CUSTOMER CHURN ANALYSIS PROJECT

Submitted by:

TAMALI SAHA

Email: tamali428@gmail.com

Introduction:

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.

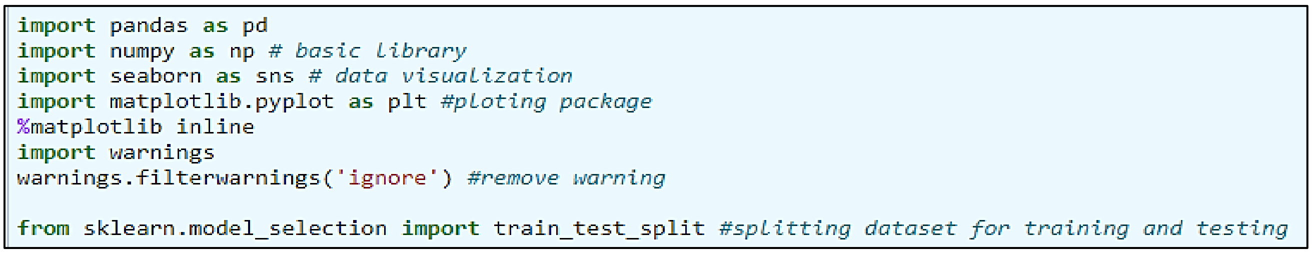
In order to develop and evaluate several customer churn prediction models, we will investigate customer data from IBM Sample Data Sets. You can also download dataset from the GitHub link here.

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.

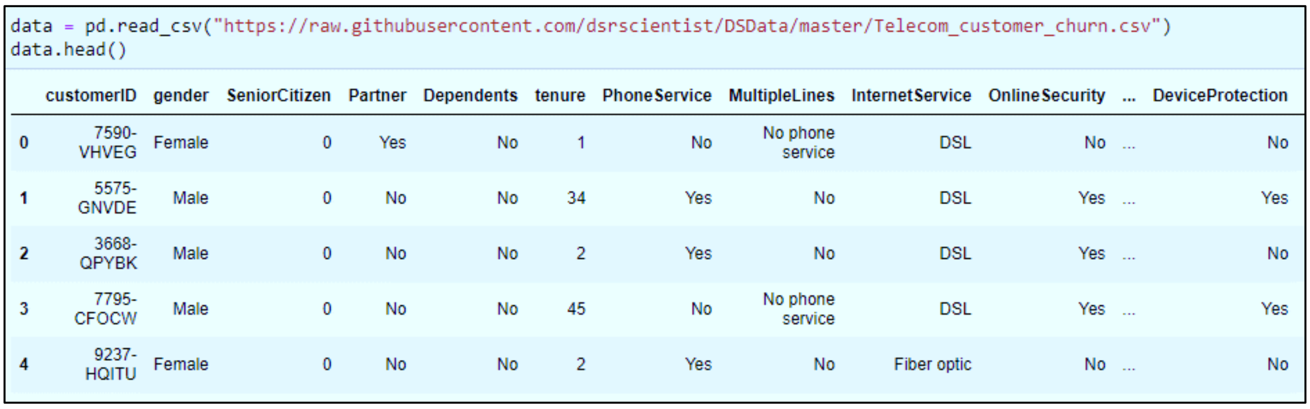
Data Analysis:

Data Preparation: Load, Clean and Format:



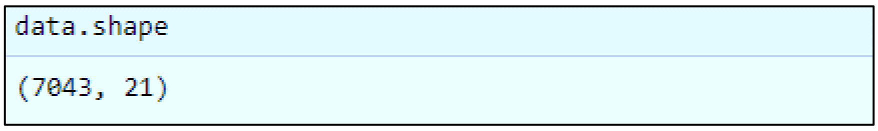
```
import pandas as pd
import numpy as np # basic library
import seaborn as sns # data visualization
import matplotlib.pyplot as plt #ploting package
%matplotlib inline
import warnings
warnings.filterwarnings('ignore') #remove warning

from sklearn.model_selection import train_test_split #splitting dataset for training and testing
```

Let's begin with importing libraries for EDA and dataset itself.

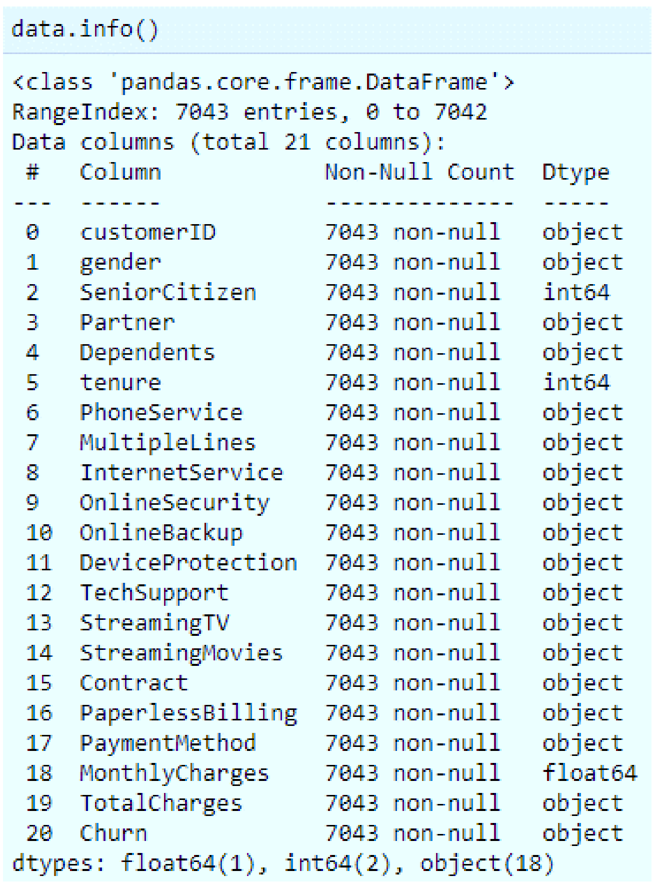


```
data = pd.read_csv("https://raw.githubusercontent.com/dsrscientist/DSData/master/Telecom_customer_churn.csv")
data.head()
```

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



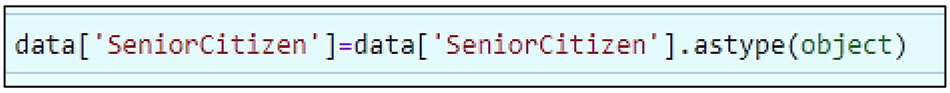
```
data.shape
(7043, 21)
```



```
data.info()
```

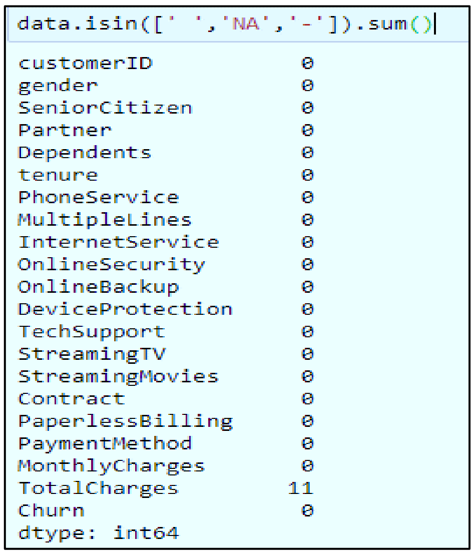
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   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)
```

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.



```
data['SeniorCitizen']=data['SeniorCitizen'].astype(object)
```

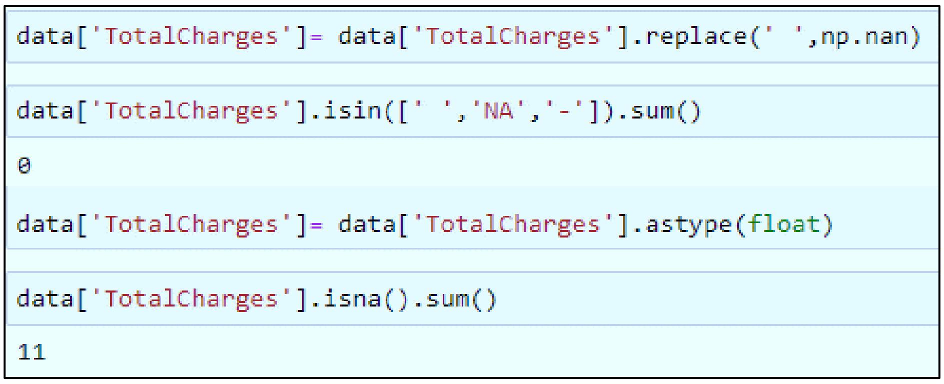
Data Integrity Check: Dataset can have missing values, duplicated entries and whitespaces. Now we will perform this integrity check of dataset.



```
data.isin([' ','NA','-']).sum()
```

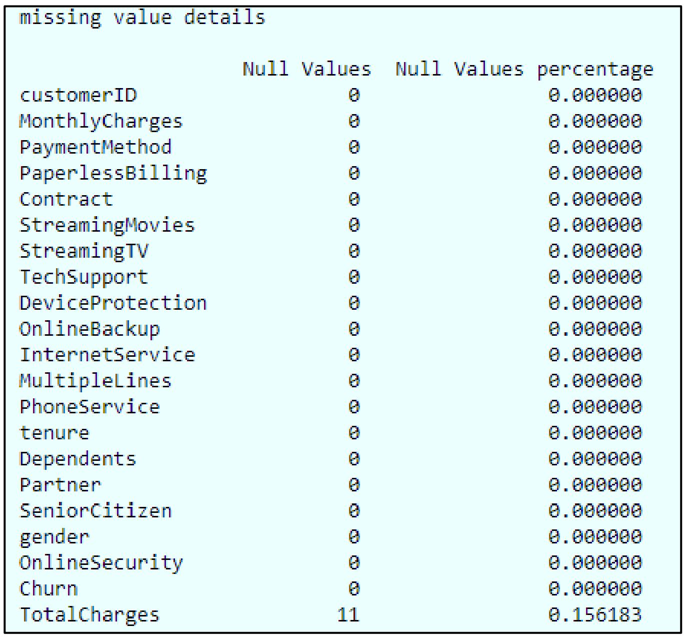
```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

Replace ' ' with np.NaN. No blank space, NA, '-', '?' exist in dataset.



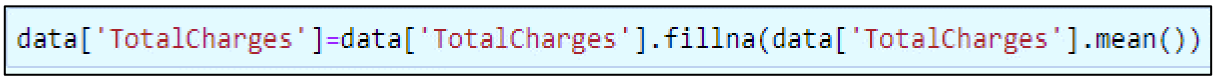
```
data['TotalCharges']= data['TotalCharges'].replace(' ',np.nan)

data['TotalCharges'].isin([' ','NA','-']).sum()

0

data['TotalCharges']= data['TotalCharges'].astype(float)

data['TotalCharges'].isna().sum()

11
```

Now check the missing values.



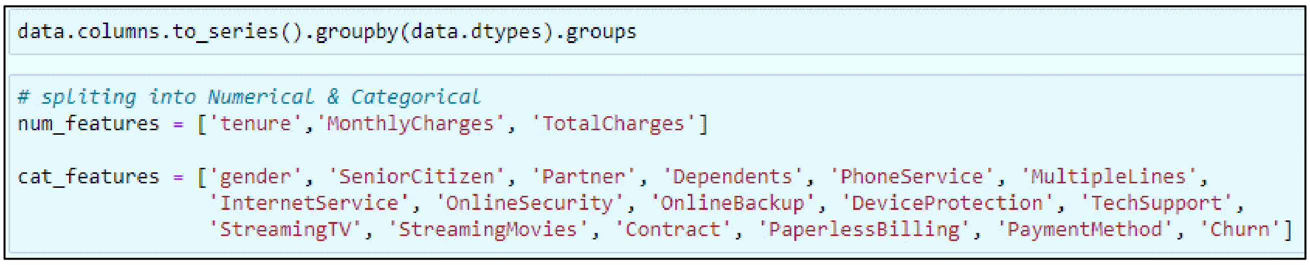
```
missing value details

                  Null Values  Null Values percentage
customerID                  0                0.000000
MonthlyCharges              0                0.000000
PaymentMethod               0                0.000000
PaperlessBilling            0                0.000000
Contract                    0                0.000000
StreamingMovies             0                0.000000
StreamingTV                 0                0.000000
TechSupport                 0                0.000000
DeviceProtection            0                0.000000
OnlineBackup                0                0.000000
InternetService             0                0.000000
MultipleLines               0                0.000000
PhoneService                0                0.000000
tenure                      0                0.000000
Dependents                  0                0.000000
Partner                     0                0.000000
SeniorCitizen               0                0.000000
gender                      0                0.000000
OnlineSecurity              0                0.000000
Churn                       0                0.000000
TotalCharges               11                0.156183
```

Here missing values are present. Let's impute this with mean as it is a continuous data.



```
data['TotalCharges']=data['TotalCharges'].fillna(data['TotalCharges'].mean())
```

Drop the unnecessary column and check different datatypes in dataset.



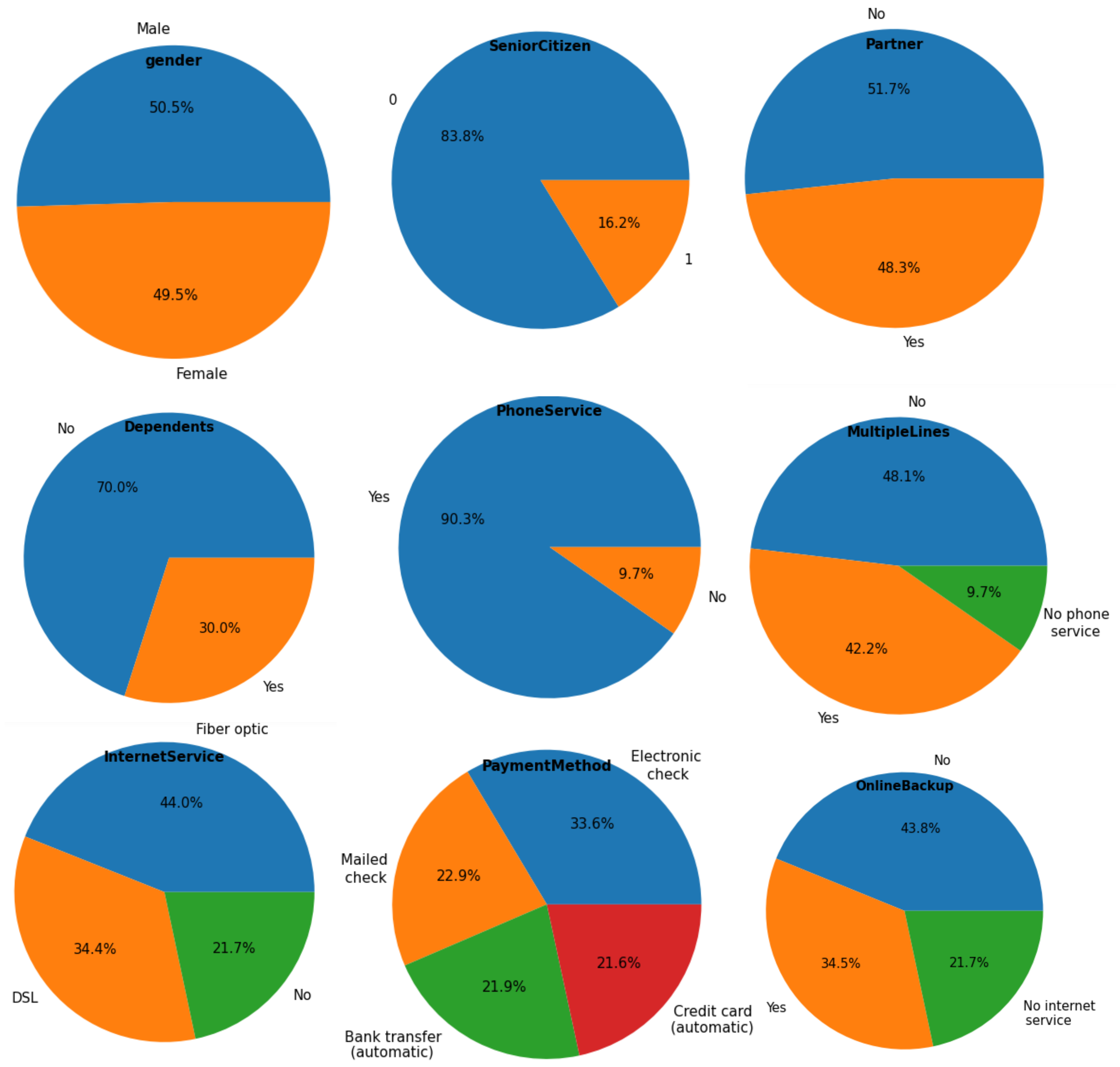
```
data.columns.to_series().groupby(data.dtypes).groups

# spliting into Numerical & Categorical
num_features = ['tenure','MonthlyCharges', 'TotalCharges']

cat_features = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
                'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
                'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'Churn']
```

Let's go for EDA.

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.*

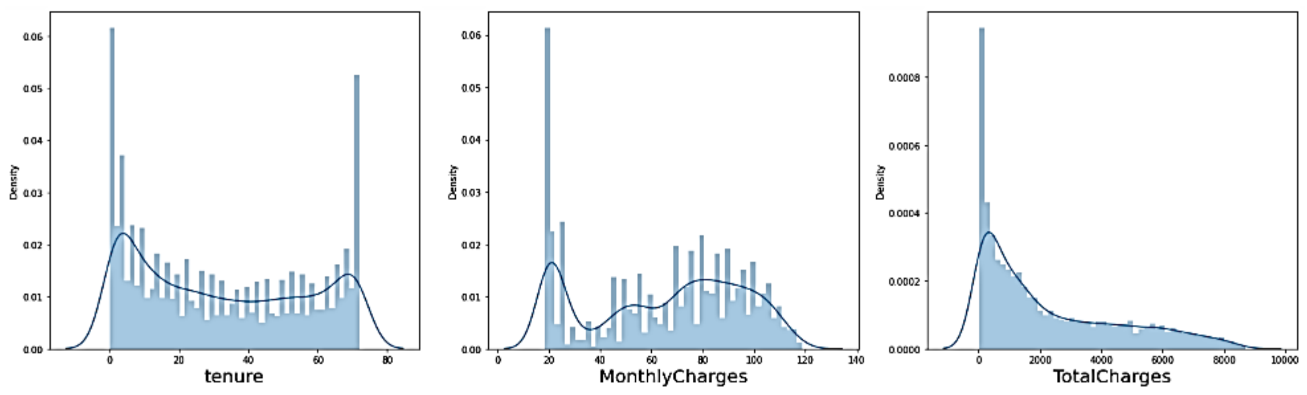


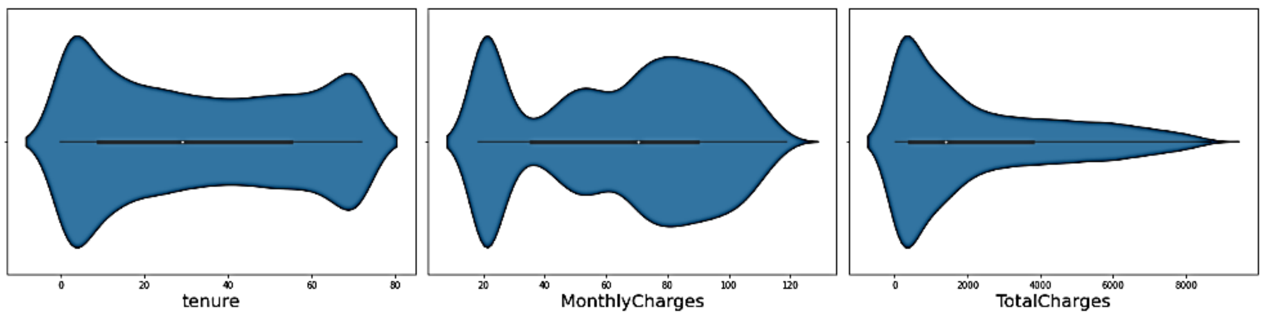
Observations from above plot:

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*:

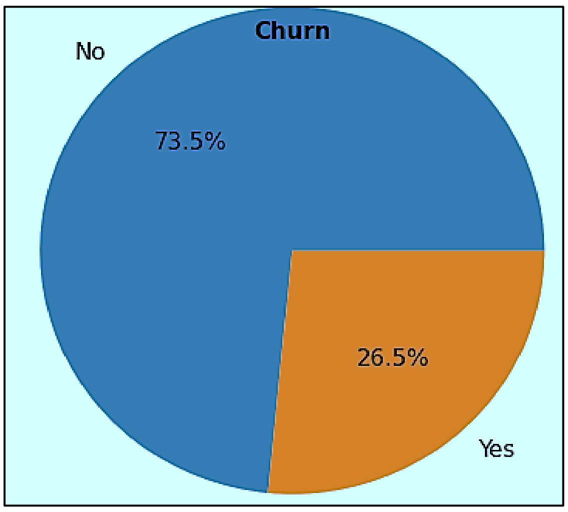


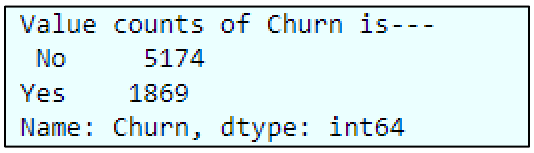


Observations from above plot:

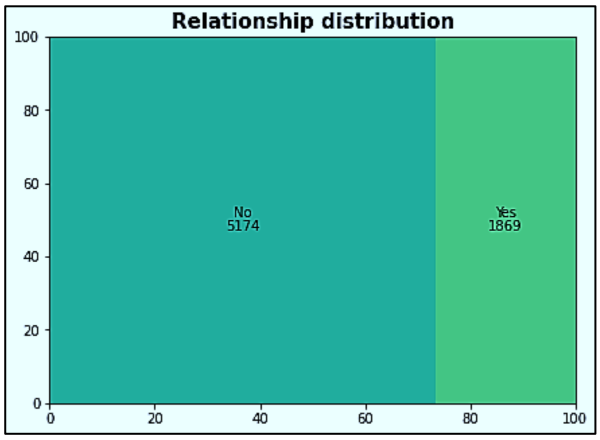
1. Average range of age is 0-70.
2. Monthly charges range is 20-120.
3. 0 value is present in TotalCharges column.
4. All the data have right skewness.

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





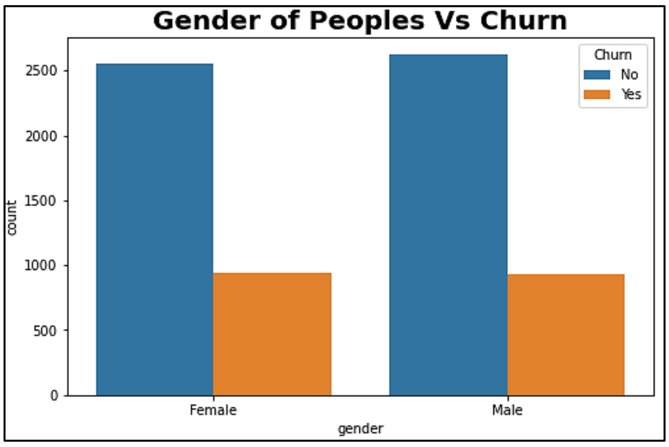
```
Value counts of Churn is---
 No     5174
Yes    1869
Name: Churn, dtype: int64
```

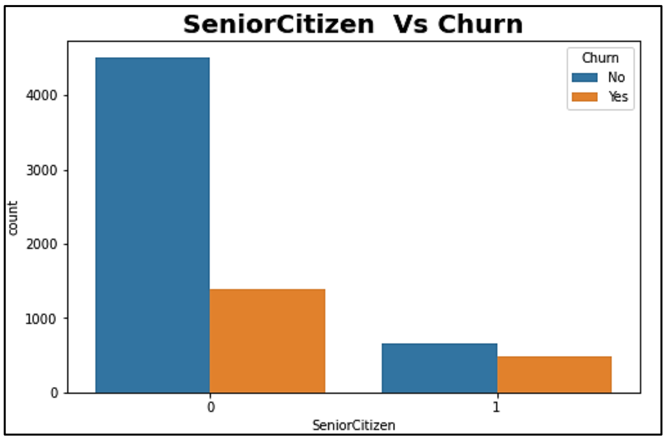


Observations from above plot:

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.

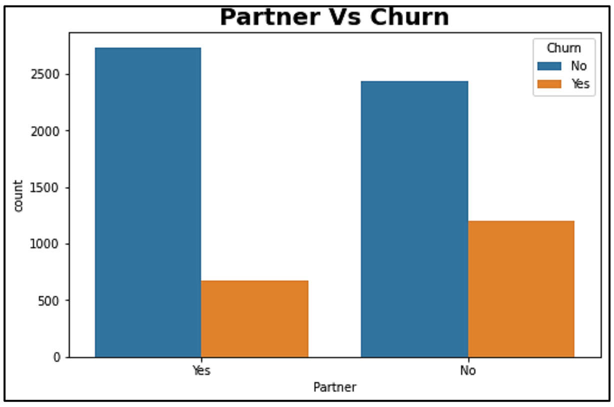
Let's analyse the impact of different ***features on target variable****.*

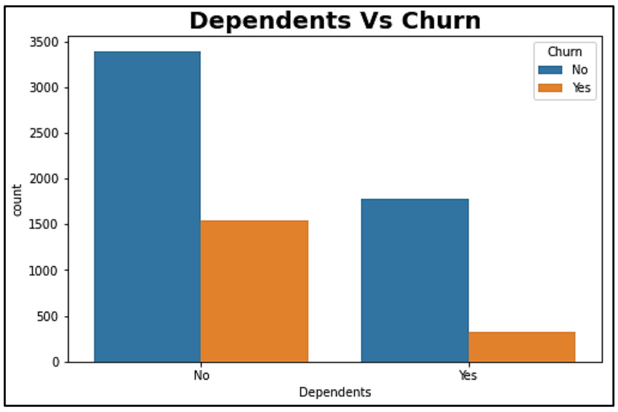




Observations from above plot:

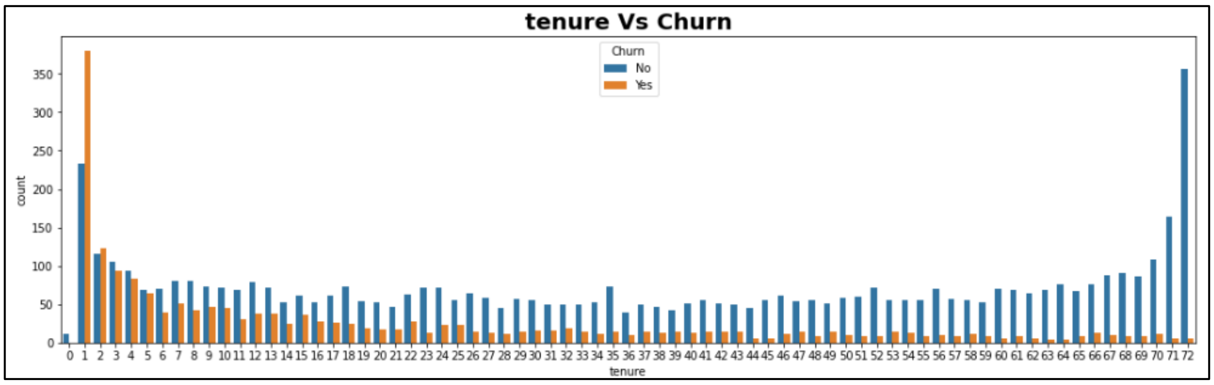
1. In terms of gender, the distribution of Churn is in same proportion with minor difference.
2. For Male, YES: NO= 26:74 and for Female, YES: NO= 27:73
3. Senior citizen have more tendency to churn with respect to others.

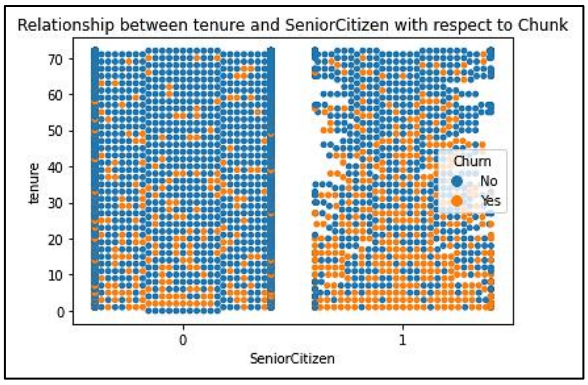




Observations from above plot:

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.

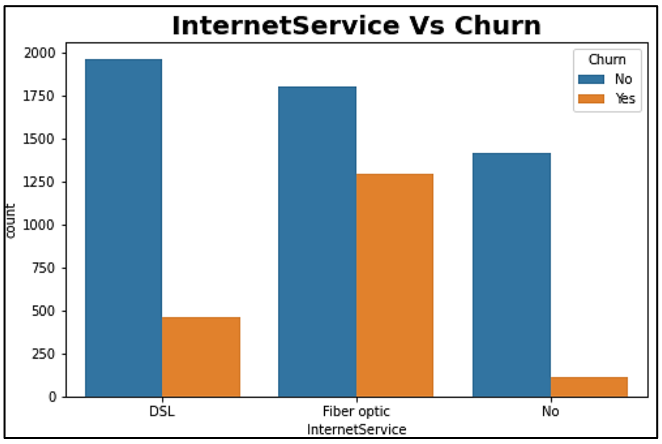


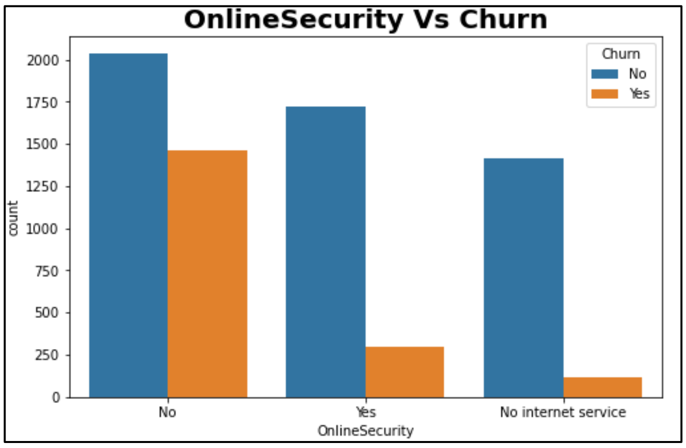


Observations from above plot:

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.



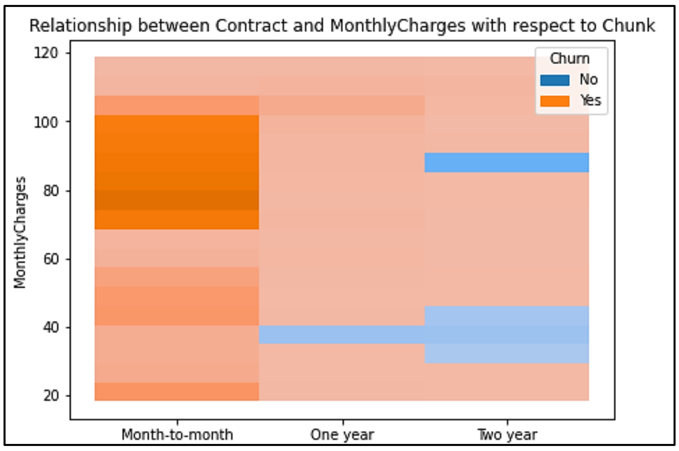


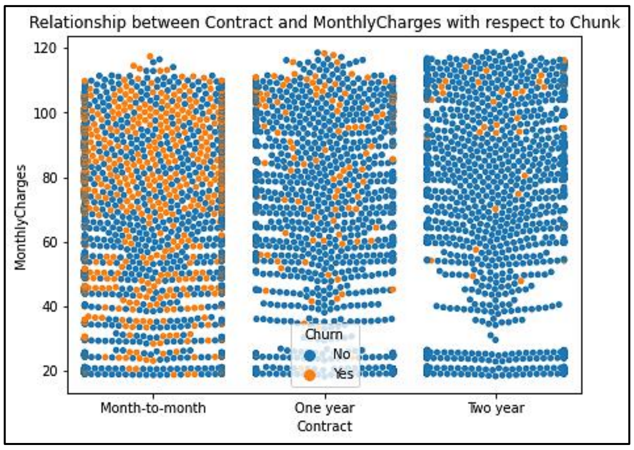
Observations from above plot:

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.

Plotting count plot of DeviceProtection, TechSupport, StreamingTV, StreamingMovies with ***Chunk***, we have the following observations:

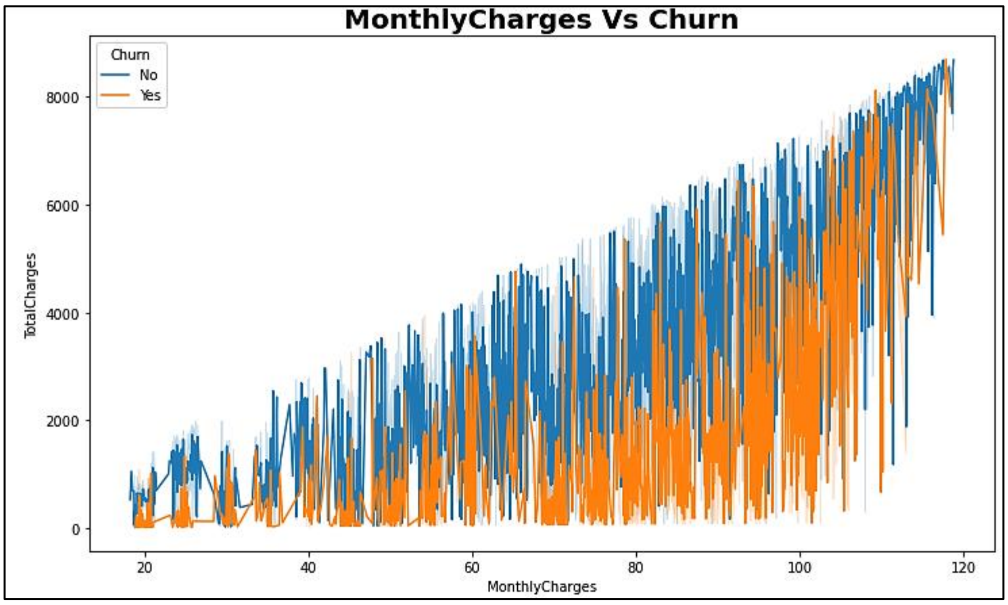
1. The tendency of chuck increased if the customer have no Online Backup. It is quite obvious!!
2. The customer with no device protection is more tendency to Chunk.
3. The customer with no tech support (just like device protection) is more tendency to Chunk.
4. Churn tendency in people who streamingTV or not are same.





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

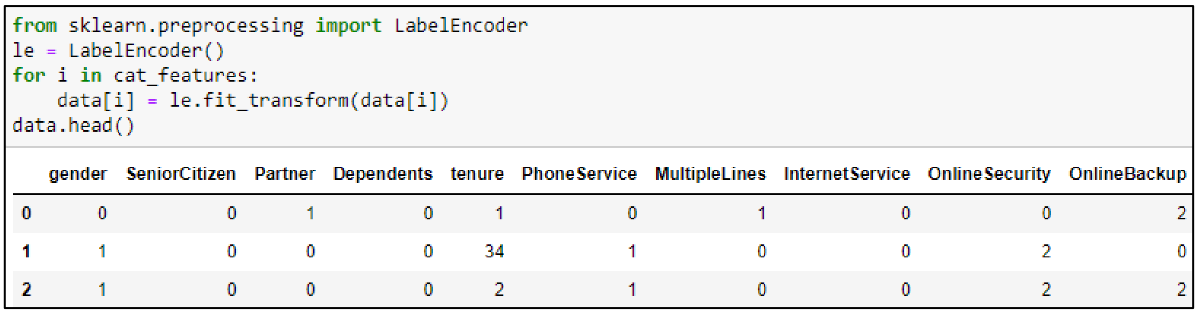
Depending on the needs of the dataset, a variety of strategies are used to obtain the above results. Effective methods include the following:

- 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.

1. Encoding categorical data using label encoding:

Label Encoding is employed over target variable ‘Attrition’ while Ordinal encoding employ for rest categorical features.

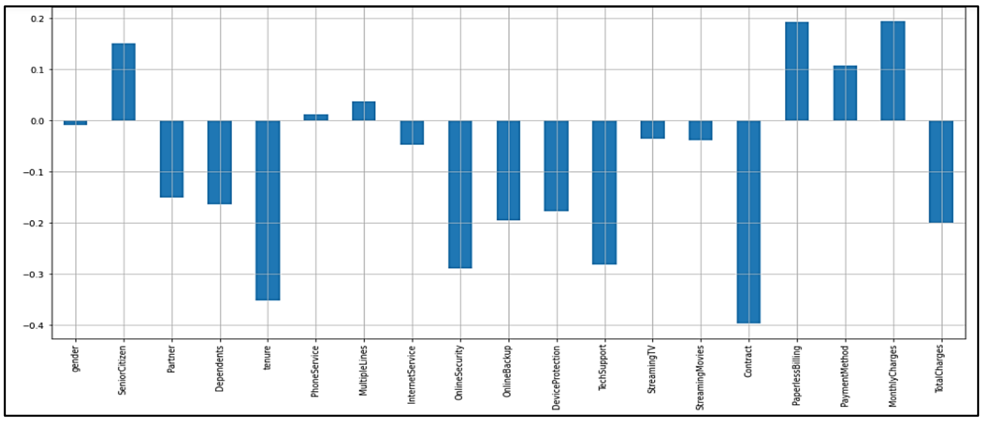


```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in cat_features:
    data[i] = le.fit_transform(data[i])
data.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
0	0	0	1	0	1	0	1	0	0	2
1	1	0	0	0	34	1	0	0	2	0
2	1	0	0	0	2	1	0	0	2	2

2. Correlation:

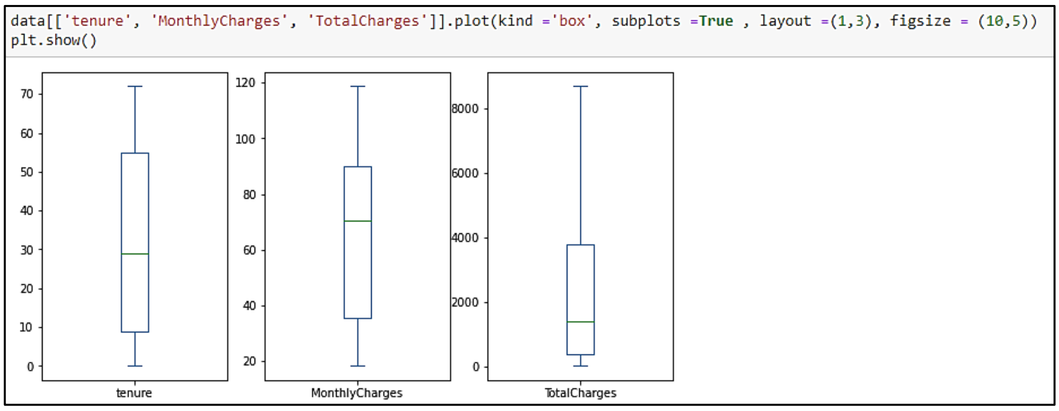
The Correlation Heat map informs us of potential multicollinearity issues by quickly displaying which variables are connected, to what extent, and in which direction. Below is a bar plot showing the target variable's correlation coefficient with independent features.



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.

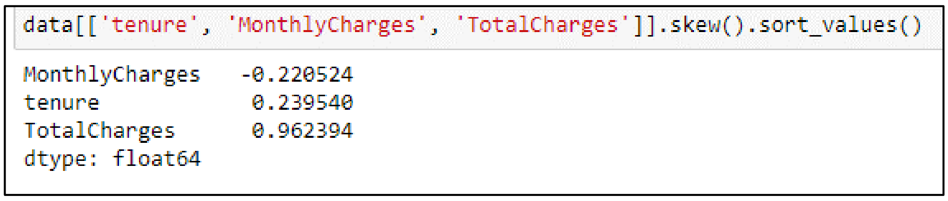
3. Outlier Detection of data and removal:



```
data[['tenure', 'MonthlyCharges', 'TotalCharges']].plot(kind ='box', subplots =True , layout =(1,3), figsize = (10,5))
plt.show()
```

Fortunately no such outliers are present in the dataset.

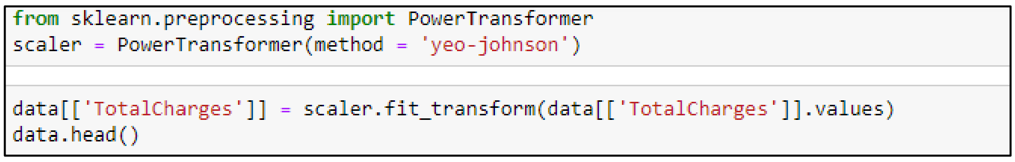
4. Skewness:



```
data[['tenure', 'MonthlyCharges', 'TotalCharges']].skew().sort_values()
```

```
MonthlyCharges   -0.220524
tenure            0.239540
TotalCharges      0.962394
dtype: float64
```

Here numerical variable is 'tenure', 'MonthlyCharges', 'TotalCharges' and out of which TotalCharges have some skewness. Let's remove the skewness.

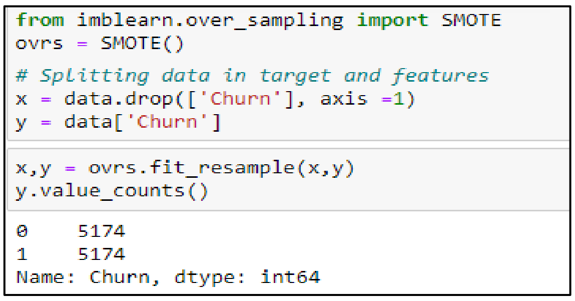


```
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method = 'yeo-johnson')

data[['TotalCharges']] = scaler.fit_transform(data[['TotalCharges']].values)
data.head()
```

Skewness is removed.

5. Balancing Imbalanced target feature(Churn) using SMOTE:

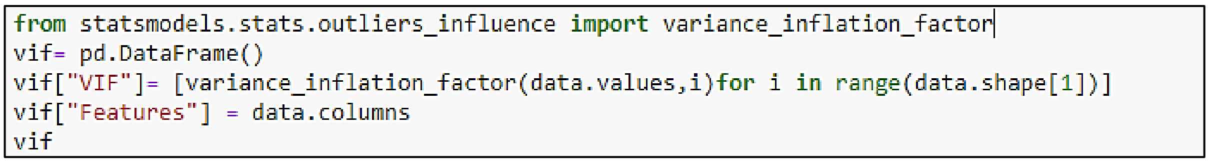
It is unbalanced in this two-class sample (75:25). Consequently, there is a chance that the model created may be biased toward the dominant and overrepresented class. By using the Synthetic Minority Oversampling Technique (SMOTE), we can oversample the minority class and overcome this problem.



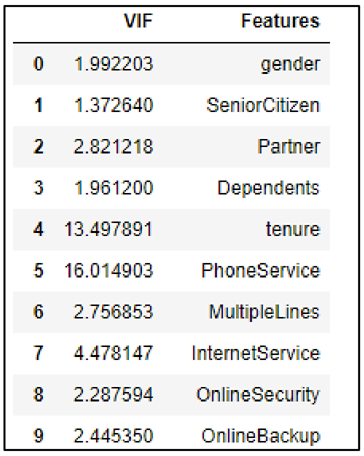
```
from imblearn.over_sampling import SMOTE
ovrs = SMOTE()
# Splitting data in target and features
x = data.drop(['Churn'], axis =1)
y = data['Churn']

x,y = ovrs.fit_resample(x,y)
y.value_counts()

0    5174
1    5174
Name: Churn, dtype: int64
```

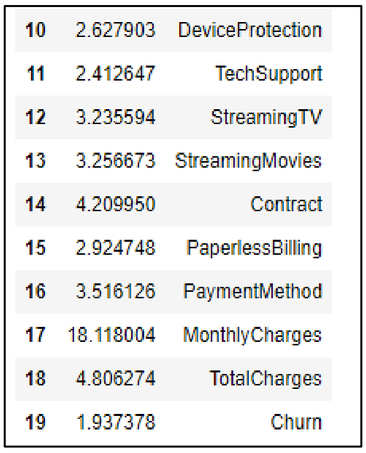
6. Checking Multicollinearity by VIF:

VIF imported from statsmodels.stats.outliers_influence to check multicollinearity between features. Multicollinearity present between different features.



```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif= pd.DataFrame()
vif["VIF"]= [variance_inflation_factor(data.values,i)for i in range(data.shape[1])]
vif["Features"] = data.columns
vif
```

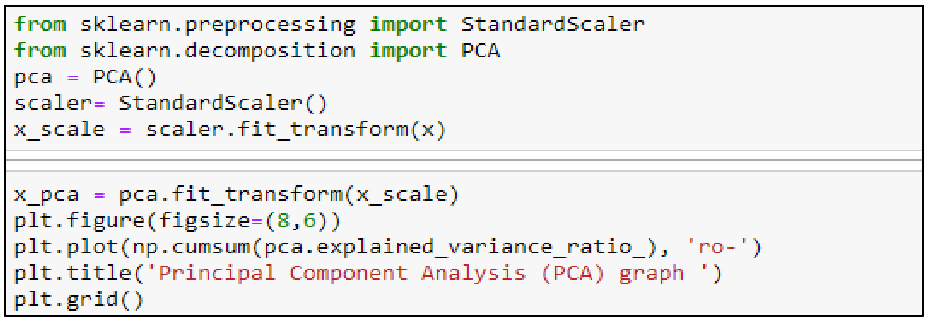




	VIF	Features
0	1.992203	gender
1	1.372640	SeniorCitizen
2	2.821218	Partner
3	1.961200	Dependents
4	13.497891	tenure
5	16.014903	PhoneService
6	2.756853	MultipleLines
7	4.478147	InternetService
8	2.287594	OnlineSecurity
9	2.445350	OnlineBackup
10	2.627903	DeviceProtection
11	2.412647	TechSupport
12	3.235594	StreamingTV
13	3.256673	StreamingMovies
14	4.209950	Contract
15	2.924748	PaperlessBilling
16	3.516126	PaymentMethod
17	18.118004	MonthlyCharges
18	4.806274	TotalCharges
19	1.937378	Churn

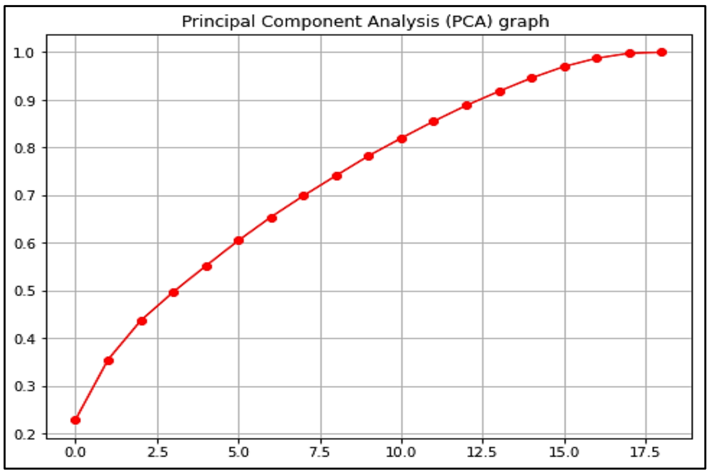
7. Principal Component Analysis (PCA):

PCA used find patterns and extract the latent features from our dataset.

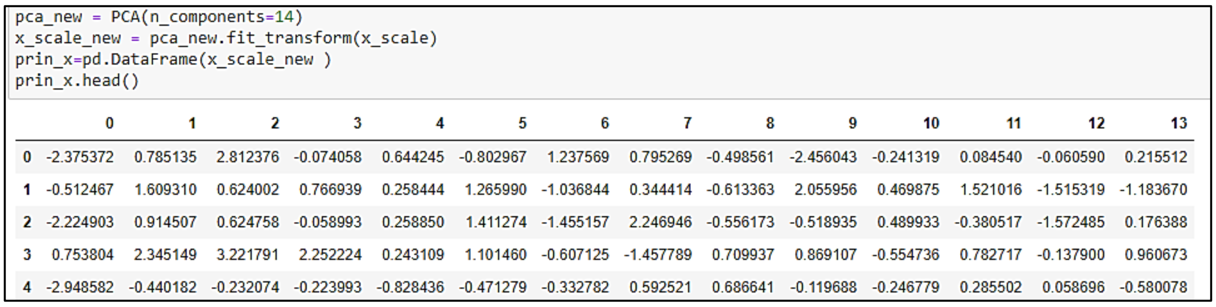


```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
pca = PCA()
scaler= StandardScaler()
x_scale = scaler.fit_transform(x)
```

```
x_pca = pca.fit_transform(x_scale)
plt.figure(figsize=(8,6))
plt.plot(np.cumsum(pca.explained_variance_ratio_), 'ro-')
plt.title('Principal Component Analysis (PCA) graph ')
plt.grid()
```



Here, around 95% variance gives the first 14 component. Let's take the first 14 components.

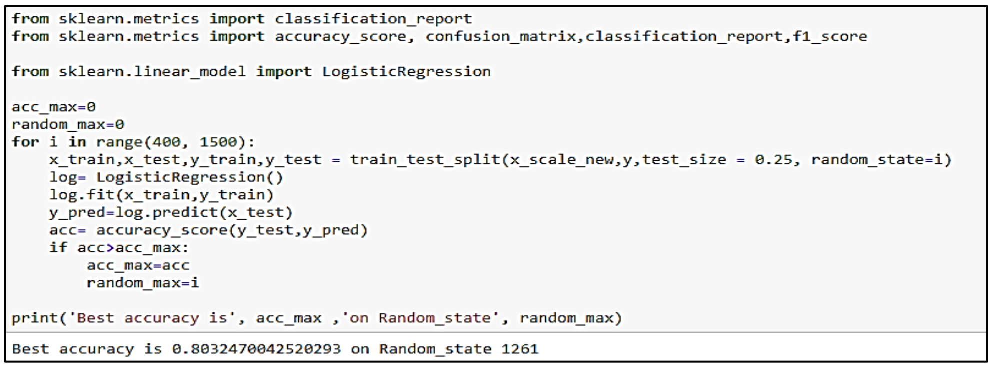


```
pca_new = PCA(n_components=14)
x_scale_new = pca_new.fit_transform(x_scale)
prin_x=pd.DataFrame(x_scale_new )
prin_x.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	-2.375372	0.785135	2.812376	-0.074058	0.644245	-0.802967	1.237569	0.795269	-0.498561	-2.456043	-0.241319	0.084540	-0.060590	0.215512
1	-0.512467	1.609310	0.624002	0.766939	0.258444	1.265990	-1.036844	0.344414	-0.613363	2.055956	0.469875	1.521016	-1.515319	-1.183670
2	-2.224903	0.914507	0.624758	-0.058993	0.258850	1.411274	-1.455157	2.246946	-0.556173	-0.518935	0.489933	-0.380517	-1.572485	0.176388
3	0.753804	2.345149	3.221791	2.252224	0.243109	1.101460	-0.607125	-1.457789	0.709937	0.869107	-0.554736	0.782717	-0.137900	0.960673
4	-2.948582	-0.440182	-0.232074	-0.223993	-0.828436	-0.471279	-0.332782	0.592521	0.686641	-0.119688	-0.246779	0.285502	0.058696	-0.580078

<u>Machine Learning Model Building:</u>

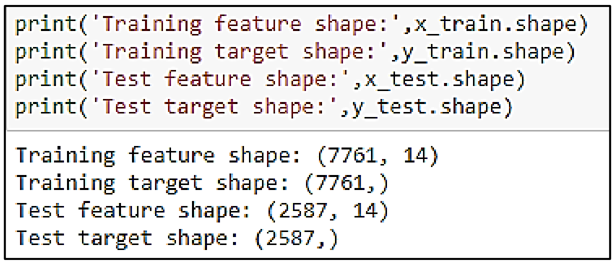
In this section we will build Supervised learning ML model-based classification algorithm. train_test_split used to split data with size of 0.25. Let’s find best Random state.



```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report,f1_score

from sklearn.linear_model import LogisticRegression

acc_max=0
random_max=0
for i in range(400, 1500):
    x_train,x_test,y_train,y_test = train_test_split(x_scale_new,y,test_size = 0.25, random_state=i)
    log= LogisticRegression()
    log.fit(x_train,y_train)
    y_pred=log.predict(x_test)
    acc= accuracy_score(y_test,y_pred)
    if acc>acc_max:
        acc_max=acc
        random_max=i

print('Best accuracy is', acc_max ,'on Random_state', random_max)
```

```
Best accuracy is 0.8032470042520293 on Random_state 1261
```

Here the best accuracy on Random_state=1261, let's take it.

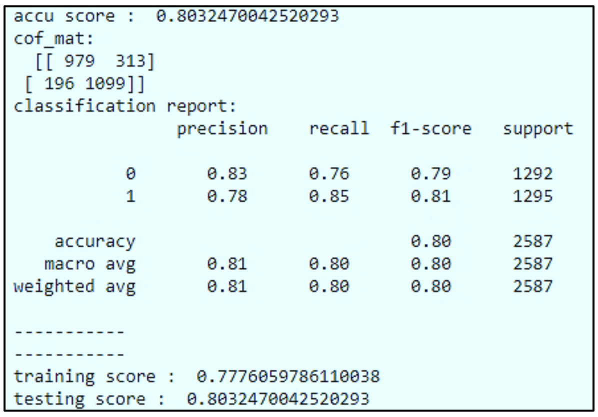


```
print('Training feature shape:',x_train.shape)
print('Training target shape:',y_train.shape)
print('Test feature shape:',x_test.shape)
print('Test target shape:',y_test.shape)
```

```
Training feature shape: (7761, 14)
Training target shape: (7761,)
Test feature shape: (2587, 14)
Test target shape: (2587,)
```

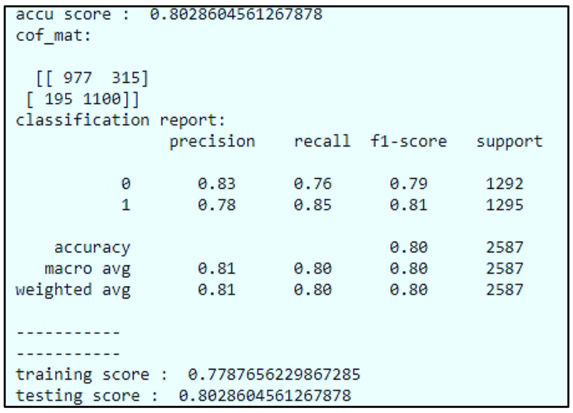
1. ***Logistic Regression*** **:**

The evaluation matrix along with classification report before and using hyper parameter tuning the evaluation matrix along with classification report for Logistic Regression is as follow:



```
accu score :  0.8032470042520293
cof_mat:
  [[ 979  313]
 [ 196 1099]]
classification report:
              precision    recall  f1-score   support

           0       0.83      0.76      0.79      1292
           1       0.78      0.85      0.81      1295

    accuracy                           0.80      2587
   macro avg       0.81      0.80      0.80      2587
weighted avg       0.81      0.80      0.80      2587

-----------
-----------
training score :  0.7776059786110038
testing score :  0.8032470042520293
```

After
Hyper
Parameter
Tuning

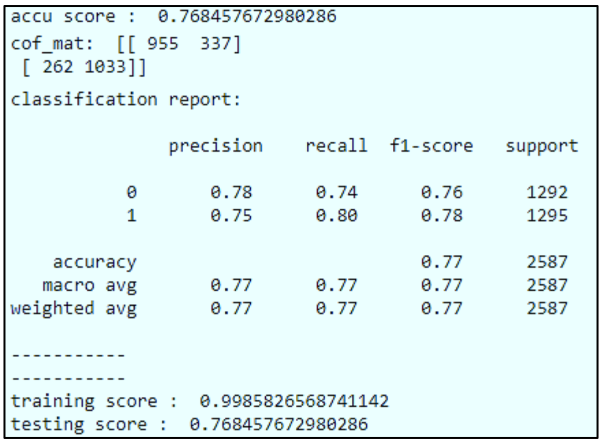


```
accu score :  0.8028604561267878
cof_mat:

 [[ 977  315]
 [ 195 1100]]
classification report:
              precision    recall  f1-score   support

           0       0.83      0.76      0.79      1292
           1       0.78      0.85      0.81      1295

    accuracy                           0.80      2587
   macro avg       0.81      0.80      0.80      2587
weighted avg       0.81      0.80      0.80      2587

-----------
-----------
training score :  0.7787656229867285
testing score :  0.8028604561267878
```

R2 score not improved after using gridsearchCV .

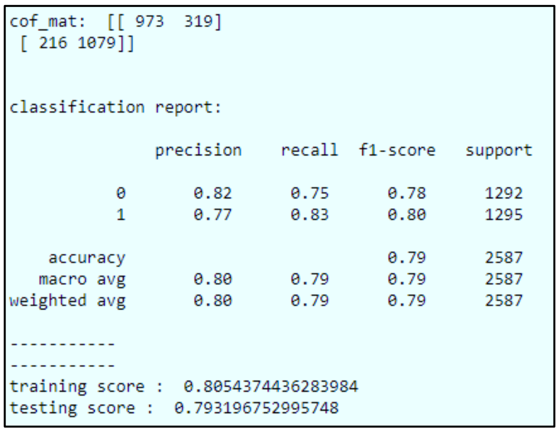
2. ***Decision Tree Classifier*** **:**

The evaluation matrix along with classification report before and using hyper parameter tuning the evaluation matrix along with classification report for Decision Tree Classifier is as follow:



```
accu score :  0.768457672980286
cof_mat:  [[ 955  337]
 [ 262 1033]]
classification report:
              precision    recall  f1-score   support

           0       0.78      0.74      0.76      1292
           1       0.75      0.80      0.78      1295

    accuracy                           0.77      2587
   macro avg       0.77      0.77      0.77      2587
weighted avg       0.77      0.77      0.77      2587

-----------
-----------
training score :  0.9985826568741142
testing score :  0.768457672980286
```

After
Hyper
Parameter
Tuning

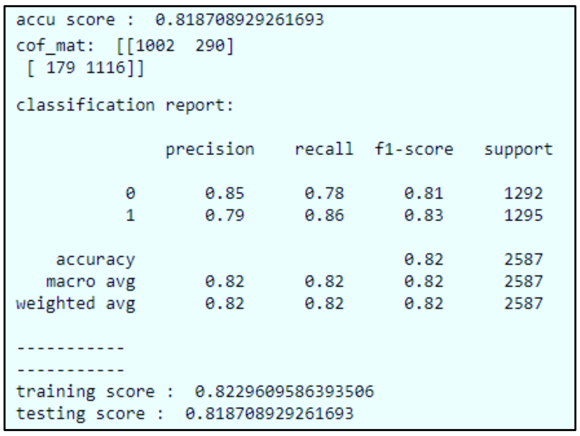


```
cof_mat:  [[ 973  319]
 [ 216 1079]]

classification report:
              precision    recall  f1-score   support

           0       0.82      0.75      0.78      1292
           1       0.77      0.83      0.80      1295

    accuracy                           0.79      2587
   macro avg       0.80      0.79      0.79      2587
weighted avg       0.80      0.79      0.79      2587

-----------
-----------
training score :  0.8054374436283984
testing score :  0.793196752995748
```

Accuracy score is slightly improved after using GridSearchCV with DecisionTreeClassifier(). The difference between training score, testing score is also decreased.

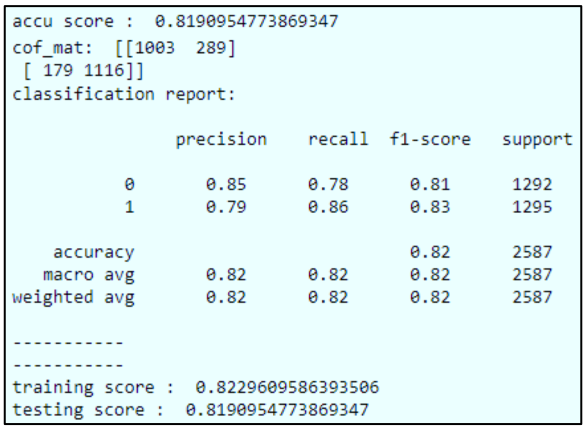
3. ***Gradient Boosting Classifier:***

The evaluation matrix along with classification report before and using hyper parameter tuning the evaluation matrix along with classification report for Gradient Boosting Classifier is as follow:



```
accu score :  0.818708929261693
cof_mat:  [[1002  290]
 [ 179 1116]]
classification report:

              precision    recall  f1-score   support

           0       0.85      0.78      0.81      1292
           1       0.79      0.86      0.83      1295

    accuracy                           0.82      2587
   macro avg       0.82      0.82      0.82      2587
weighted avg       0.82      0.82      0.82      2587

-----------
-----------
training score :  0.8229609586393506
testing score :  0.818708929261693
```

After Hyper Parameter Tuning

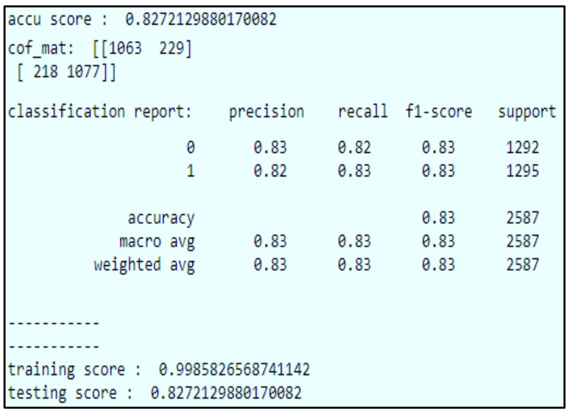


```
accu score :  0.8190954773869347
cof_mat:  [[1003  289]
 [ 179 1116]]
classification report:
              precision    recall  f1-score   support

           0       0.85      0.78      0.81      1292
           1       0.79      0.86      0.83      1295

    accuracy                           0.82      2587
   macro avg       0.82      0.82      0.82      2587
weighted avg       0.82      0.82      0.82      2587

-----------
-----------
training score :  0.8229609586393506
testing score :  0.8190954773869347
```

Accuracy score, training score, testing score are not improved after using GridSearchCV with GradientBoostingClassifier().

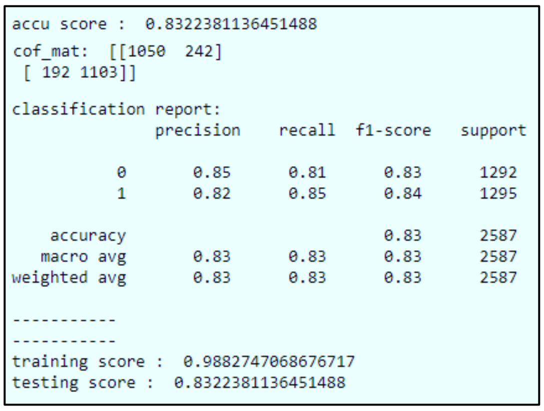
4. ***Random Forest Classifier:***

The evaluation matrix along with classification report before and using hyper parameter tuning the evaluation matrix along with classification report for Random Forest Classifier is as follow:



```
accu score :  0.8272129880170082
cof_mat:  [[1063  229]
 [ 218 1077]]
classification report:              precision    recall  f1-score   support
           0       0.83      0.82      0.83      1292
           1       0.82      0.83      0.83      1295

    accuracy                           0.83      2587
   macro avg       0.83      0.83      0.83      2587
weighted avg       0.83      0.83      0.83      2587

-----------
-----------
training score :  0.9985826568741142
testing score :  0.8272129880170082
```

After Hyper Parameter Tuning

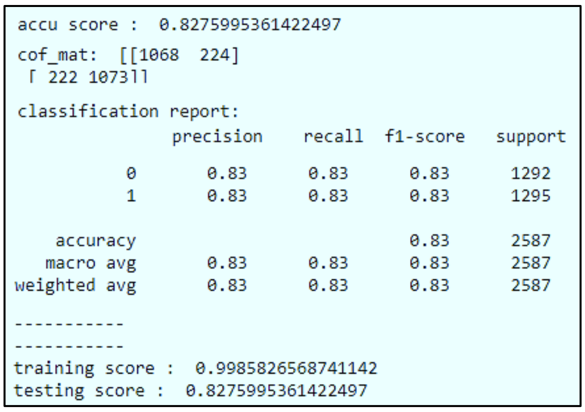


```
accu score :  0.8322381136451488
cof_mat:  [[1050  242]
 [ 192 1103]]
classification report:
              precision    recall  f1-score   support

           0       0.85      0.81      0.83      1292
           1       0.82      0.85      0.84      1295

    accuracy                           0.83      2587
   macro avg       0.83      0.83      0.83      2587
weighted avg       0.83      0.83      0.83      2587

-----------
-----------
training score :  0.9882747068676717
testing score :  0.8322381136451488
```

Accuracy score is slightly improved after using GridSearchCV with RandomForestClassifier().

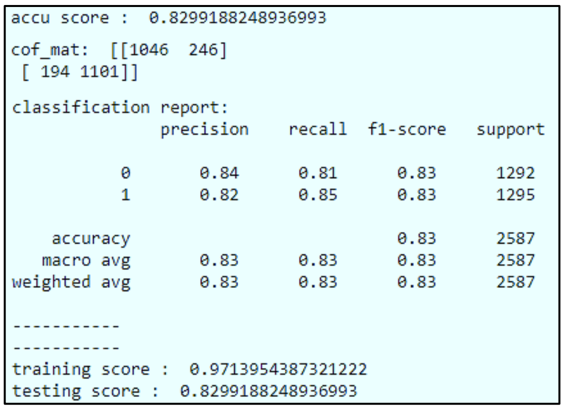
5. ***Extra Trees Classifier:***

The evaluation matrix along with classification report before and using hyper parameter tuning the evaluation matrix along with classification report Extra Trees Classifier is as follow:



```
accu score :  0.8275995361422497
cof_mat:  [[1068  224]
 [ 222 1073]]
classification report:
              precision    recall  f1-score   support

           0       0.83      0.83      0.83      1292
           1       0.83      0.83      0.83      1295

    accuracy                           0.83      2587
   macro avg       0.83      0.83      0.83      2587
weighted avg       0.83      0.83      0.83      2587

-----------
-----------
training score :  0.9985826568741142
testing score :  0.8275995361422497
```

After Hyper Parameter Tuning

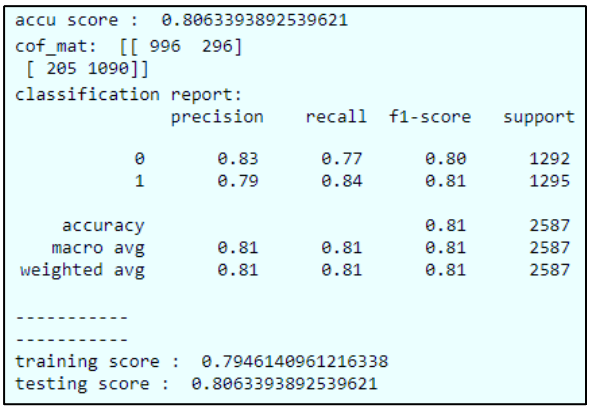


```
accu score :  0.8299188248936993
cof_mat:  [[1046  246]
 [ 194 1101]]
classification report:
              precision    recall  f1-score   support

           0       0.84      0.81      0.83      1292
           1       0.82      0.85      0.83      1295

    accuracy                           0.83      2587
   macro avg       0.83      0.83      0.83      2587
weighted avg       0.83      0.83      0.83      2587

-----------
-----------
training score :  0.9713954387321222
testing score :  0.8299188248936993
```

Accuracy score is slightly improved after using GridSearchCV with ExtraTreesClassifier()

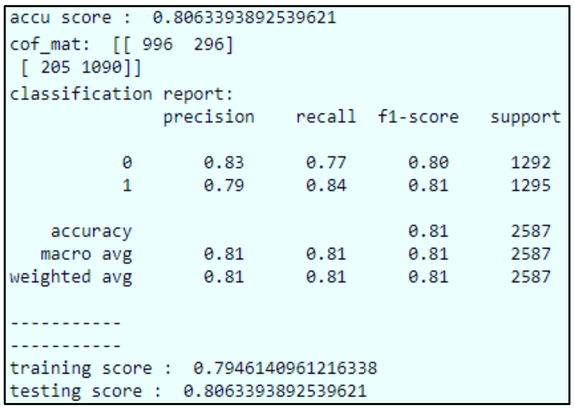
6. ***Ada Boost Classifier:***

The evaluation matrix along with classification report before and using hyper parameter tuning the evaluation matrix along with classification report Ada Boost Classifier is as follow:



```
accu score :  0.8063393892539621
cof_mat:  [[ 996  296]
 [ 205 1090]]
classification report:
              precision    recall  f1-score   support

           0       0.83      0.77      0.80      1292
           1       0.79      0.84      0.81      1295

    accuracy                           0.81      2587
   macro avg       0.81      0.81      0.81      2587
weighted avg       0.81      0.81      0.81      2587

-----------
-----------
training score :  0.7946140961216338
testing score :  0.8063393892539621
```

After Hyper Parameter Tuning

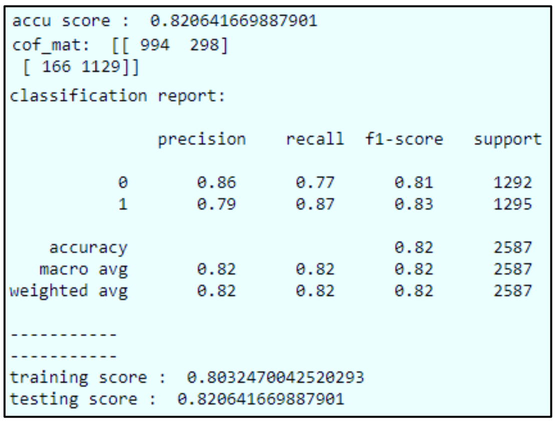


```
accu score :  0.8063393892539621
cof_mat:  [[ 996  296]
 [ 205 1090]]
classification report:
              precision    recall  f1-score   support

           0       0.83      0.77      0.80      1292
           1       0.79      0.84      0.81      1295

    accuracy                           0.81      2587
   macro avg       0.81      0.81      0.81      2587
weighted avg       0.81      0.81      0.81      2587

-----------
-----------
training score :  0.7946140961216338
testing score :  0.8063393892539621
```

Accuracy score is not improved after using GridSearchCV with AdaBoostClassifier().

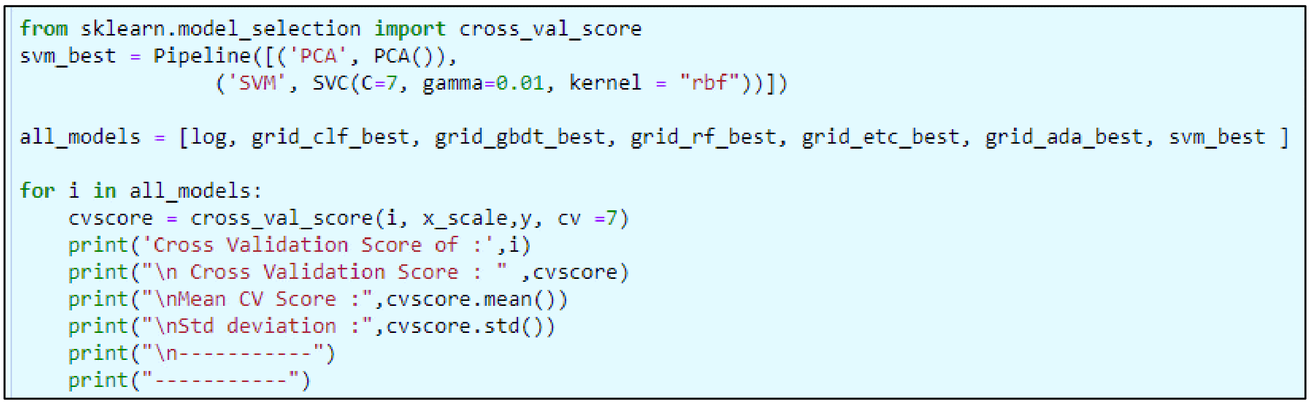
7. ***SVC:***

The evaluation matrix along with classification report is best for SVC using ‘rbf’ is as follow:



```
accu score :  0.820641669887901
cof_mat:  [[ 994  298]
 [ 166 1129]]
classification report:

              precision    recall  f1-score   support

           0       0.86      0.77      0.81      1292
           1       0.79      0.87      0.83      1295

    accuracy                           0.82      2587
   macro avg       0.82      0.82      0.82      2587
weighted avg       0.82      0.82      0.82      2587

-----------
-----------
training score :  0.8032470042520293
testing score :  0.820641669887901
```

Cross Validation:



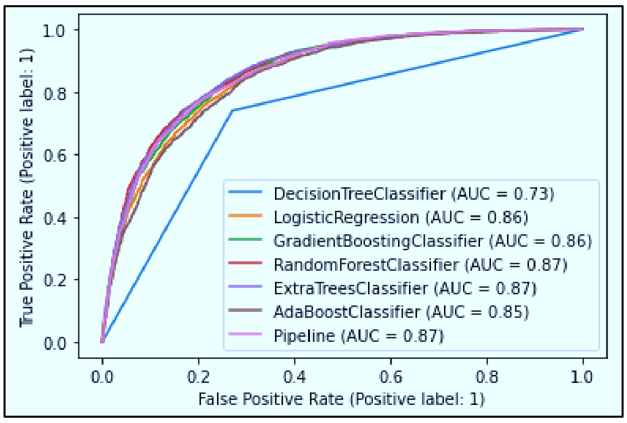
```
from sklearn.model_selection import cross_val_score
svm_best = Pipeline([('PCA', PCA()),
                ('SVM', SVC(C=7, gamma=0.01, kernel = "rbf"))])

all_models = [log, grid_clf_best, grid_gbdt_best, grid_rf_best, grid_etc_best, grid_ada_best, svm_best ]

for i in all_models:
    cvscore = cross_val_score(i, x_scale,y, cv =7)
    print('Cross Validation Score of :',i)
    print("\n Cross Validation Score : " ,cvscore)
    print("\nMean CV Score :",cvscore.mean())
    print("\nStd deviation :",cvscore.std())
    print("\n-----------")
    print("-----------")
```

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.

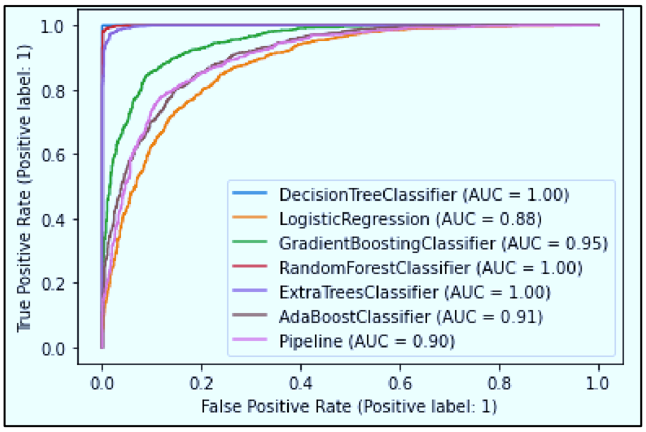
AUC-ROC Curve:

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.

For training data



For testing data



Here Randomforest gives best AUC score. So it is the final model for this dataset.

Now in a table let's view all 7 algorithm with ***Accuracy Score, CV Mean Score, f-1 Score, Recall and Precision.***

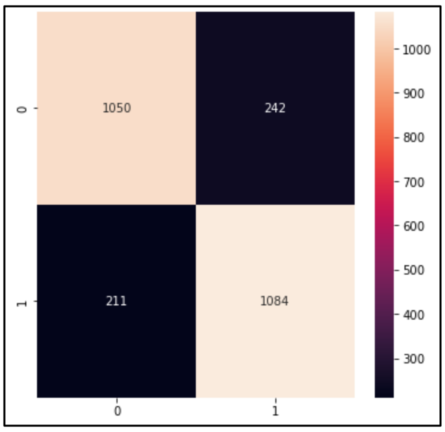
Tables of Findings using different algorithms after Hyper Parameter Tuning

Algorithm	*Accuracy Score*	*CV Mean Score*	*f-1 Score*	*Recall*	*Precision*
Logistic Regression	0.8032	0.7900	0.80	0.80	0.81
Decision Tree Classifier	0.7931	0.7820	0.79	0.79	0.80
Gradient Boosting Classifier	0.8190	0.8133	0.82	0.82	0.82
Random Forest Classifier	0.8272	0.8439	0.83	0.83	0.83
Extra Trees Classifier	0.8299	0.8356	0.83	0.83	0.83
Ada Boost Classifier	0.8063	0.8029	0.81	0.81	0.81
SVC ('rbf')	0.8206	0.8091	0.82	0.82	0.82

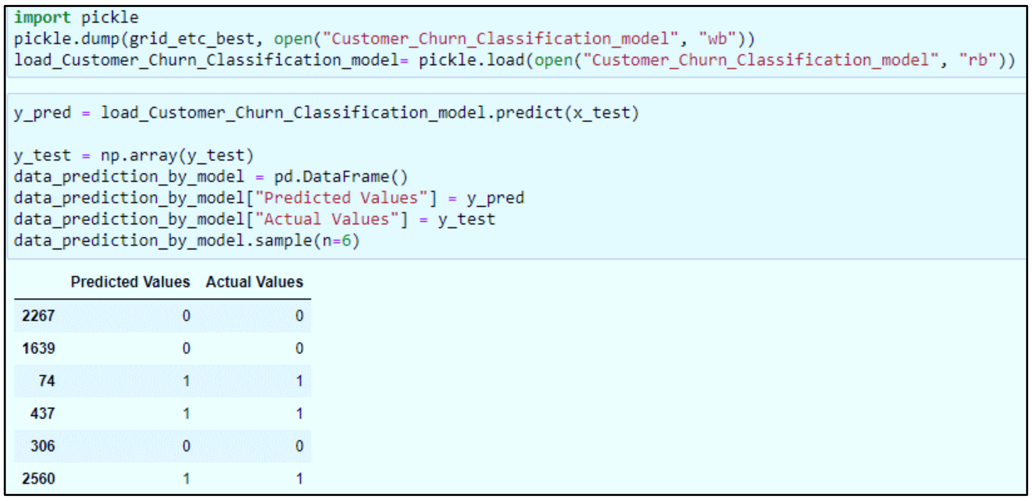
(Min Value in column -Green, Max Value in column - Yellow Colour)

Final Confusion Matrix:

The final model is Random forest for which accuracy score is 0.8248936992655586. The confusion matrix of the problem is as followes:



At last, we will save final model with pickle library, so it can be deploy on cloud platform.



```
import pickle
pickle.dump(grid_etc_best, open("Customer_Churn_Classification_model", "wb"))
load_Customer_Churn_Classification_model= pickle.load(open("Customer_Churn_Classification_model", "rb"))
```

```
y_pred = load_Customer_Churn_Classification_model.predict(x_test)

y_test = np.array(y_test)
data_prediction_by_model = pd.DataFrame()
data_prediction_by_model["Predicted Values"] = y_pred
data_prediction_by_model["Actual Values"] = y_test
data_prediction_by_model.sample(n=6)
```

	Predicted Values	Actual Values
2267	0	0
1639	0	0
74	1	1
437	1	1
306	0	0
2560	1	1

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.

You can get code of this case study from my GitHub Profile.