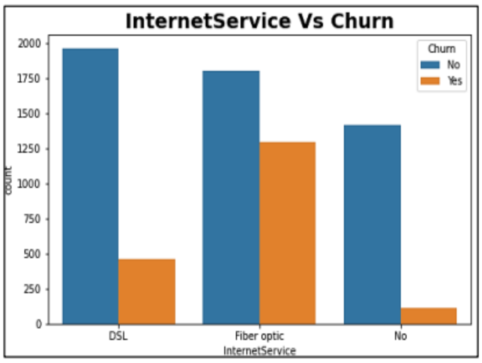
Observations from above plot:

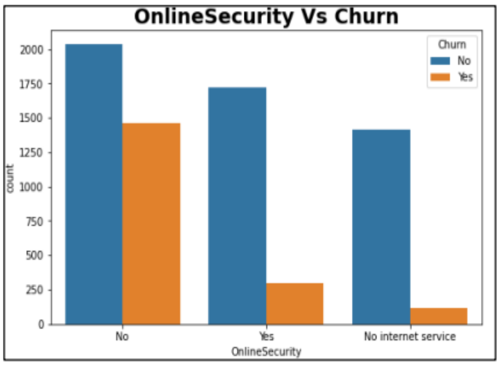
1. Here for the tenure 1, the number of customer with the tendency to Churn is much greater than the number of customer who have no tendency to Churn.

2. There is no clear relationship between SeniorCitizen and tenure.









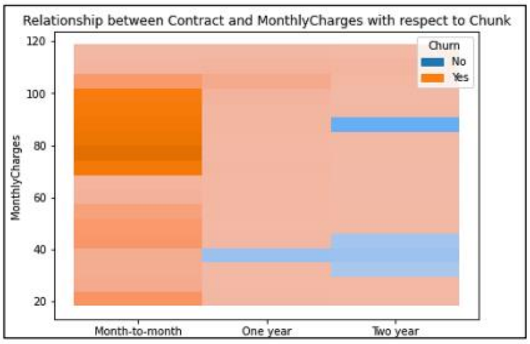
Observations from above plot:

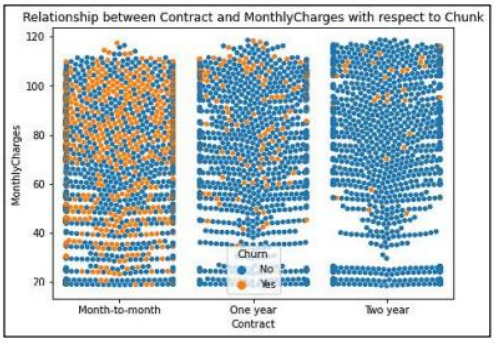
1. Mainly the positive Chunk are in the category with Fiber optic connection of internet service.

2. Out of total 3096 Fiber Optic connection, 1297 customer have tendency to Churn.

3. The maximum customer who have tendency to Churn are with No Online Security. no tech support (just like device protection) is more tendency to Chunk. 4. Churn tendency in people who streamingTV or not are same.

Plotting the Count Plot :

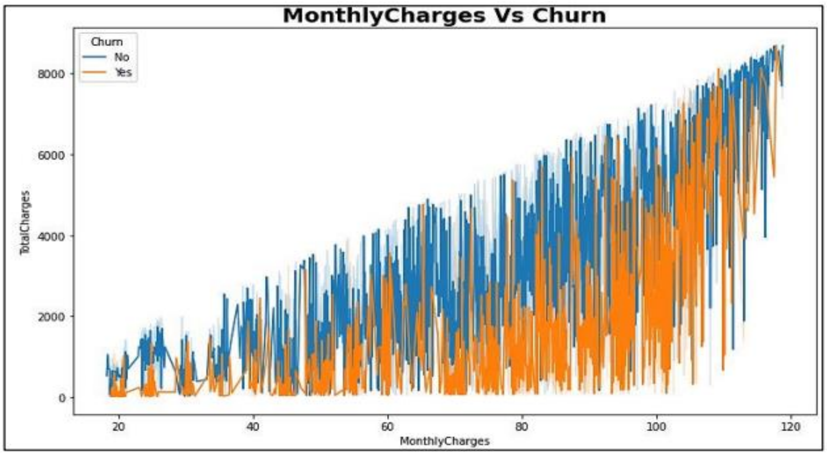




Observations from above plot:

1. If the contract type is month to month, there is a high churn rate in the customer.

2. No relation is found between MonthlyCharges and Contract.



Observations from above plot:

1. If Monthly Charges is high, then the customers are more tendence to choose churn compare to rest.

2. Also if Total Charges is high, then the customers are more tendence to choose churn compare to rest.

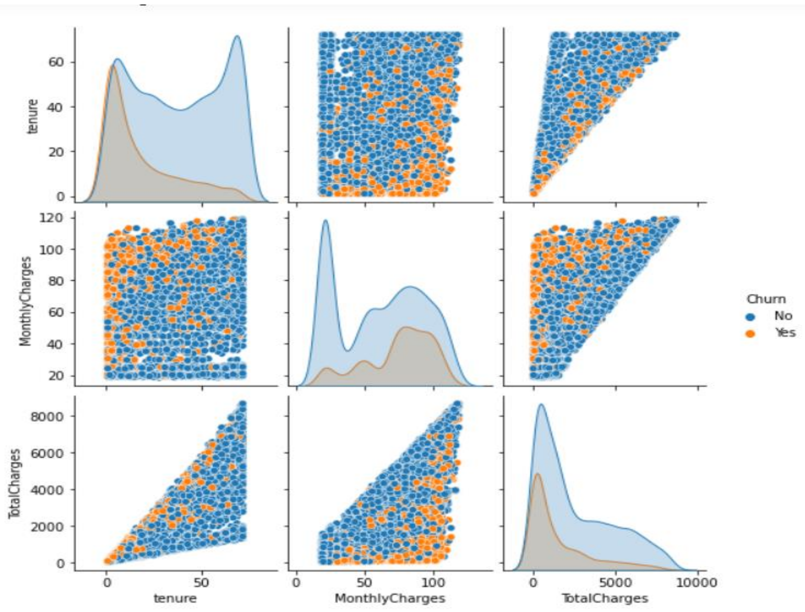
Pre-Processing Pipeline :

When developing a machine learning model, feature engineering is a crucial stage. Machine learning initiatives might be successful or unsuccessful. What distinguishes them? The features that are utilised are clearly the most crucial element. Feature engineering may be carried out for a number of reasons. Following are a few of them:

- Feature Extraction: Automatic generation of new features from unprocessed data (Dimensionality reduction Technique like PCA).
- Feature Importance: An evaluation of a feature's usefulness.
- Feature Selection: From many features to a few that are useful Depending on the needs of the dataset, a variety of strategies are used to obtain the above results. Effective methods include the following:
- Handling missing values
- Encoding categorical data using one hot encoding, label / ordinal encoding.
- Correlation between different features and target variable.
- Outliers' detection and removal using Z-score, IQR
- Skewness correction using Box-cox or yeo-Johnson method
- Handling imbalanced data using SMOTE
- Checking Multicollinearity among feature using variance inflation factor(VIF)
- If Multicollinearity present, checking Principal Component Analysis (PCA).
- Scaling of data using Standard Scalar.

We will employ some of the above-mentioned feature engineering techniques in this case study, one at a time.

Multivariate Analysis



- *We can observe relationship between all the continuous column and the target column by this pairplot in pairs which are plotted on basis of target column.*

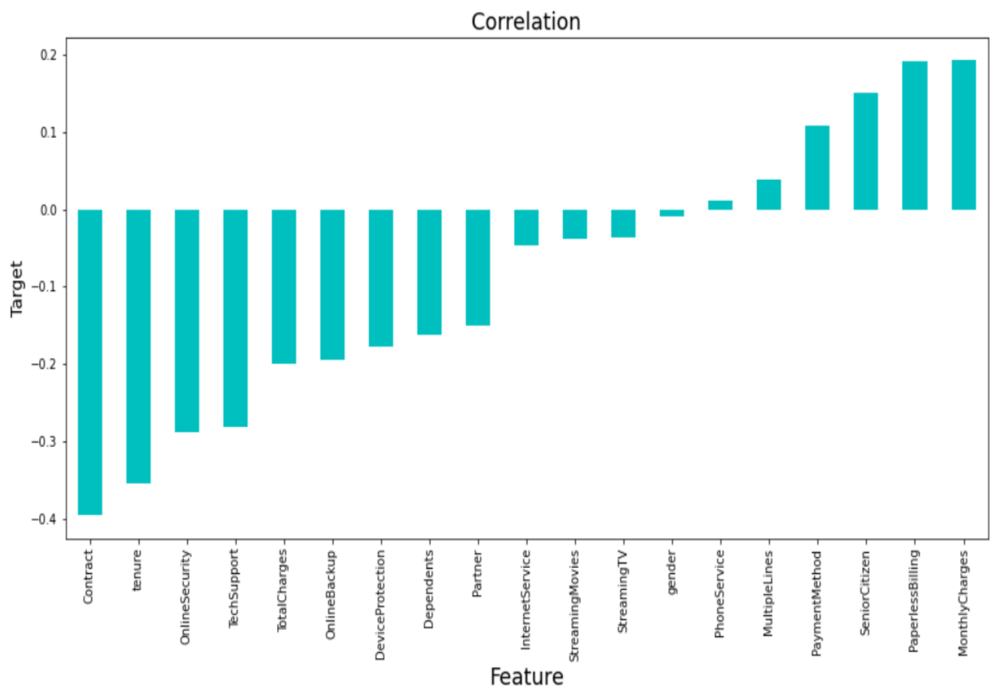
1. Encoding categorical data using label encoding

```
In [81]: from sklearn.preprocessing import LabelEncoder

In [82]: enc = LabelEncoder()
         for i in churn.columns:
             if churn[i].dtypes=="object":
                 churn[i]=enc.fit_transform(churn[i].values.reshape(-1,1))
```

This gives the correlation between the dependent and independent variables.

2. Correlation :

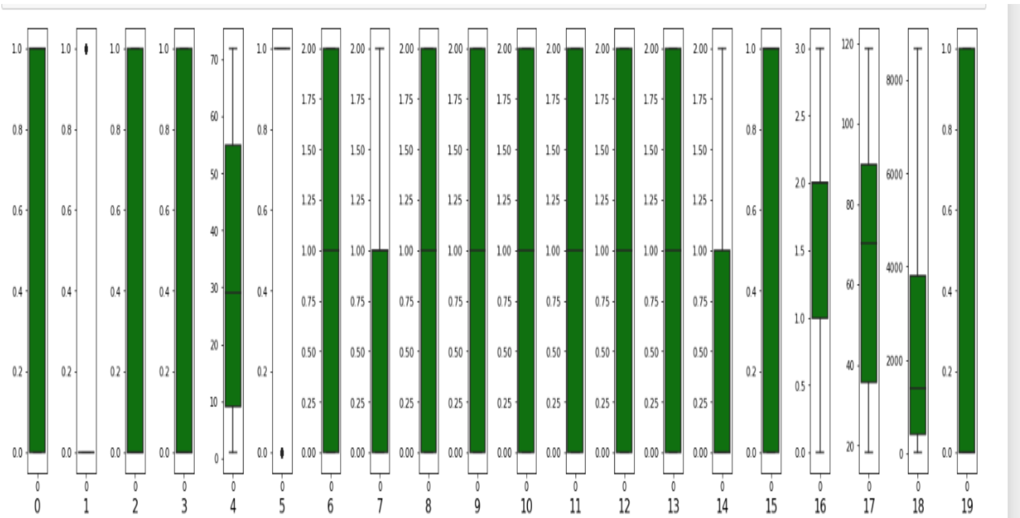
The Correlation Heat map informs us of potential multicollinearity issues by quickly displaying which variables are connected, to what extent, and in which direction. Below is a bar plot showing the target variable's correlation coefficient with independent features.



Observations from above plot:

1. Churn has a highly negative relationship with Contract.
2. In other hand, paperless billing and monthly charges are positively correlated with churn.
3. All the features are correlated with each other.

3. Outlier Detection of data and removal:



Observations :

- We can see Outliers are present only in 2 columns: "SeniorCitizen" and "PhoneService". But both column are categorical, so we will not remove outliers.

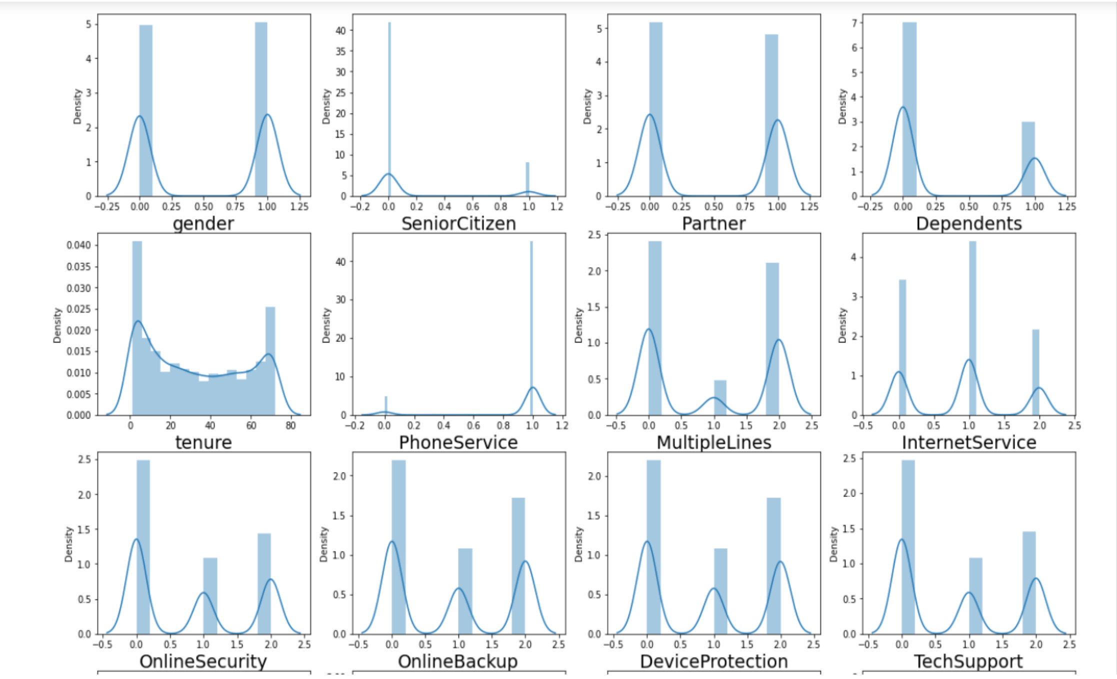
4.Skewness :

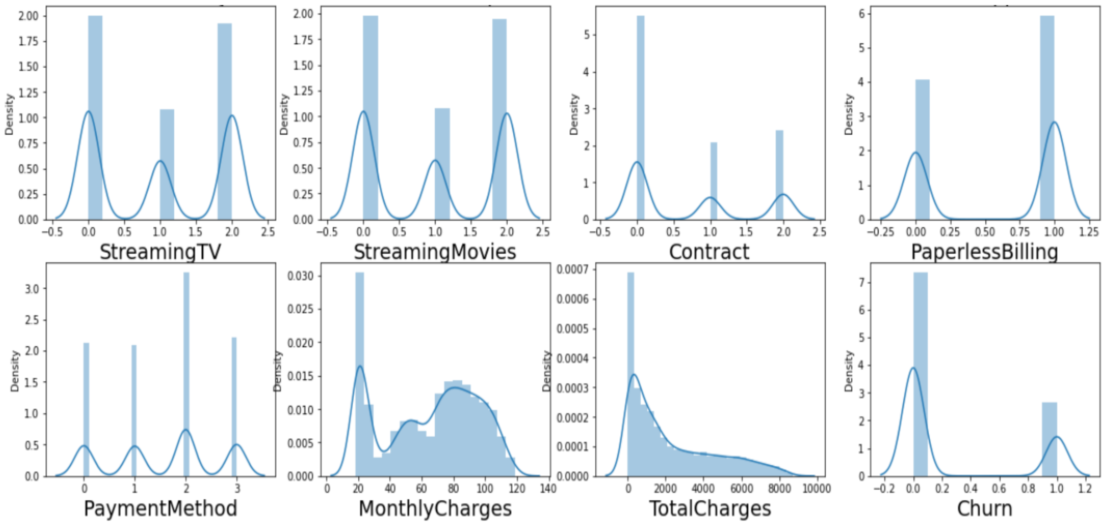
```
In [90]:  churn.skew()

Out[90]:  gender              -0.018776
          SeniorCitizen        1.831103
          Partner              0.070024
          Dependents           0.880908
          tenure               0.237731
          PhoneService        -2.729727
          MultipleLines        0.118623
          InternetService      0.205704
          OnlineSecurity       0.418619
          OnlineBackup         0.184089
          DeviceProtection     0.188013
          TechSupport          0.403966
          StreamingTV          0.029366
          StreamingMovies      0.013851
          Contract             0.635149
          PaperlessBilling    -0.377503
          PaymentMethod       -0.169388
          MonthlyCharges      -0.222103
          TotalCharges         0.961642
          Churn                1.060622
          dtype: float64
```

Observations :

- Skewness threshold taken is +/-0.25
- All the columns are not normally distributed, they are skewed. Columns which are having skewness: Senior Citizen, Dependents, Phone Service, Online Security, Tech Support, Contract, Paper less Billing and Total Charges.
- Since Senior Citizen, Dependents, Phone Service, Online Security, Tech Support, Contract and Paper less Billing are categorical column so we will not remove skewness from them.
- Only we will remove skewness from Total Charges as this column contains continuous data.

Data Visualization of Skewness :





Removing skewness using yeo-john son method :

```
from sklearn.preprocessing import PowerTransformer
```

```
collist=['TotalCharges']
churn[collist]=power_transform(churn[collist],method='yeo-johnson')
churn[collist]
```

	TotalCharges
0	-1.810069
1	0.254257
2	-1.386091
3	0.233220
4	-1.248808
...	...
7038	0.296583
7039	1.565846
7040	-0.858393
7041	-0.921477
7042	1.483370

7032 rows × 1 columns

Checking after Skewness :

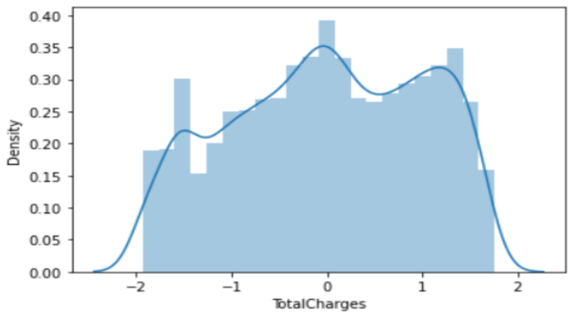
```
churn.skew()
```

```
gender              -0.018776
SeniorCitizen        1.831103
Partner              0.070024
Dependents           0.880908
tenure               0.237731
PhoneService        -2.729727
MultipleLines        0.118623
InternetService      0.205704
OnlineSecurity       0.418619
OnlineBackup         0.184089
DeviceProtection     0.188013
TechSupport          0.403966
StreamingTV          0.029366
StreamingMovies      0.013851
Contract             0.635149
PaperlessBilling    -0.377503
PaymentMethod       -0.169388
MonthlyCharges      -0.222103
TotalCharges        -0.144643
Churn                1.060622
dtype: float64
```

Checking skewness after removal through data visualization using dist plot :

```
sns.distplot(churn['TotalCharges'])
```

```
<AxesSubplot:xlabel='TotalCharges', ylabel='Density'>
```



Observations :

The data is not normal but the skewness has got removed compared to the old data.

Data Preprocessing :

```
x=churn.drop("Churn",axis=1)
y=churn["Churn"]
```

```
x.columns
```

```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges'],
      dtype='object')
```

Observations :

The data is not balanced. So, we will use oversampling method to balance it.

Oversampling using the SMOTE :

```
from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE
```

```
SM = SMOTE()
x, y = SM.fit_resample(x,y)
y.value_counts()
```

```
0    5163
1    5163
Name: Churn, dtype: int64
```

Scaling data using Standard Scaler :

```
scaler = StandardScaler()
```

```
x = pd.DataFrame(scaler.fit_transform(x), columns = x.columns)
```

```
x.head()
```

```
x.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	Tec
0	-0.891916	-0.412414	1.269462	-0.528729	-1.113521	-3.010702	0.053467	-1.264963	-0.757727	1.410296	-0.897508	
1	1.121182	-0.412414	-0.787736	-0.528729	0.262587	0.332148	-1.010312	-1.264963	1.694644	-0.892292	1.397321	
2	1.121182	-0.412414	-0.787736	-0.528729	-1.071821	0.332148	-1.010312	-1.264963	1.694644	1.410296	-0.897508	
3	1.121182	-0.412414	-0.787736	-0.528729	0.721290	-3.010702	0.053467	-1.264963	1.694644	-0.892292	1.397321	
4	-0.891916	-0.412414	-0.787736	-0.528729	-1.071821	0.332148	-1.010312	0.214648	-0.757727	-0.892292	-0.897508	

```
x
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
0	-0.891916	-0.412414	1.269462	-0.528729	-1.113521	-3.010702	0.053467	-1.264963	-0.757727	1.410296	-0.897508
1	1.121182	-0.412414	-0.787736	-0.528729	0.262587	0.332148	-1.010312	-1.264963	1.694644	-0.892292	1.397321
2	1.121182	-0.412414	-0.787736	-0.528729	-1.071821	0.332148	-1.010312	-1.264963	1.694644	1.410296	-0.897508
3	1.121182	-0.412414	-0.787736	-0.528729	0.721290	-3.010702	0.053467	-1.264963	1.694644	-0.892292	1.397321
4	-0.891916	-0.412414	-0.787736	-0.528729	-1.071821	0.332148	-1.010312	0.214648	-0.757727	-0.892292	-0.897508
...	...	...	...	...	...	...	...	...	...	...	...
10321	-0.891916	-0.412414	-0.787736	-0.528729	-1.113521	-3.010702	0.053467	-1.264963	-0.757727	-0.892292	-0.897508
10322	-0.891916	-0.412414	-0.787736	-0.528729	-1.030121	-3.010702	0.053467	-1.264963	0.468459	-0.892292	-0.897508
10323	-0.891916	2.424750	-0.787736	-0.528729	0.637890	0.332148	0.053467	0.214648	-0.757727	1.410296	0.249907

Variance Threshold Method :

It removes all features which variance doesn't meet some threshold. By default, it removes all zero-variance features

```
var_threshold = VarianceThreshold(threshold=0)
var_threshold.fit(x)
```

```
VarianceThreshold(threshold=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
var_threshold.get_support()
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True])
```

```
x.columns[var_threshold.get_support()]
```

```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges'],
      dtype='object')
```

Observations :

- So we can see that, with the help of variance threshold method, we got to know all the features here are important. So now we will check through Select KBest method.

SelectKBest method :

```
from sklearn.feature_selection import  SelectKBest, f_classif
```

```
best_fit = SelectKBest(score_func = f_classif, k ='all')
fit = best_fit.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
```

```
fit = best_fit.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
dfcolumns.head()
featureScores = pd.concat([dfcolumns,dfscores],axis = 1)
featureScores.columns = ['Feature', 'Score']
print(featureScores.nlargest(12,'Score'))
```

Observations :

Selecting the best features based on above scores, we can see that the column "gender" has most lowest features for the prediction, so we will drop this column.

Checking for Multicollinearity using Variance Inflation Factor :

VIF (Variance Inflation factor)¶

```
vif = pd.DataFrame()
vif['VIF values']= [variance_inflation_factor(x.values,i) for i in range(len(x.columns))]
vif['Features'] = x.columns
vif
```

	VIF values	Features
0	1.096229	SeniorCitizen
1	1.534895	Partner
2	1.431463	Dependents
3	11.908024	tenure
4	1.746379	PhoneService
5	1.413832	MultipleLines
6	1.765931	InternetService
7	1.348203	OnlineSecurity
8	1.236171	OnlineBackup
9	1.314740	DeviceProtection
10	1.395731	TechSupport
11	1.502428	StreamingTV
12	1.479420	StreamingMovies
13	2.631163	Contract
14	1.167333	PaperlessBilling
15	1.178887	PaymentMethod
16	4.418548	MonthlyCharges
17	13.509013	TotalCharges

Observations :

The VIF value is more than 10 in the columns 'tenure' and 'Total Charges'. But column 'Total Charges' is having highest VIF value. So, we will drop column 'Total Charges'.

Checking again Multicolinearity using VIF

```
vif = pd.DataFrame()
vif['VIF values']= [variance_inflation_factor(x.values,i) for i in range(len(x.columns))]
vif['Features'] = x.columns
vif
```

	VIF values	Features
0	1.096160	SeniorCitizen
1	1.533631	Partner
2	1.428811	Dependents
3	2.802302	tenure
4	1.746140	PhoneService
5	1.406814	MultipleLines
6	1.739261	InternetService
7	1.342647	OnlineSecurity
8	1.235337	OnlineBackup
9	1.312849	DeviceProtection
10	1.387621	TechSupport
11	1.501616	StreamingTV
12	1.478149	StreamingMovies
13	2.472397	Contract
14	1.167085	PaperlessBilling
15	1.176577	PaymentMethod
16	2.722763	MonthlyCharges

Observations :

Now, we can check Multi collinearity is removed from the columns as VIF value of all columns are less than 10. So, we will create model now.

Machine Learning Model Building:

In this section we will build Supervised learning ML model-based classification algorithm. train_test_split used to split data with size of 0.25. Let's find best Random state.

```
maxAccu=0
maxRS=0
for i in range(1,100):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state =i)
    DTC = DecisionTreeClassifier()
    DTC.fit(x_train, y_train)
    pred = DTC.predict(x_test)
    acc=accuracy_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu," on Random_state ",maxRS)
```

```
Best accuracy is  0.8050355067785668  on Random_state  85
```

Observations :

At random state 85 , we are getting best accuracy score i.e., 80%.

1. Logistic Regression :

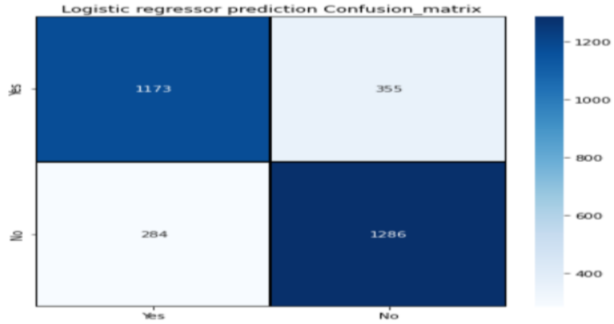
The evaluation matrix along with the Confusion matrix for Logistic Regression.

```
lr=LogisticRegression()
lr.fit(x_train,y_train)
pred_lr=lr.predict(x_test)

print("accuracy_score: ", accuracy_score(y_test, pred_lr))
print("confusion_matrix: \n", confusion_matrix(y_test, pred_lr))
print("classification_report: \n", classification_report(y_test,pred_lr))
```

```
accuracy_score:  0.7937378954163977
confusion_matrix:
 [[1173  355]
 [ 284 1286]]
classification_report:
               precision    recall  f1-score   support

           0       0.81      0.77      0.79      1528
           1       0.78      0.82      0.80      1570

    accuracy                           0.79      3098
   macro avg       0.79      0.79      0.79      3098
weighted avg       0.79      0.79      0.79      3098
```

Confusion Matrix for Logistic Regression :

```
cm = confusion_matrix(y_test,pred_lr)
x_axis_labels = ["Yes","No"]
y_axis_labels = ["Yes","No"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("Logistic regressor prediction Confusion_matrix")
```



Observations :

Here we are getting 79% accuracy using Logistic Regression.

Classification Algorithms :

2.Random Forest Classifier :

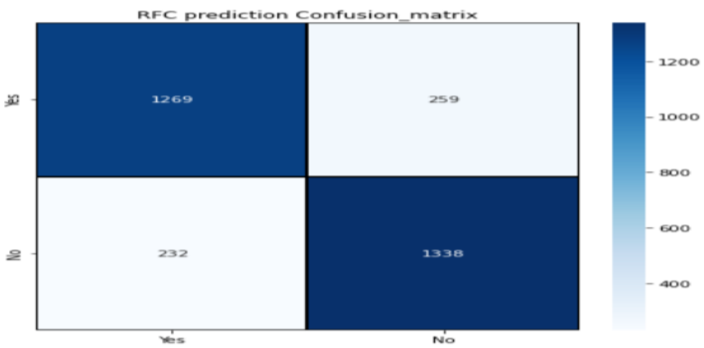
The evaluation matrix along with the Confusion matrix for Random Classifier

```
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(x_train,y_train)
pred_rfc = rfc.predict(x_test)

print("accuracy_score: ",accuracy_score(y_test, pred_rfc))
print("confusion_matrix: \n",confusion_matrix(y_test, pred_rfc))
print("classification_report: \n",classification_report(y_test,pred_rfc))
```

```
accuracy_score:  0.84151065203357
confusion_matrix:
 [[1269  259]
 [ 232 1338]]
classification_report:
               precision    recall  f1-score   support

           0       0.85      0.83      0.84      1528
           1       0.84      0.85      0.84      1570

    accuracy                           0.84      3098
   macro avg       0.84      0.84      0.84      3098
weighted avg       0.84      0.84      0.84      3098
```

Confusion Matrix for Random Forest Classifier :

```
cm = confusion_matrix(y_test,pred_rfc)
x_axis_labels = ["Yes","No"]
y_axis_labels = ["Yes","No"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("RFC prediction Confusion_matrix")
```



Observations :

Here we are getting 84% accuracy using Random Forest Classifier.

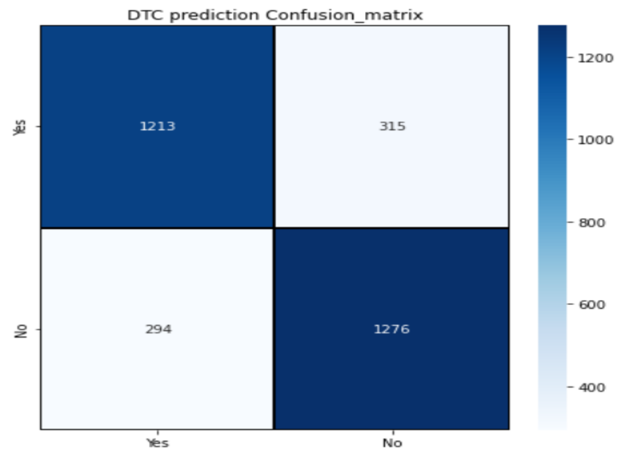
Decision Tree Classifier :

```
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)
pred_dtc = dtc.predict(x_test)

print("accuracy_score: ",accuracy_score(y_test, pred_dtc))
print("confusion_matrix: \n",confusion_matrix(y_test, pred_dtc))
print("classification_report: \n",classification_report(y_test,pred_dtc))
```

```
accuracy_score:  0.8034215622982569
confusion_matrix:
 [[1213  315]
 [ 294 1276]]
classification_report:
               precision    recall  f1-score   support

           0       0.80      0.79      0.80      1528
           1       0.80      0.81      0.81      1570

    accuracy                           0.80      3098
   macro avg       0.80      0.80      0.80      3098
weighted avg       0.80      0.80      0.80      3098
```

Confusion Matrix for Decision Tree Classifier :

```
cm = confusion_matrix(y_test,pred_rfc)
x_axis_labels = ["Yes","No"]
y_axis_labels = ["Yes","No"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("RFC prediction Confusion_matrix")
```



Observations :

Here we are getting 79% accuracy using Decision Tree Classifier

Support Vector Machine Classifier :

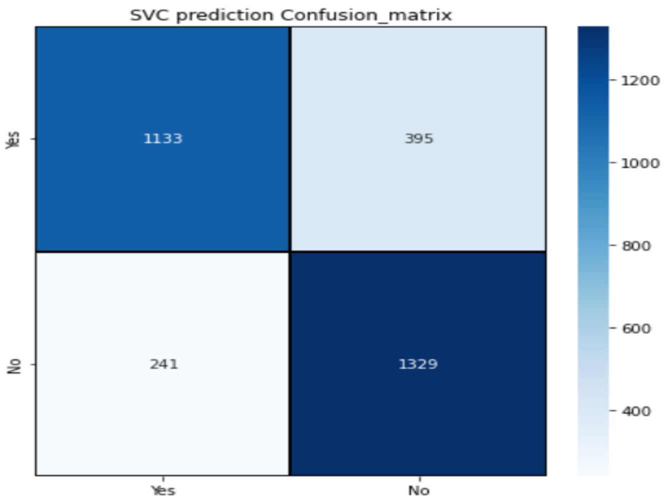
```
]: svc = SVC(kernel='linear', gamma=3)
   svc.fit(x_train,y_train)
   pred_svc = svc.predict(x_test)

   print("accuracy_score: ", accuracy_score(y_test, pred_svc))
   print("confusion_matrix: \n", confusion_matrix(y_test, pred_svc))
   print("classification_report: \n", classification_report(y_test,pred_svc))
```

```
accuracy_score:  0.7947062621045836
confusion_matrix:
 [[1133  395]
 [ 241 1329]]
classification_report:
               precision    recall  f1-score   support

           0       0.82      0.74      0.78      1528
           1       0.77      0.85      0.81      1570

    accuracy                           0.79      3098
   macro avg       0.80      0.79      0.79      3098
weighted avg       0.80      0.79      0.79      3098
```

Confusion Matrix for SVC :

```
cm = confusion_matrix(y_test,pred_svc)
x_axis_labels = ["Yes","No"]
y_axis_labels = ["Yes","No"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("SVC prediction Confusion_matrix")
```

```
Text(0.5, 1.0, 'SVC prediction Confusion_matrix')
```



Observations :

Here we are getting 78% accuracy using Support Vector Machine Classifier.

KNN Classifier :

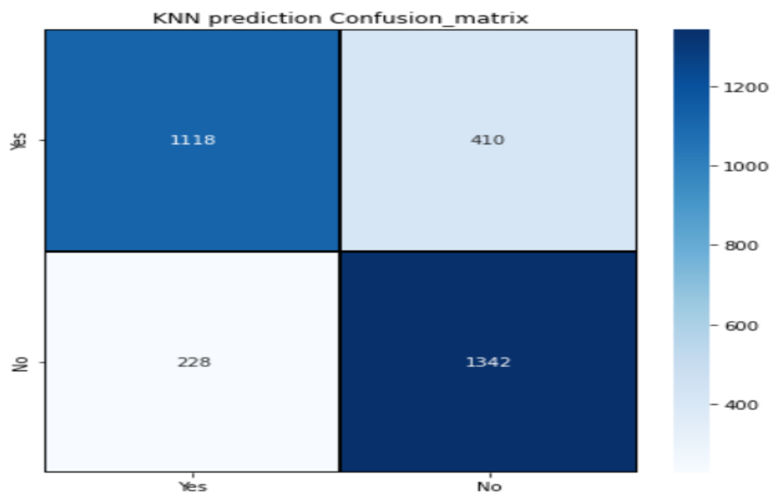
```
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
pred_knn = knn.predict(x_test)

print("accuracy_score: ",accuracy_score(y_test, pred_knn))
print("confusion_matrix: \n",confusion_matrix(y_test, pred_knn))
print("classification_report: \n",classification_report(y_test,pred_knn))
```

```
accuracy_score:  0.7940606843124597
confusion_matrix:
 [[1118  410]
 [ 228 1342]]
classification_report:
               precision    recall  f1-score   support

           0       0.83      0.73      0.78      1528
           1       0.77      0.85      0.81      1570

    accuracy                           0.79      3098
   macro avg       0.80      0.79      0.79      3098
weighted avg       0.80      0.79      0.79      3098
```

Confusion Matrix for KNN Classifier :

```
cm = confusion_matrix(y_test,pred_knn)
x_axis_labels = ["Yes","No"]
y_axis_labels = ["Yes","No"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("KNN prediction Confusion_matrix")
```

```
Text(0.5, 1.0, 'KNN prediction Confusion_matrix')
```



Observations :

Here we are getting 79% accuracy using KNN.

Gradient Boosting Classifier :

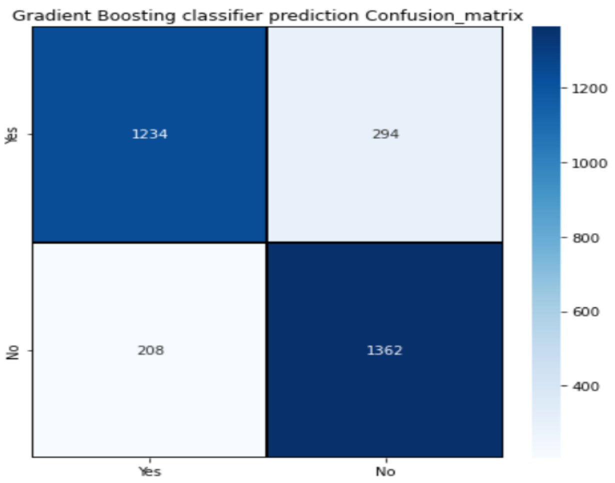
```
gb = GradientBoostingClassifier(n_estimators =100,learning_rate=0.1, max_depth=4)
gb.fit(x_train,y_train)
pred_gb = gb.predict(x_test)

print("accuracy_score: ",accuracy_score(y_test, pred_gb))
print("confusion_matrix: \n",confusion_matrix(y_test, pred_gb))
print("classification_report: \n",classification_report(y_test,pred_gb))
```

```
accuracy_score:  0.8379599741768883
confusion_matrix:
 [[1234  294]
 [ 208 1362]]
classification_report:
               precision    recall  f1-score   support

           0       0.86      0.81      0.83      1528
           1       0.82      0.87      0.84      1570

    accuracy                           0.84      3098
   macro avg       0.84      0.84      0.84      3098
weighted avg       0.84      0.84      0.84      3098
```

Confusion Matrix for Gradient Boost Classifier:

```
cm = confusion_matrix(y_test,pred_gb)
x_axis_labels = ["Yes","No"]
y_axis_labels = ["Yes","No"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("Gradient Boosting classifier prediction Confusion_matrix")
```

```
Text(0.5, 1.0, 'Gradient Boosting classifier prediction Confusion_matrix')
```



Observations :¶

Here we are getting 83% accuracy using Gradient Boosting classifier.

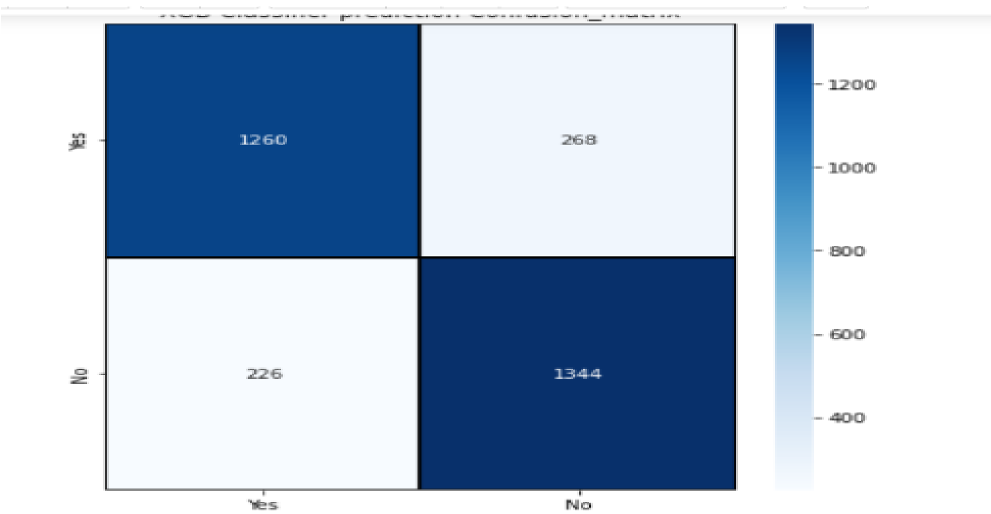
XGB Classifier :

```
]: XGBC= XGBClassifier()
XGBC.fit(x_train,y_train)
pred_XGBC = XGBC.predict(x_test)

print("accuracy_score: ",accuracy_score(y_test, pred_XGBC))
print("confusion_matrix: \n",confusion_matrix(y_test, pred_XGBC))
print("classification_report: \n",classification_report(y_test,pred_XGBC))
```

```
accuracy_score:  0.8405422853453841
confusion_matrix:
 [[1260  268]
 [ 226 1344]]
classification_report:
               precision    recall  f1-score   support

           0       0.85      0.82      0.84      1528
           1       0.83      0.86      0.84      1570

    accuracy                           0.84      3098
   macro avg       0.84      0.84      0.84      3098
weighted avg       0.84      0.84      0.84      3098
```

```
: cm = confusion_matrix(y_test,pred_XGBC)
x_axis_labels = ["Yes","No"]
y_axis_labels = ["Yes","No"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("XGB Classifier prediction Confusion_matrix")
```

```
: Text(0.5, 1.0, 'XGB Classifier prediction Confusion_matrix')
```



Observations :

Here we are getting 84% accuracy using XGB Classifier.

Cross Validation :

Now apply Cross Validation method and check the best model by selecting maximum score of cross validation and minimum standard deviation value. Here we can see that, Random Forest Classifier gives the best score.

```
#CV Score for Logistic Regression
print('CV score for Logistic Regression: ',cross_val_score(lr,x,y,cv=5).mean())

#CV Score for Random Forest Classifier
print('CV score for Random forest Classifier: ',cross_val_score(rfc,x,y,cv=5).mean())

#CV Score for Decision Tree Classifier
print('CV score for Decision Tree Classifier: ',cross_val_score(dtc,x,y,cv=5).mean())

#CV Score for Support Vector Classifier
print('CV score for Support Vector  Classifier: ',cross_val_score(svc,x,y,cv=5).mean())

#CV Score for KNN Classifier
print('CV score for KNN Classifier: ',cross_val_score(knn,x,y,cv=5).mean())

#CV Score for Gradient Boosting Classifier
print('CV score for Gradient Boosting Classifier: ',cross_val_score(gb,x,y,cv=5).mean())

#CV Score for XGB Classifier
print('CV score for XGB Classifier: ',cross_val_score(XGBC,x,y,cv=5).mean())
```

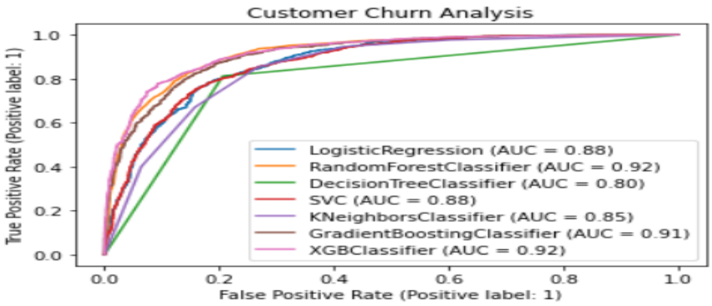
```
CV score for Logistic Regression:  0.7856896741665474
CV score for Random forest Classifier:  0.8398285161111879
CV score for Decision Tree Classifier:  0.7910198790987016
CV score for Support Vector  Classifier:  0.7808465434839169
CV score for KNN Classifier:  0.7736825204100051
CV score for Gradient Boosting Classifier:  0.8190075217577804
CV score for XGB Classifier:  0.8335354605523768
```

AUC-ROC Curve:

Now check AUC-ROC curve.

The receiver operating characteristic curve (ROC curve) is a graph that displays how well a classification model performs across all categorization levels. The term "Area under the ROC Curve" (AUC) refers to measuring the complete two-dimensional area beneath the entire ROC curve. Here also we plot AUC- ROC curve and choose the best model by maximum area under the curve.

```
#Lets plot roc curve and check auc and performance of all algorithms
disp = plot_roc_curve(lr, x_test, y_test)
plot_roc_curve(rfc, x_test, y_test, ax = disp.ax_)
plot_roc_curve(dtc, x_test, y_test, ax = disp.ax_)
plot_roc_curve(svc, x_test, y_test, ax = disp.ax_)
plot_roc_curve(knn, x_test, y_test, ax = disp.ax_)
plot_roc_curve(gb, x_test, y_test, ax = disp.ax_)
plot_roc_curve(XGBC, x_test, y_test, ax = disp.ax_)
plt.title("Customer Churn Analysis")
plt.legend(prop={"size" :10} ,loc = 'lower right')
plt.show()
```



Observations :

Here , RandomForest Classifier and XGB C lassifier gives the best model with an accuracy of 0.92.

Hyper parameter tuning for best model using GridsearchCV :

The XGB Classifier with GridsearchCV:

Checking the Accuracy of the model

```
from sklearn.model_selection import KFold
```

```
params = {
    'n_estimators': [100, 200, 500],
    'learning_rate': [0.01,0.05,0.1],
    'booster': ['gbtree', 'gblinear'],
    'gamma': [0, 0.5, 1],
    'reg_alpha': [0, 0.5, 1],
    'reg_lambda': [0.5, 1, 5],
    'base_score': [0.2, 0.5, 1]
}
```

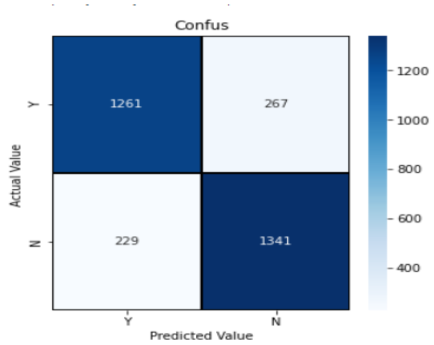
```
CV_XGB = GridSearchCV(XGBClassifier(n_jobs=-1), params, n_jobs=-1, cv=KFold(n_splits=3), scoring='roc_auc')
```

f the

```
58]:
pred = Customer_Churn.predict(x_test)
print("accuracy score: ",accuracy_score(y_test,pred))
print("confusion_matrix: \n",confusion_matrix(y_test,pred))
print("classification_report: \n",classification_report(y_test,pred))
```

```
accuracy score:  0.8398967075532602
confusion_matrix:
 [[1261  267]
 [ 229 1341]]
classification_report:
               precision    recall  f1-score   support

           0       0.85      0.83      0.84      1528
           1       0.83      0.85      0.84      1570

    accuracy                           0.84      3098
   macro avg       0.84      0.84      0.84      3098
weighted avg       0.84      0.84      0.84      3098
```

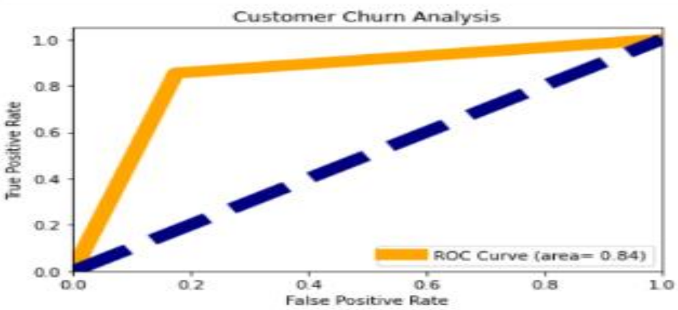
Confusion Matrix :



ROC-AUC Curve:

```
]:
fpr, tpr, threshold = roc_curve(y_test,pred)
auc = roc_auc_score(y_test,pred)
```

```
]:
plt.figure()
plt.plot(fpr,tpr,color="orange",lw=10,label="ROC Curve (area= %0.2f)" % auc)
plt.plot([0,1],[0,1],color="navy",lw=10,linestyle="--")
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Customer Churn Analysis")
plt.legend(loc="lower right")
plt.show()
```



Conclusion :

- This is the AUC-ROC curve for the models which is plotted False positive rate against True positive rate. So the best model has the area under curve as 0.84.

Saving the Model :

```
import pickle
filename='Customer_Churn_Analysis.pickle'
pickle.dump(CV_XGB,open(filename,'wb'))
loaded_model = pickle.load(open(filename, 'rb'))
loaded_model.predict(x_test)
```

```
array([1, 1, 1, ..., 0, 0, 1])
```

```
mport pickle.
file_name='my_file.pkl'
f = open(file_name,'wb')
pickle. dump(my_data,f)
f. close()
```

Checking predicted and original values

```
a =np.array(y_test)
predicted=np.array(loaded_model.predict(x_test))
Customer_Churn_Analysis=pd.DataFrame({'Orginal':a,'Predicted':predicted}, index=range(len(a)))
Customer_Churn_Analysis
```

	Orginal	Predicted
0	1	1
1	1	1
2	1	1
3	1	1
4	1	1
...	...	...
3093	0	0
3094	0	0
3095	1	0
3096	0	0
3097	1	1

3098 rows × 2 columns

Learning Outcomes of the Study in respect of Data Science :

- Encoding of categorical data is an important part of any problem.
- Scaling and standardization of data is mandatory.
- Feature selection played an important role in any ML problem. Unnecessary features and the features correlated with another needs to be removed.
- Most of the case accuracy score is improved after applying hyper parameter tuning.
- Data needs to be much precise and detailed for much better score.
- PCA used find patterns and extract the latent features from our dataset. It has an important role of building ML models.

Concluding Remarks on EDA and ML Model:

● For maximum case, Customer Churn is No.

● The maximum customer who have tendency to Churn are with No Online Security.

● If Monthly Charges is high, then the customers are more tendency to choose churn compare to rest.

● Different feature engineering techniques like balancing data, outliers' removal, label encoding, feature selection & PCA are perform on data. ● Random Forest is the best model for this particular dataset.