Customer Churn Analysis Based on Telecom Company

INTRODUCTION

Telecom service suppliers in the world are facing difficult market conditions and revenue declines due to high subscriber's churn rate. Telecommunications business derives its revenues/profits from the subscriptions from their customers. As it involves high cost of fixed network infrastructure establishments and its regular maintenance, customer loyalty is the key to their revenue generation. Nowadays, competition and government regulations have made easier for customers to switch from one network to other. People change their telecom service providers on minor of the issues like high billing, spam calls, emails, poor customer services to major issues like bad network connectivity, pathetic internet services speed, call drops etc. As a result, it is not surprising to learn that telecommunications companies have a high customer churn rate which is particularly problematic in this industry.

Problem Definition:

Customer churn is when a company's customers stop doing business with that company. Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, using marketing and sales budgets to gain additional customers. Existing customers will often have a higher volume of service consumption and can generate additional customer referrals.

Customer retention can be achieved with good customer service and products. But the most effective way for a company to prevent attrition of customers is to truly know them. The vast volumes of data collected about customers can be used to build churn prediction models. Knowing who is most likely to defect means that a company can prioritize focused marketing efforts on that subset of their customer base.

Preventing customer churn is critically important to the telecommunications sector, as the barriers to entry for switching services are so low.

Objective:

In this project, we build a model to predict how likely a customer will churn by analyzing its characteristics like information of customer details and information of services. The objective is to obtain a customer retention can be achieved with good customer service and products. But the most effective way for a company to prevent attrition of customers is to truly know them.

Preventing customer churn is critically important to the telecommunications sector Here, with the help of the given data set, we would be able to predict whether a customer is going to churn or not.

Steps of the project:

The project consists of the following process:

- Importing Libraries and Data Loading
- Exploratory Data Analysis
- Data Cleaning and Preprocessing
- Univariate and Bivariate data Visualization
- Feature Engineering
- Splitting the data in training and testing sets
- Assessing multiple algorithms
- Algorithm selected and Hyperparameter tuning

- Saving and loading of selected the model
- Conclusions and Remarked
-

Data Description (Features):

The dataset has total 21 columns and 7043 rows which has customer records of telecom company.

1. Customer ID---customer unique id
2. Gender — M/F
3. Senior Citizen — Whether the customer is a senior citizen or not (1, 0)
4. Partner — Whether customer has a partner or not (Yes, No)
5. Dependents — Whether customer has dependents or not (Yes, No)
6. Phone Service — Whether the customer has a phone service or not (Yes, No)
7. Multiplies — Whether the customer has multiple lines or not (Yes, No, No Phone Service)
8. Internet Service — Customer's internet service type (DSL, Fiber Optic, None)
9. Online Security — Whether the customer has Online Security add-on (Yes, No, No Internet Service)
10. Online Backup — Whether the customer has Online Backup add-on (Yes, No, No Internet Service)
11. Device Protection — Whether the customer has Device Protection add-on (Yes, No, No Internet Service)
12. Tech Support — Whether the customer has Tech Support add-on (Yes, No, No Internet Service)
13. Streaming TV — Whether the customer has streaming TV or not (Yes, No, No Internet Service)
14. Streaming Movies — Whether the customer has streaming movies or not (Yes, No, No Internet Service)
15. Contract — Term of the customer's contract (Monthly, 1-Year, 2-Year)
16. Paperless Billing — Whether the customer has paperless billing or not (Yes, No)
17. Payment Method — The customer's payment method (E-Check, Mailed Check, Bank Transfer (Auto), Credit Card (Auto))
18. Tenure— Number of months the customer has stayed with the company.
19. Total Charges-- The amount charged to the customer monthly.
20. Monthly Charges-- The total amount charged to the customer.
21. Churn-- The Churn column (target variable) indicates whether the customer departed within the last month or not.

Data Analysis: -

Data Analysis is a Process of Obtaining raw data and converting it into useful information by using charts, graphs, map and other visualization technique.

Importing Libraries

Import all the libraries at a time which are required for the process of Exploratory data analysis and building machine learning model.

```
#Importing required packages.
import pandas as pd
import numpy as np
import scipy.stats as st
#ploting libraries
import matplotlib.pyplot as plt
import seaborn as sns
#feature engineering
from sklearn.preprocessing import StandardScaler, LabelEncoder
#train test split and cross validation
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from scipy.stats import zscore
#metrics
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report,accuracy_score
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import plot_roc_curve
#ML models for classification
import sklearn
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Extracting dataset

Dataset contain 7043 rows and 21 columns. We will use this data for prediction of our machine learning model from which dataset 70% of data is used for training a model and 30% for testing a model.

- Loading and saving the given Dataset in a variable.
- Find out the shapes of our dataset.
- Display the columns of dataset.

```
#loading data sets
df = pd.read_csv('customer_churn.csv')

# No. of rows and columns
df.shape
```

```
(7043, 21)
```

```
# column names
df.columns
```

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

Checking datatypes of columns

The **info** method is used to obtain summary of the dataset, including the column names and their data types, the number of non-null values, and the amount of memory used by the dataset.

```
# check datatypes and memory captured
df.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
```

Observation

- There is both numerical and categorical type of data is present.
- We observed that Total charges is floating value but here shows the object that mean any space value may be in total charges we have to check.

Statistical analysis

Statistical analysis of Numerical Features represents the count, min, max, median ,25% and 75% value of each feature which is help us to find out the presence of Nan value, outliers and skewness in our datasets.

```
1 df.describe(exclude=[object]).T
```

	count	mean	std	min	25%	50%	75%	max
SeniorCitizen	7043.0	0.162147	0.368612	0.00	0.0	0.00	0.00	1.00
tenure	7043.0	32.371149	24.559481	0.00	9.0	29.00	55.00	72.00
MonthlyCharges	7043.0	64.761692	30.090047	18.25	35.5	70.35	89.85	118.75

Observations:

- There is no column with single unique values
- Null values is not present because the value of count for all columns are same (7043).
- Skewness are present because higher difference between 75% and max value of Monthly charges and
- Total charges is descriptive type data but did not came in descriptive analysis because may be some incorrect variable is present in this column.

Statistical analysis of categorical Features represents the top, count, unique and frequent value of each column. which is help us to find out uniqueness and mode value of each column in our datasets.

```
1 df.describe(include=[object]).T
```

	count	unique	top	freq
customerID	7043	7043	7590-VHVEG	1
gender	7043	2	Male	3555
Partner	7043	2	No	3641
Dependents	7043	2	No	4933
PhoneService	7043	2	Yes	6361
MultipleLines	7043	3	No	3390
InternetService	7043	3	Fiber optic	3096
OnlineSecurity	7043	3	No	3498
OnlineBackup	7043	3	No	3088
DeviceProtection	7043	3	No	3095
TechSupport	7043	3	No	3473
StreamingTV	7043	3	No	2810
StreamingMovies	7043	3	No	2785
Contract	7043	3	Month-to-month	3875
PaperlessBilling	7043	2	Yes	4171
PaymentMethod	7043	4	Electronic check	2365
TotalCharges	7043	6531		11
Churn	7043	2	No	5174

Check null value

Check missing value present in any columns by coding and visualization.

```
df.isnull().sum().sum()
```

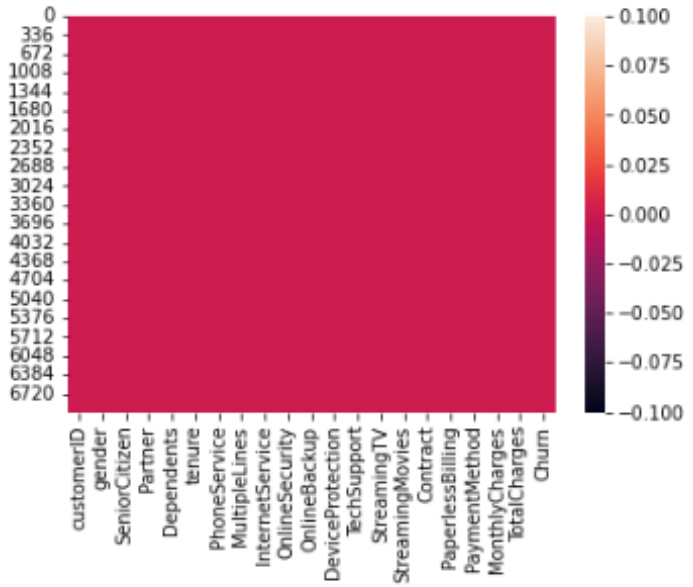
```
0
```

```
# find out in which column missing value is present

[features for features in df.columns if df[features].isnull().sum()>0]
```

```
[]
```

```
sns.heatmap(df.isnull())##check missing value through visualization
```

```
<AxesSubplot:>
```



We observed that no missing value present in our dataset.

Check duplicate value

```
df.duplicated().sum()
```

```
0
```

```
# if duplicate value present drop this row
df.drop_duplicates(subset=['customerID'])
```

No Duplicate value present there if any duplicates present then drop this particular row.

Data cleaning

In the process of data cleaning, we fixing the missing, incorrect, incomplete, and duplicate value.

Total Charges

we observe that the column Total Charges was wrongly detected as an object. This column represents the total amount charged to the customer and it is, therefore, a numeric variable. For further analysis, we need to transform this column into a numeric data type.

```
df.loc[df['TotalCharges']==" "] # check for space value in totalcharges
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtec
488	4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	No	
753	3115-CZMZD	Male	0	No	Yes	0	Yes	No	No	No internet service	No internet service	No inte ser
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	Yes	
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No inte ser
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	Yes	
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No inte ser
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No inte ser
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No inte ser
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No inte ser
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	Yes	
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	Yes	

Observed that we have 11 rows that contain space value firstly we will replace it with nan value then impute this nan value by using mean value after that change datatype of column object to float.

```
# replace totalcharges space value by nan .

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

```
# find out in which column missing value is present

[features for features in df.columns if df[features].isnull().sum()>0]
```

```
['TotalCharges']
```

Type casting- change the datatype of column totalcharges

```
df['TotalCharges']=df['TotalCharges'].astype(float)
```

```
df.TotalCharges.dtype # type casting done
```

```
dtype('float64')
```

now in totalcharges column the value of space is replace by nan value

Handling the nan value by replacing by it with mean

```
df["TotalCharges"].isnull().sum()
```

```
11
```

Replace the null value with mean value of Totalcharges

```
df['TotalCharges'].fillna(df["TotalCharges"].mean(), inplace=True)
```

```
df["TotalCharges"].isnull().sum()
```

```
0
```

Now Total Charges is float and no null value present.

Payment Method

Some payment method denominations contain in parenthesis the word automatic. These denominations are too long to be used as for visualizations. Therefore, we remove this clarification in parenthesis from the entries of the Payment Method column.

```
# remove (automatic) from payment method names

df['PaymentMethod'] = df['PaymentMethod'].str.replace(' (automatic)', '', regex=False)
```

```
# unique elements of the PaymentMethod column after the modification
df.PaymentMethod.unique()
```

```
array(['Electronic check', 'Mailed check', 'Bank transfer', 'Credit card'],
      dtype=object)
```

No Internet Services

The customer who don't have Internet services they are also not able to do any facility by using internet like OnlineSecurity, OnlineBackup, StreamingTV, StreamingMovies, and deviceProtection, So we can change the columns which has No internet service into 'No' .

```
for col in df:
    if col in ("OnlineSecurity","OnlineBackup","DeviceProtection","TechSupport","StreamingTV","StreamingMovies"):
        df[col] = df[col].replace({'No internet service':'No'})
```

```
df["OnlineSecurity"].unique()
```

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

```
df["OnlineBackup"].unique()
```

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

```
df["DeviceProtection"].unique()
```

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

```
df["StreamingTV"].unique()
```

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

```
df["StreamingMovies"].unique()
```

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

Tenure

change tenure into group of years

```
def month_to_year(df):
    if df["tenure"] <=12:
        return "0_1_year"
    elif (df["tenure"] > 12) & (df["tenure"] <= 24 ):
        return "1_2_year"
    elif (df["tenure"] > 24) & (df["tenure"] <= 36) :
        return "2_3_year"
    elif (df["tenure"] > 36) & (df["tenure"] <= 48) :
        return "3_4_year"
    elif df["tenure"] > 48 & (df["tenure"] <= 60):
        return "4_5_year"
    elif df["tenure"] > 60 & (df["tenure"] <= 72):
        return "5_6_year"
df["Tenure_Group"] = df.apply(lambda df:month_to_year(df),axis = 1)

```

```
1 df["Tenure_Group"]
```

```
0        0_1_year
1        2_3_year
2        0_1_year
3        3_4_year
4        0_1_year
           ...
7038     1_2_year
7039     4_5_year
7040     0_1_year
7041     0_1_year
7042     4_5_year
Name: Tenure_Group, Length: 7043, dtype: object
```

Check unique values present in each Column

```
1 df.nunique().to_frame("Unique Values")
```

	Unique Values
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

Observation:

- Customer Id, monthly charges and total charges has higher unique value.
- All value of customer id is unique and not useful for prediction so we can drop this column.
- No anyone unique value present there.

Drop unnecessary columns

- Drop column customerID which has lots of unique value and no need for model prediction.
- Drop column tenure because I added new column tenure group which is more understandable.

```
df.drop("customerID",inplace = True, axis = 1)
df.drop("tenure",inplace = True, axis = 1)
```

```
df.columns
```

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

Segregate the Numerical and Categorical features

Now we separate numerical and categorical columns so that we can visualize our data easily.

```
categorical = [feature for feature in df.columns if df[feature].dtypes=='object']
df[categorical].sample(5)
```

	gender	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	St
2787	Male	Yes	Yes	Yes	Yes	No	No	No	No	No	No	
3313	Male	No	No	Yes	No	No	No	No	No	No	No	
5412	Male	Yes	No	Yes	Yes	Fiber optic	No	Yes	No	Yes	No	
4583	Female	Yes	No	Yes	Yes	DSL	Yes	Yes	Yes	Yes	Yes	
138	Male	No	Yes	Yes	No	No	No	No	No	No	No	

```
numerical = [feature for feature in df.columns if df[feature].dtypes!='object']
df[numerical].sample(5)
```

	SeniorCitizen	MonthlyCharges	TotalCharges
4678	0	94.20	193.80
1933	0	19.70	415.90
3365	0	19.65	358.15
3375	0	19.90	1389.35
6766	0	19.75	309.35

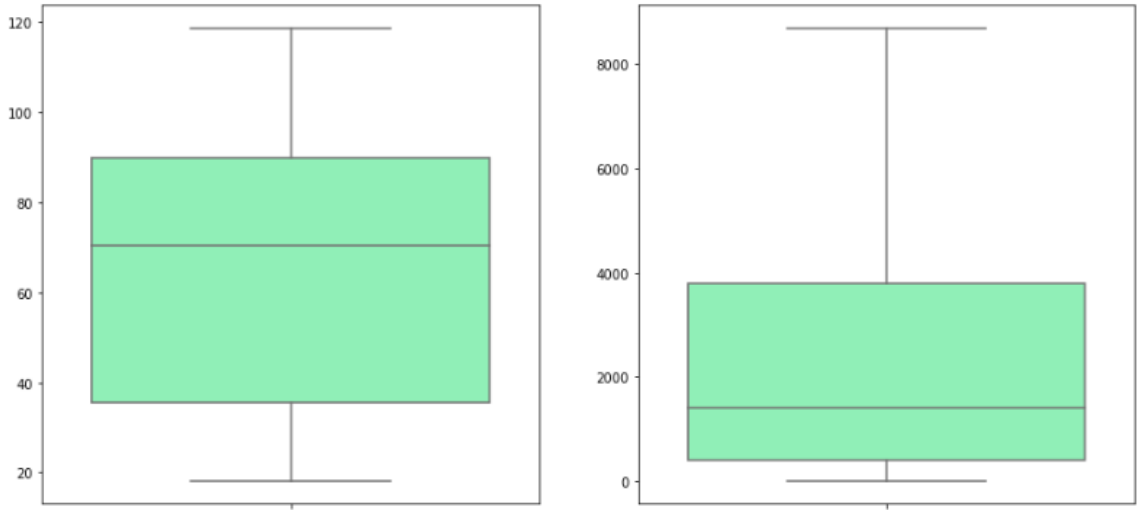
Univariate Analysis

Check Outliers

We always check outliers for numerical features. Here Senior citizen is numerical value but nature is categorical type of data so we can check outliers for Monthly Charges and Total Charges features only.

```
1 f, axes = plt.subplots( ncols=2, figsize=(15, 7))
2 sns.boxplot(data=df["MonthlyCharges"],palette='rainbow',ax = axes[0])
3 sns.boxplot(data=df["TotalCharges"], palette='rainbow',ax = axes[1])
```

```
<AxesSubplot:>
```

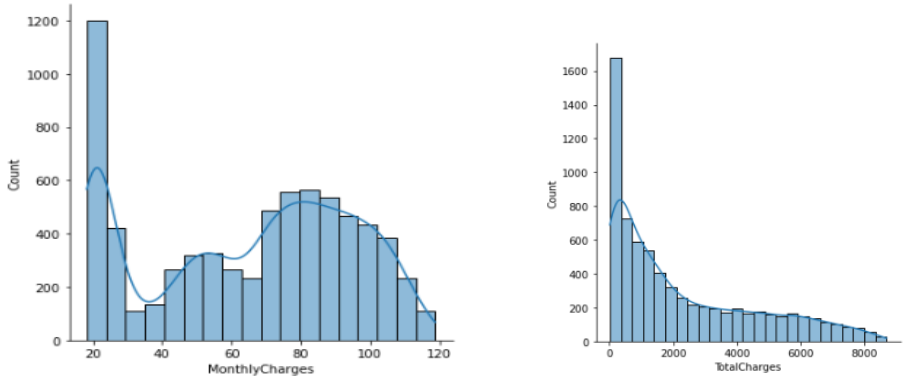


Observation:

- No outliers present in Total charges and monthly charges of data sets.
- If any outliers are present then it will be removed by using Z-score or IQR technique.

Check Skewness

Let's we will check skewness is present in our dataset by using distribution plot.



Observation:

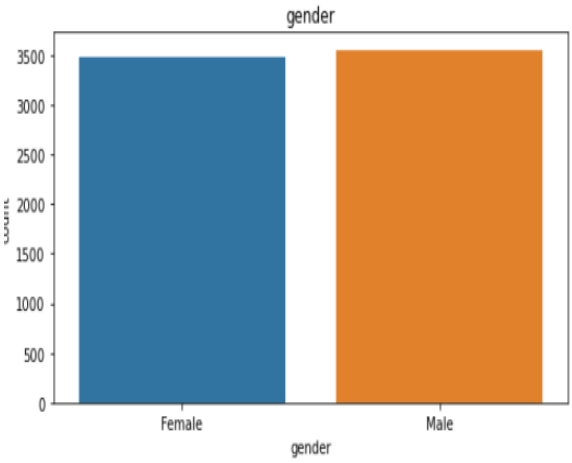
Skewness is present in total charges and monthly charges so later we have to remove it by using any transform technique.

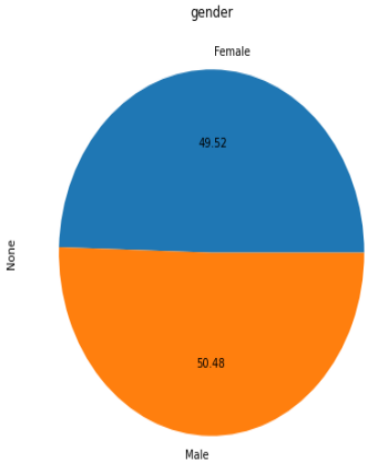
Visual Analysis of Categorical Variable

We will visualize all categorical variable by using count plot and pie chart.

- Gender

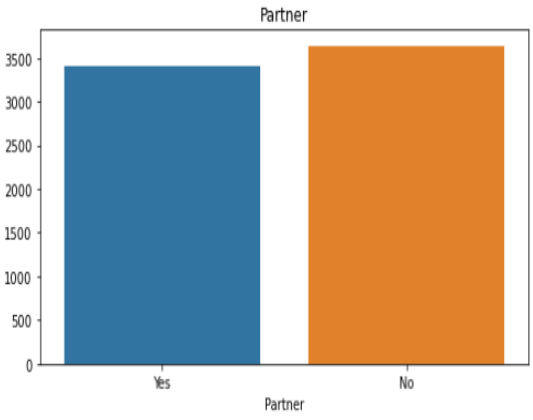
Count and percentage of male and female user almost same.

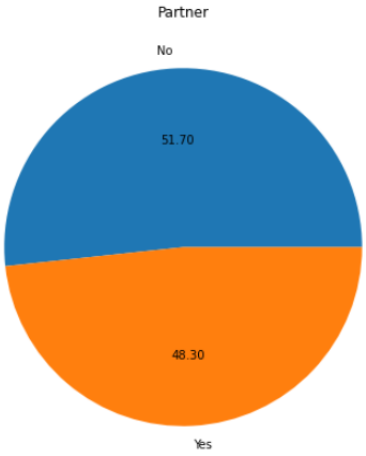




- Partner

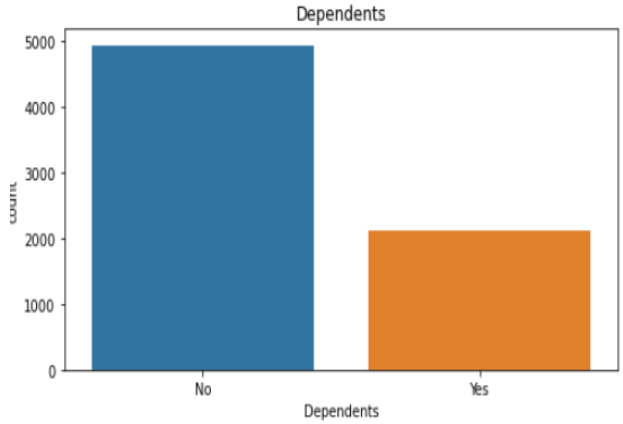
Count and percentage of partner is little bit difference.

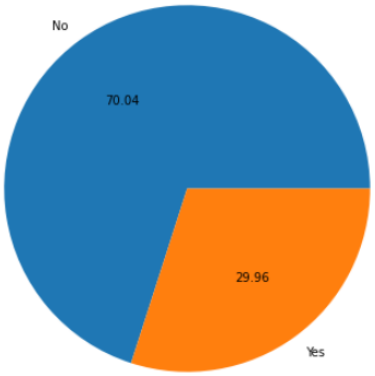




- ## Dependence

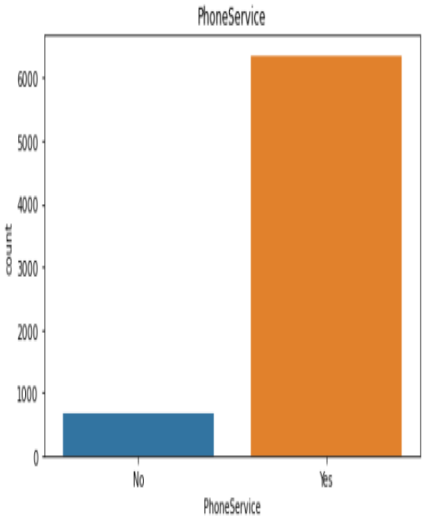
The percentage of independence customer are 40% higher than dependent customer.

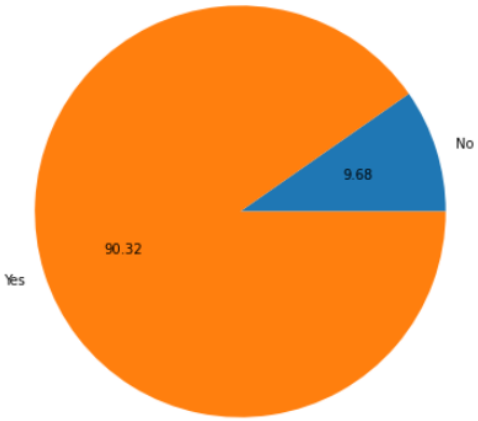




- ## Phone service

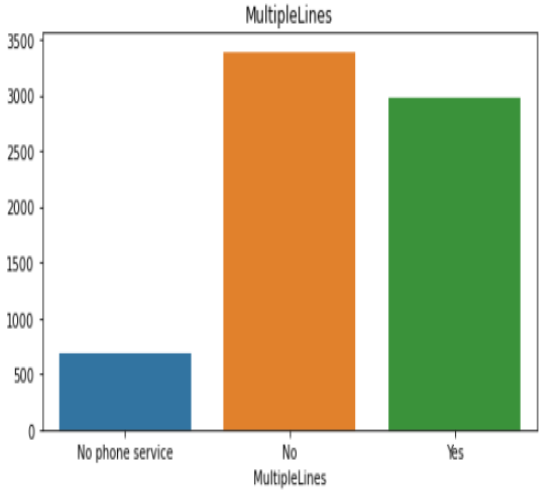
90% of customer preferred for phone service.

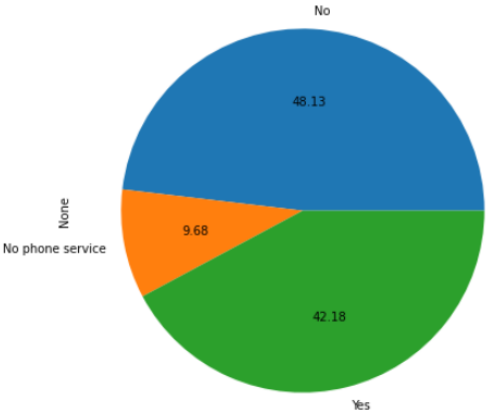




- ## Multiple lines

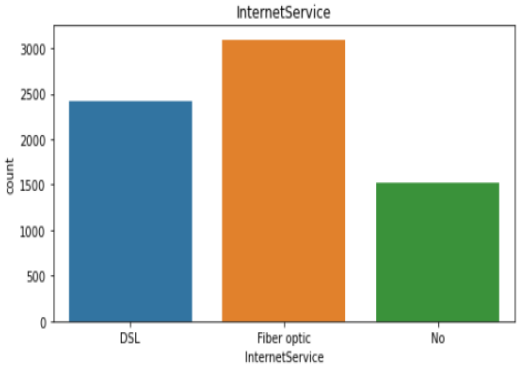
some customers are preferred for multiple lines but maximum customer is preferring for single line and very few people do not have any services.

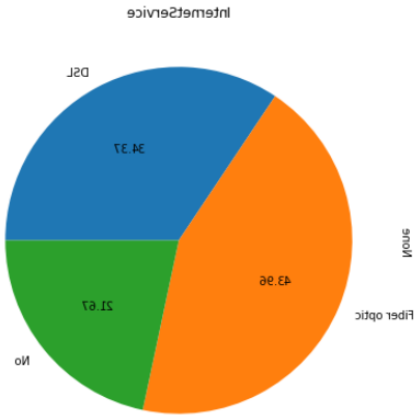




- <u>Internet Service</u>

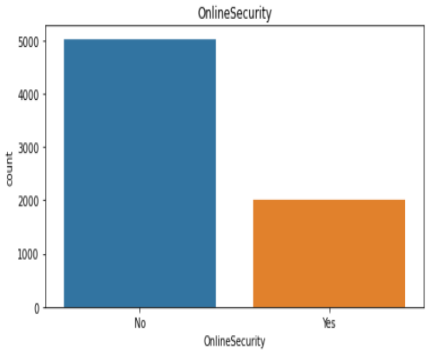
The customer who had Internet services almost preferred for fiber optics.

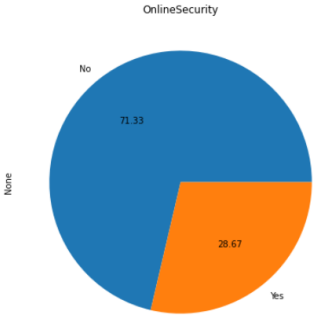




- <u>Online security</u>

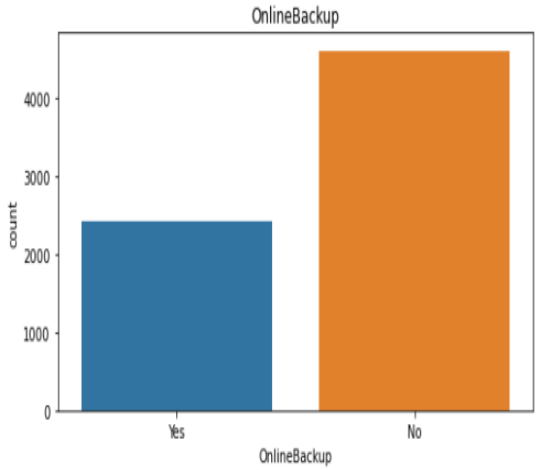
71% customer doesn't have online security because the customer doesn't have internet service is not able to do this service.

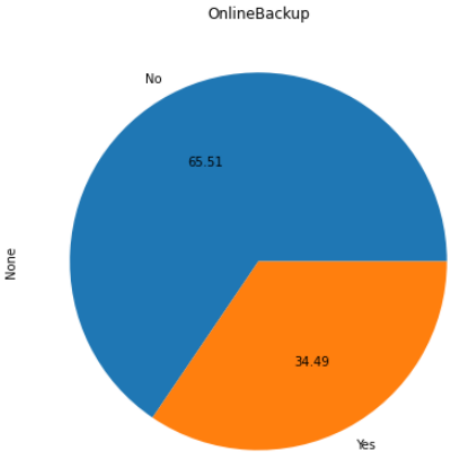




- ## Online Backup

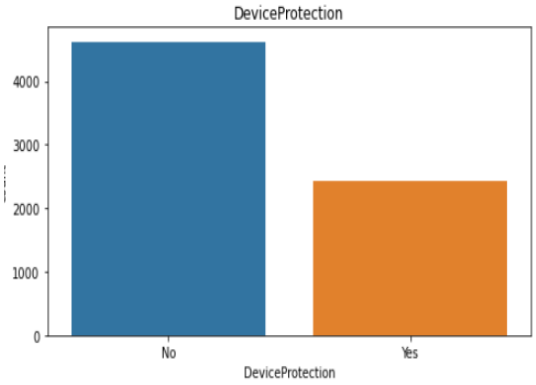
65% customer doesn’t have online backup facility because the customer doesn’t have internet service is not able to do this service.

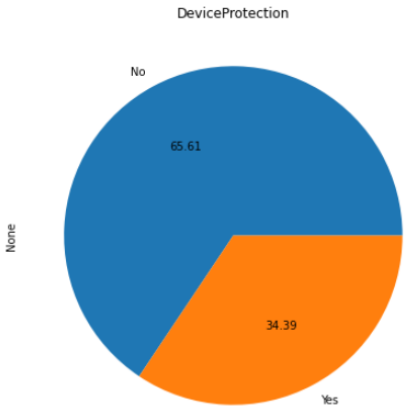




- ## Device Protection

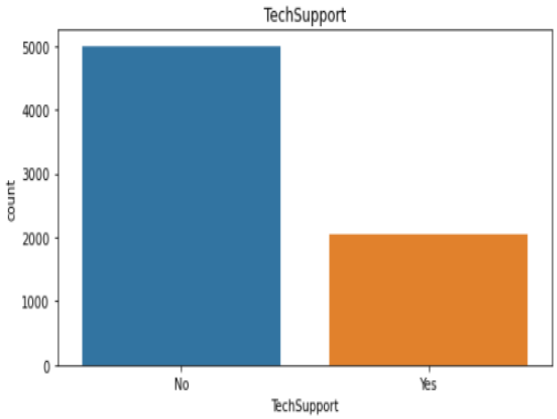
The number of customers who has Device protection is less because the customer doesn’t have internet service is not able to do this service.

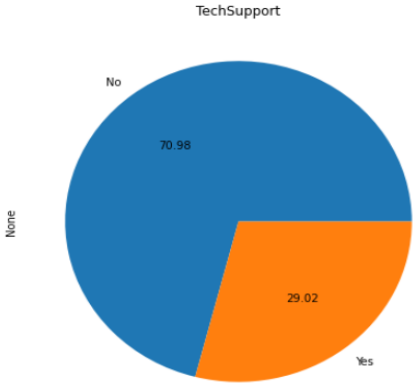




- ## Tech Support

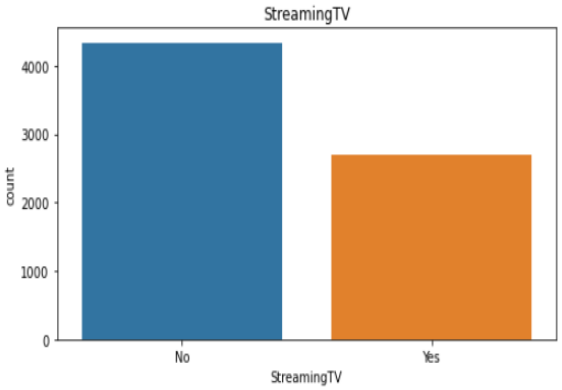
The number of customers who has tech Support facility is less because the customer doesn’t have internet service is not able to do this facility.

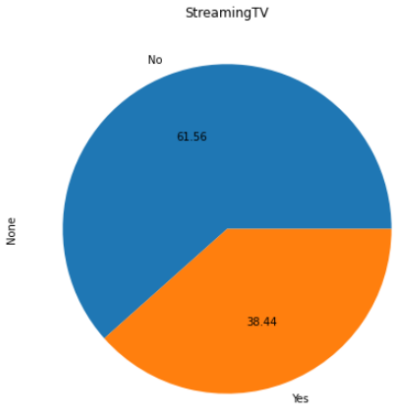




- Streaming TV

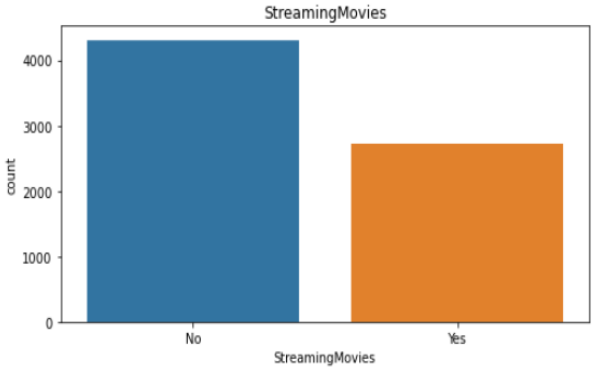
The No. of customer who had no streaming TV is higher because the customer doesn't have internet service is not able to do Streaming TV.

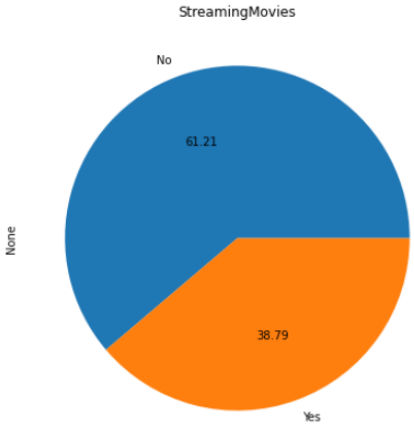




- Streaming Movie

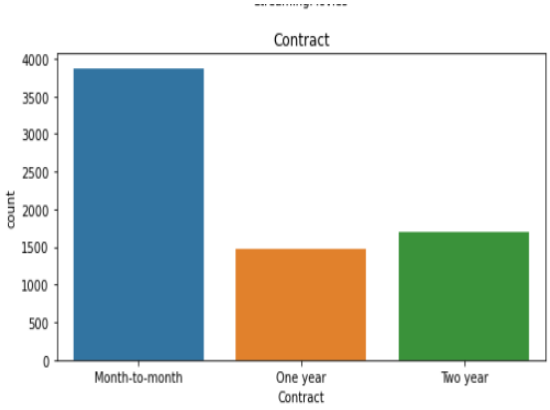
The No. of customer who had no streaming Movie is higher because the customer doesn't have internet service is not able to do Streaming Movie.

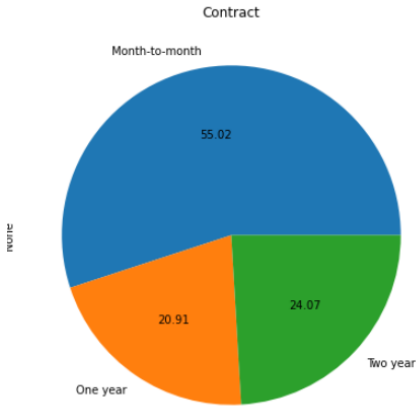




- Contract

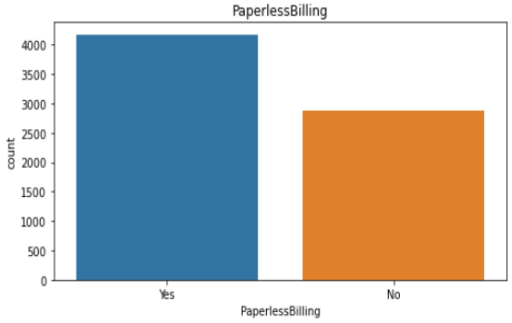
The customer having mouth-to-mouth contract will have higher chance of churn. So, the telecom company should focus this type of customer first.

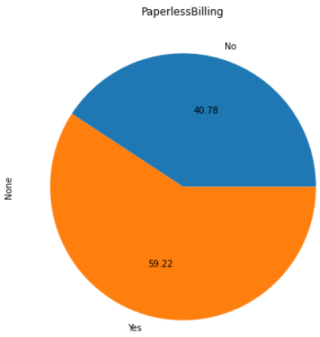




- Paperless Billing

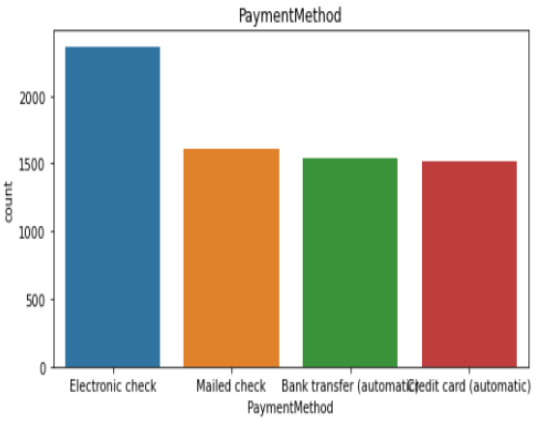
Maximum customer preferred paperless billing, so telecom operator should provide online payment mode.

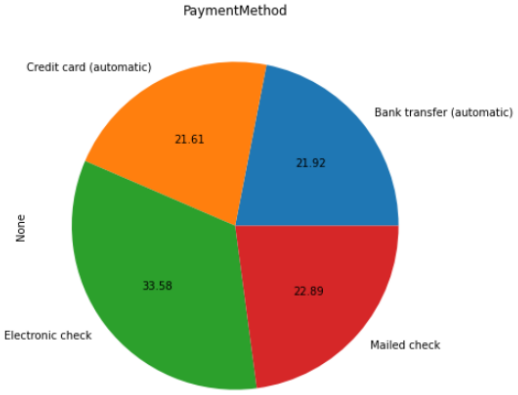




- Payment method

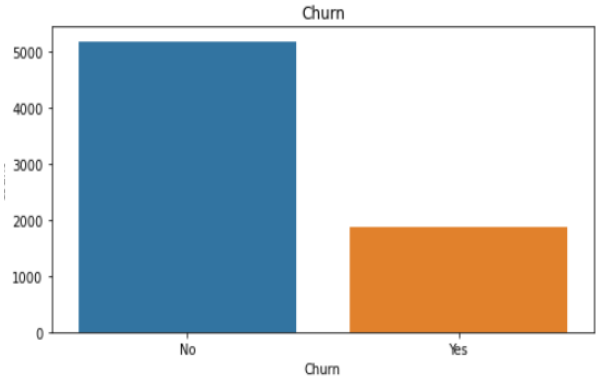
Maximum customer preferred payment by electronic cheque.

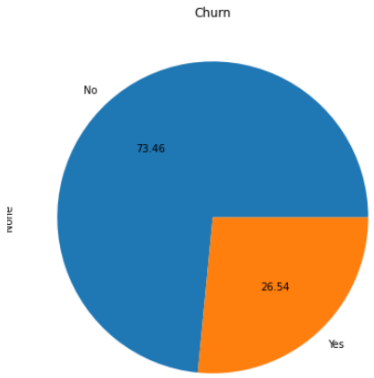




- Churn

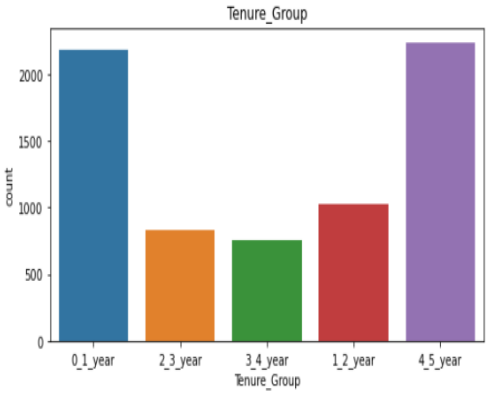
The customer churn rate is 26% in our dataset.

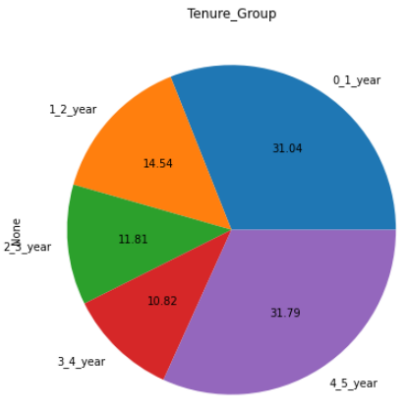




- Tenure Group

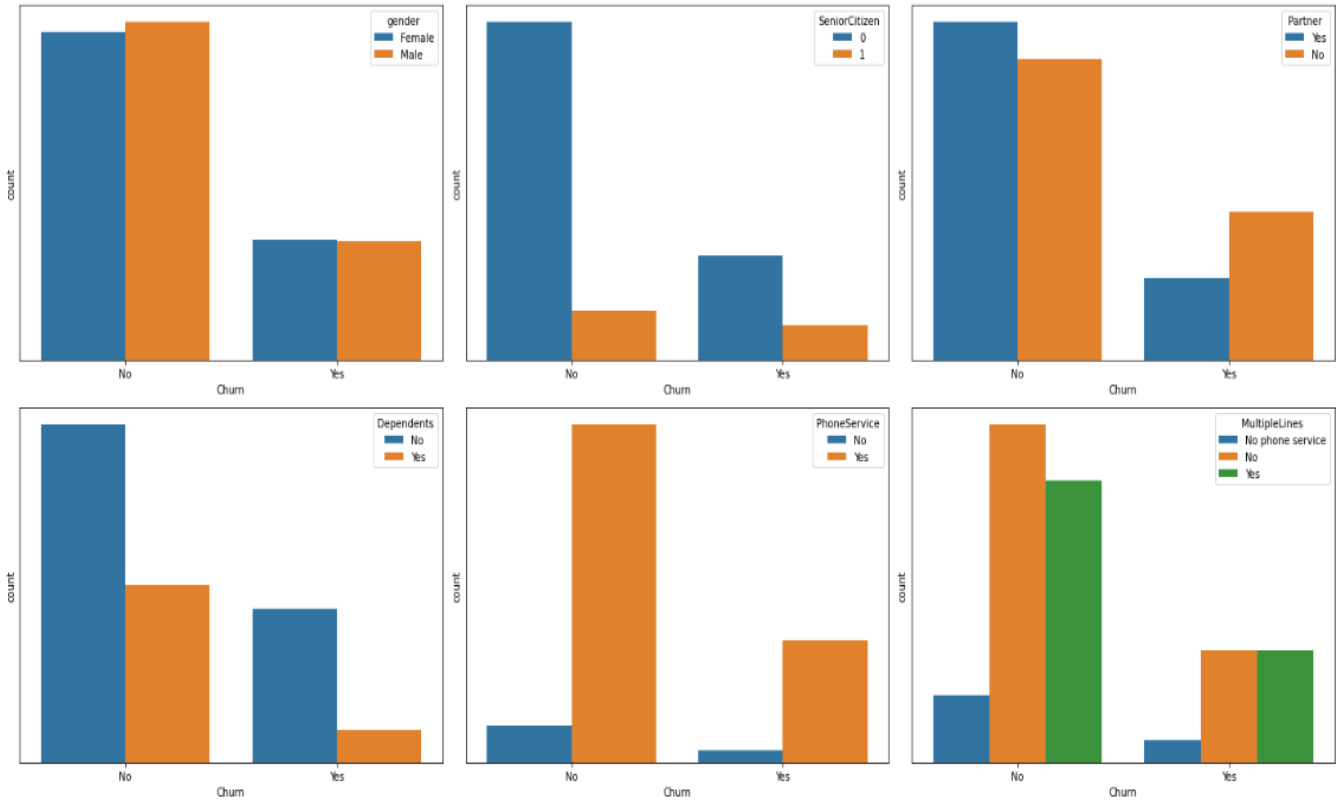
  The customer who belongs from tenure group medium tenure group has chance to churn higher.

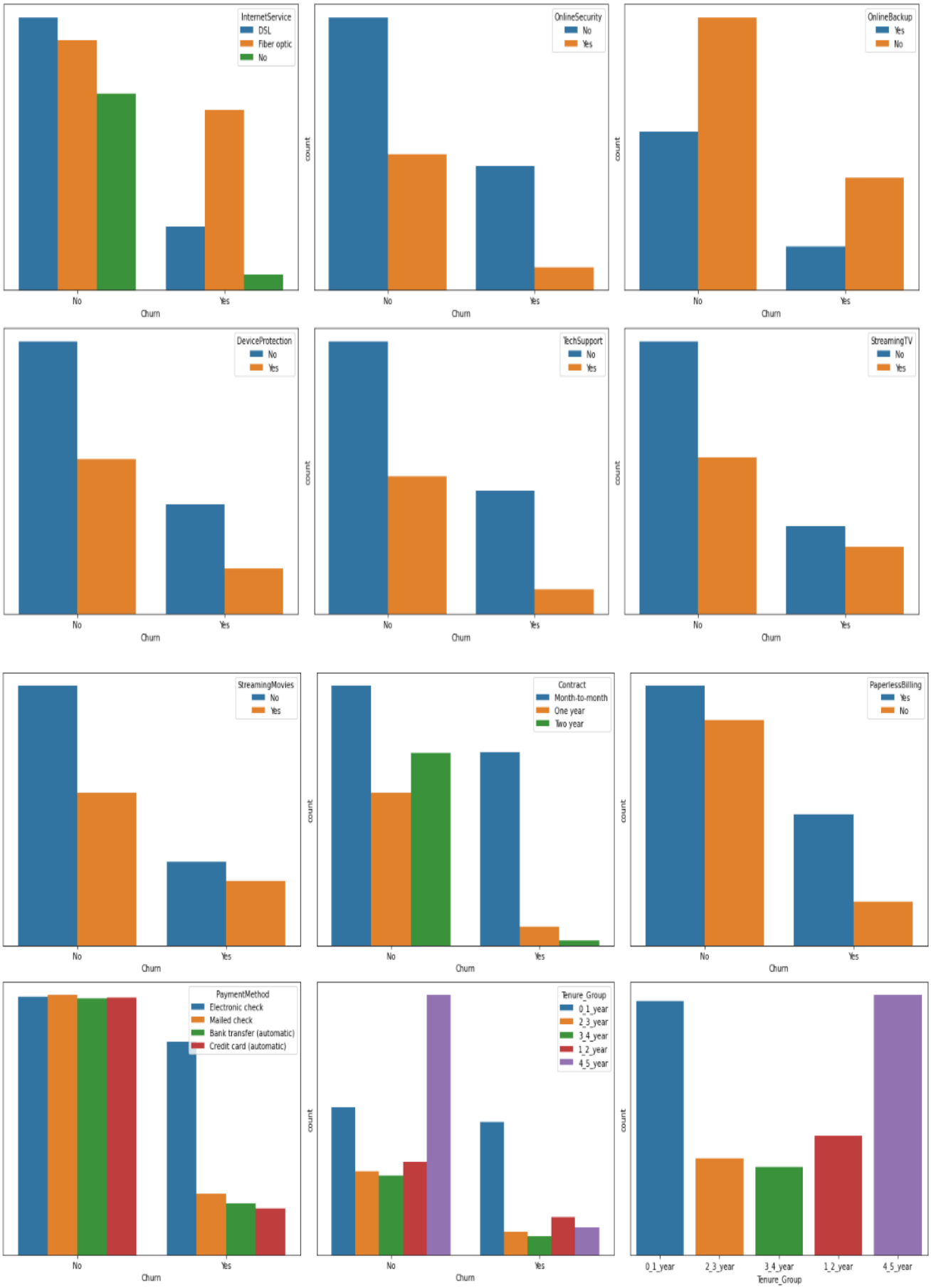




Bivariate Analysis:

Visualize all categorical features with respect to target variable churn by using count plot.




InternetService
DSL
Fiber optic
No
OnlineSecurity
No
Yes
OnlineBackup
Yes
No
DeviceProtection
No
Yes
TechSupport
No
Yes
StreamingTV
No
Yes
StreamingMovies
No
Yes
Contract
Month-to-month
One year
Two year
PaperlessBilling
Yes
No
PaymentMethod
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
Tenure_Group
0_1 year
2_3 year
3_4 year
1_2 year
4_5 year
count
Churn
No
Yes
Tenure_Group
0_1 year
2_3 year
3_4 year
1_2 year
4_5 year

Observation From above graph

1.The count of churn is no for maximum customer but that is not too much dependent on gender means churn rate is same for both male and female So, we can drop the gender column because it is not useful for model prediction.

2.We saw that in above pie chart only 16% Senior Citizen exist in telecom operator but now I observed that the churn rate is higher as compare to younger customers may be its possible the older people is not used phone services above the age of 70.

There is Less Senior Citizens among the group and within the senior citizens churning is less, hence we would need to focus on the younger group to reduce the number of churns through various technologies and innovations they can more relate to.

3.In given data set 48% customer preferred for partnership but the people who is partner the churn rate is less. Churning is occurring more among the customers who aren't partners. Hence more amount of marketing would need to brough among non-partner to make them a partner or build their trust and affinity to the product.

4.70 % people are not dependent. There are more people with no dependents, If the customer has a dependent, there can be seen a higher chance to churn.

5.We saw 90% of them have phone services. People who don't have phone service is more likely to churn

6.Multiple lines are present among 42%, Only single lines are present among 47% and No phone service is among 9.7% Highest people are churning if they have multiple line. There is almost equal churning in single line and No phone service

7.Fibre optics is used by maximum customers, Highest churning is seen for People Using Fiber Optic

8.Online backup is not present among 43% users; Highest churning can be seen when there is no online backup.

9.Device protection is not present in 44%, There is highest churning when there is no Device Protection. Hence Having a Device Protection is more important than having internet service.

10.Tech support is not present among 49%, The highest churning when there is no tech support. Hence Having a tech support is more important as compare to having internet service.

11.Streaming TV is not present among 40%. Having or not having streaming TV does not contribute much towards churning. Both who had and didn't have Streaming TV have almost equally churned

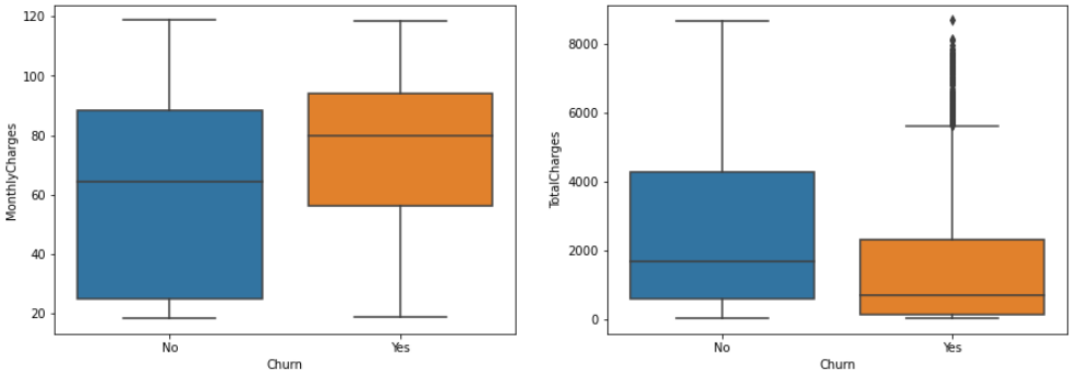
12.55% customers have month to month contract. Churning is highest among month-to-month contracts So need to focus who had among month-to-month contracts.

13.The Customer who preferred Paperless billing is among 60% employees and churning is highest among them.

14.Electronic cheque is used by 34% employees and churning is highest among them.

15.The people whose tenure is higher 4-5 years is less chance to churn.

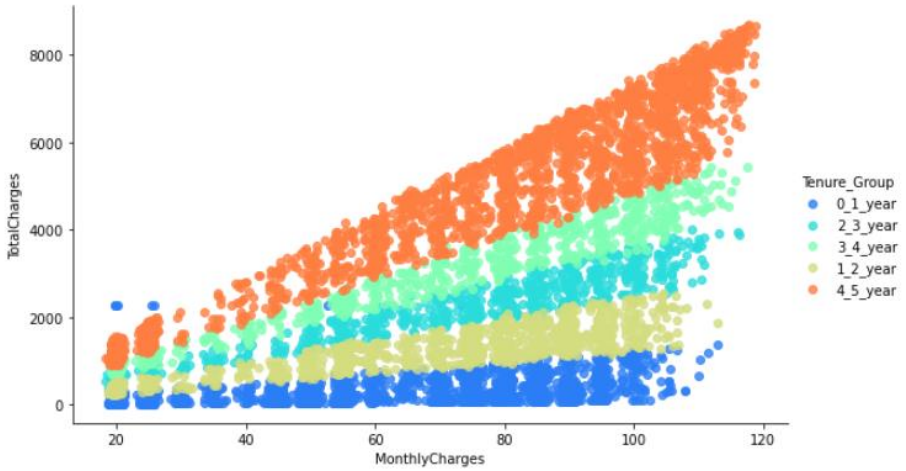
Bivariate Analysis of numerical variable with respect to customer churn



Observation:

When the total charges increase the churn rate is also increases. When monthly charges are less then less chance to churn the no. of users.

Relation between Total Charges Monthly Charges and Tenure Group



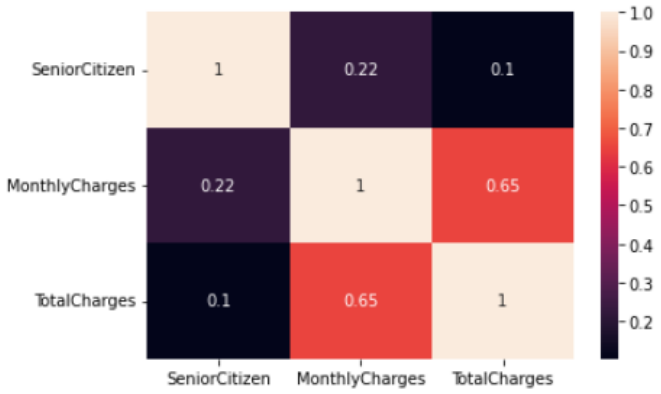
Observation:

when the monthly charges are increases obviously Total charges is also increases but who belongs from 4–5-year tenure group pay maximum bill and less chance to churn so, need to be focus the customer who belongs from tenure-group 1-2 because they played minimum bill and more chance of churn.

Correlation

Now we check correlation of numerical variable

```
sns.heatmap(df.corr(),annot=True)
```



Observation:

Senior citizen is categorical type data but the monthly charges and Total charges is highly Correlated to each other's.

Data Preprocessing

After understanding the nuances of our dataset and the main issues in the data through EDA, data preprocessing comes into play by preparing our dataset for use in the model.

Some preprocessing techniques in order to make the data more usable. These techniques will facilitate its use in machine learning (ML) algorithms, reduce the complexity to prevent overfitting, and result in a better model.

Remove Skewness

In above distribution plot we notice that skewness is present in Totalcharges and monthlycharges column now we remove that skewness by using Transform technique.

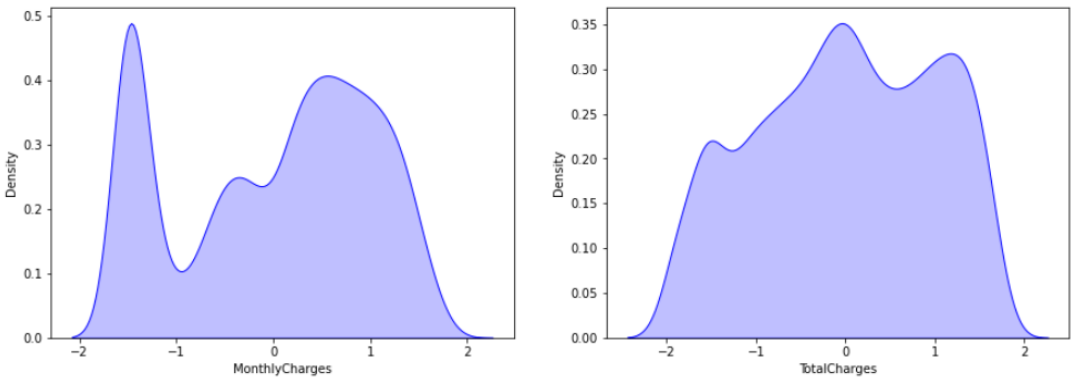
```
#create a list of columns to remove skewness from these columns

skew=["MonthlyCharges","TotalCharges"]
```

```
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method="yeo-johnson")
df[skew]=scaler.fit_transform(df[skew].values)
df[skew].skew()
```

```
MonthlyCharges    -0.259035
TotalCharges      -0.144899
dtype: float64
```

Check skewness are removed are not

```
fig, ax = plt.subplots(ncols=2, nrows=1, figsize=(15,5))
index = 0
ax = ax.flatten()
for col, value in df[skew].items():
    sns.distplot(value, ax=ax[index], hist=False, color="b",kde_kws={"shade": True})
    index += 1

plt.show()
```



Now we have successfully removed the skewness from above two columns.

Label Encoding

convert categorical data into numerical variable by using Label encoding technique.

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
#for changing all catigorical value into numeric value

df=df.apply(le.fit_transform)
```

```
df.head()
```

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV
0	0	0	1	0	0	1	0	0	1	0	0	0
1	1	0	0	0	1	0	0	1	0	1	0	0
2	1	0	0	0	1	0	0	1	1	0	0	0
3	1	0	0	0	0	1	0	1	0	1	1	0
4	0	0	0	0	1	0	1	0	0	0	0	0

Model Building

Churn is a target variable and it is binary type so, we have to take classification models to predict

Feature and Label Selection:

First, we create a variable **x** to store the **independent attributes** of the dataset. Additionally, we create a variable **y** to store only the **target variable** (`Churn`).

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

```
print(x.shape)
print(y.shape)
```

```
(7043, 19)
(7043,)
```

Balancing the target dataset:

Given dataset is unbalanced so by using oversampling method we will balance our data.

```
from imblearn.over_sampling import SMOTE

smt=SMOTE()# instance of smote
```

```
trainx,trainy=smt.fit_resample(x,y)
```

```
trainy.value_counts()
```

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

Multicollinearity:

By using vector inflammation factor, we will check if any multicollinearity presents in our columns.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
def cal_vif(x):
    vif=pd.DataFrame()
    vif['variables']=x.columns
    vif["vif_factor"]=[variance_inflation_factor(x.values,i) for i in range (x.shape[1])]
    return(vif)
```

Out[75]:

	variables	vif_factor
0	gender	1.928279
1	SeniorCitizen	1.357040
2	Partner	2.812292
3	Dependents	1.960382
4	PhoneService	12.011610
5	MultipleLines	2.779784
6	InternetService	4.432031
7	OnlineSecurity	2.032649
8	OnlineBackup	2.252424
9	DeviceProtection	2.397355
10	TechSupport	2.126398
11	StreamingTV	3.197558
12	StreamingMovies	3.202355
13	Contract	3.830085
14	PaperlessBilling	2.706758
15	PaymentMethod	2.962178
16	MonthlyCharges	20.023213
17	TotalCharges	52.948427
18	Tenure_Group	22.761205

We observed that Total charges column is suffering from multicollinearity so we can delete it.

Data normalization:

Data normalization transforms multiscale data to the same scale. After normalization, all variables have a similar influence on the model, improving the stability and performance of the learning algorithm.

Scaling our data by using standard scaler.

```
from sklearn.preprocessing import StandardScaler
SC=StandardScaler()
x=SC.fit_transform(x)
x
```

```
array([[-1.00955867, -0.43991649,  1.03453023, ..., -1.13176632,
        -1.55971787, -1.18918503],
       [ 0.99053183, -0.43991649, -0.96662231, ..., -0.38773977,
         0.26850167,  0.01331751],
       [ 0.99053183, -0.43991649, -0.96662231, ..., -0.51731743,
        -1.32179184, -1.18918503],
       ...,
       [-1.00955867, -0.43991649,  1.03453023, ..., -1.14221613,
        -0.94636309, -1.18918503],
       [ 0.99053183,  2.27315869,  1.03453023, ...,  0.23297901,
        -1.00198217, -1.18918503],
       [ 0.99053183, -0.43991649, -0.96662231, ...,  1.4723266 ,
         1.56988499,  1.21582004]])
```

Now our data is in range between 0 to 1.

Splitting the data in training and testing sets

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.20, random_state=100)
```

```
print("shape of x_train= ",x_train.shape)
print("shape of y_train= ",y_train.shape)
print("shape of x_test= ",x_test.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of x_train=  (5634, 19)
shape of y_train=  (5634,)
shape of x_test=  (1409, 19)
shape of y_test=  (1409,)
```

The first step when building a model is to **split the data into two groups**, which are typically referred to as **training and testing sets**. The training set is used by the machine learning algorithm to build the model. The test set contains samples that are not part of the learning process and is used to evaluate the model's performance.

Finding Best Random State and splitting the data:

```
from sklearn.linear_model import LogisticRegression
max_accu=0 #maxi accuracy define as 0
max_rs=0# best random state for which max accurecy achived

lr=LogisticRegression()

for i in range(0,200):
    x_train, x_test, y_train, y_test = train_test_split( x, y, test_size=.25, random_state=i)

    lr.fit(x_train,y_train)

    pred=lr.predict(x_test)# predicted target variable

    acc=accuracy_score(y_test,pred)# accuracy score

    print("testing accuracy ",acc, " random state =", i)
    if acc>max_accu:
        max_accu=acc
        max_rs=i
        print(" max accuracy score",acc, " max random state =",i)
```

```
print(" max accuracy score",max_accu, " max random state =", max_rs)
```

```
max accuracy score 0.8256672345258376  max random state = 150
```

Classification model selection

Churn is a target variable and it is binary type so, we have to take classification models to predict.

Algorithm selection is a key challenge in any machine learning project since there is not an algorithm that is the best across all projects. Generally, we need to evaluate a set of potential candidates and select for further evaluation those that provide better performance.

In this project, we compare 6 different algorithms

1. **Ada booster**
2. **K Nearest Neighbors**
3. **Logistic Regression**
4. **Support Vector Machines**
5. **Random Forest**
6. **Decision Tree**
7. **Gradient Boosting**

Now we will define a function for model accuracy and cross validation and called it.

```
# model_selection is function  created bcz when we want to use another model then only call t
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.20, random_state=26)
def model_selection(algorithm_instance,x_train, x_test, y_train, y_test):
    algorithm_instance.fit(x_train,y_train)
    model1_pred=algorithm_instance.predict(x_test)

    print("accuracy",accuracy_score(y_test,model1_pred)*100)
    print(confusion_matrix(y_test,model1_pred))
    print(classification_report(y_test,model1_pred))



def model_cv(algorithm_instance,trainx,trainy):
    for j in range(4,10):
        lsscore=cross_val_score(algorithm_instance,trainx,trainy,cv=j)
        print(lsscore)
        lsc=lsscore.mean()
        print(" At cv =",j)
        print("cross validation score ",lsc*100)
```

AdaBoostClassifier

```
from sklearn.ensemble import AdaBoostClassifier
ad=AdaBoostClassifier()
model_selection(ad,x_train, x_test, y_train, y_test)
model_cv(ad,trainx,trainy)
```

```
accuracy 81.40525195173882
[[941 108]
 [154 206]]
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      1049
           1       0.66      0.57      0.61       360

    accuracy                           0.81      1409
   macro avg       0.76      0.73      0.74      1409
weighted avg       0.81      0.81      0.81      1409

[0.72825667 0.80363355 0.83764979 0.85233862]
 At cv = 4
cross validation score  80.54696559721684
```

KNeighborsClassifier

```
knn=KNeighborsClassifier()
model_selection(knn,x_train, x_test, y_train, y_test)
model_cv(knn,trainx,trainy)
```

```
accuracy 76.72107877927608
[[891 158]
 [170 190]]
              precision    recall  f1-score   support

           0       0.84      0.85      0.84      1049
           1       0.55      0.53      0.54       360

    accuracy                           0.77      1409
   macro avg       0.69      0.69      0.69      1409
weighted avg       0.76      0.77      0.77      1409

[0.76420564 0.78044066 0.77077696 0.78933127]
 At cv = 4
cross validation score  77.6188635485118
```

RandomForestClassifier

```
rf= RandomForestClassifier(random_state=26)
model_selection(rf,x_train, x_test, y_train, y_test)
model_cv(rf,trainx,trainy)
```

```
accuracy 79.347054648687
[[944 105]
 [186 174]]
              precision    recall  f1-score   support

           0       0.84      0.90      0.87      1049
           1       0.62      0.48      0.54       360

    accuracy                           0.79      1409
   macro avg       0.73      0.69      0.71      1409
weighted avg       0.78      0.79      0.78      1409

[0.73405489 0.82180131 0.87398531 0.89331272]
 At cv = 4
cross validation score  83.07885581754928
```

Support Vector Classifier

```
sv=SVC()
model_selection(sv,x_train, x_test, y_train, y_test)
model_cv(sv,trainx,trainy)
```

```
accuracy 80.26969481902059
[[956  93]
 [185 175]]
              precision    recall  f1-score   support

           0       0.84      0.91      0.87      1049
           1       0.65      0.49      0.56       360

    accuracy                           0.80      1409
   macro avg       0.75      0.70      0.72      1409
weighted avg       0.79      0.80      0.79      1409

[0.7309625  0.7325087  0.73714727 0.73637418]
 At cv = 4
cross validation score  73.42481638964051
[0.7236715  0.74057971 0.7410628  0.7341711  0.73368777]
 At cv = 5
```

DecisionTreeClassifier

```
# model dtc
dtc=DecisionTreeClassifier(random_state=26)
model_selection(dtc,x_train, x_test, y_train, y_test)
model_cv(dtc,trainx,trainy)
```

```
accuracy 75.51454932576294
[[875 174]
 [171 189]]
              precision    recall  f1-score   support

           0       0.84      0.83      0.84      1049
           1       0.52      0.53      0.52       360

    accuracy                           0.76      1409
   macro avg       0.68      0.68      0.68      1409
weighted avg       0.76      0.76      0.76      1409

[0.68805566 0.78237341 0.82102822 0.83223811]
 At cv = 4
cross validation score  78.09238500193274
[0.68985507 0.72077295 0.82995169 0.82455292 0.84678589]
 At cv = 5
```

LogisticRegression

```
# model dtc
lr=LogisticRegression(random_state=26)
model_selection(lr,x_train, x_test, y_train, y_test)
model_cv(lr,trainx,trainy)
```

```
accuracy 80.76650106458482
[[944 105]
 [166 194]]
              precision    recall  f1-score   support

           0       0.85      0.90      0.87      1049
           1       0.65      0.54      0.59       360

    accuracy                           0.81      1409
   macro avg       0.75      0.72      0.73      1409
weighted avg       0.80      0.81      0.80      1409

[0.72671048 0.79126401 0.82605334 0.83958253]
 At cv = 4
cross validation score  79.59025898724391
```

GradientBoostingClassifier

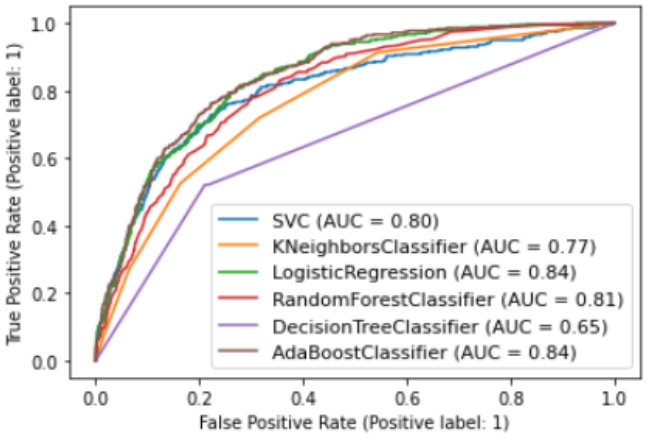
```
gbc=GradientBoostingClassifier()
model_selection(gbc,x_train, x_test, y_train, y_test)
model_cv(gbc,trainx,trainy)
```

```
accuracy 80.55358410220013
[[945 104]
 [170 190]]
              precision    recall  f1-score   support

           0       0.85      0.90      0.87      1049
           1       0.65      0.53      0.58       360

    accuracy                           0.81      1409
   macro avg       0.75      0.71      0.73      1409
weighted avg       0.80      0.81      0.80      1409

[0.73018941 0.80981832 0.85001933 0.86818709]
 At cv = 4
cross validation score  81.45535369153461
[0.73816425 0.7647343  0.85072464 0.86321895 0.8699855 ]
 At cv = 5
cross validation score  81.73655269996708
```

ROC_AUC_Curve:

AUC - ROC curve is a performance measurement for the classification problems at various threshold settings, Logistic Regression has ROC AUC highest score, hence choosing it is our best model.

```
disp = plot_roc_curve(sv,x_test,y_test)
plot_roc_curve(knn, x_test, y_test, ax=disp.ax_)      # ax_=Axes with confusion matrix
plot_roc_curve(lr, x_test,y_test, ax=disp.ax_)
plot_roc_curve(rf, x_test,y_test, ax=disp.ax_)
plot_roc_curve(dtc, x_test,y_test, ax=disp.ax_)
plot_roc_curve(ad, x_test,y_test, ax=disp.ax_)


plt.legend(prop={'size':11}, loc='lower right')
plt.show()
```



Compare Performance of All Model:

	Model	Model accuracy	Cross validation	Diff between cv and Acc
0	RandomForest classifier	77	83	-6
1	DecisionTree classifier	72	79	7
2	KNeighbour classifier	75	79	4
3	svm	79	73	5
4	LogisticRegression	80	80	0
5	AdaBoostClassifier	80	81	-1

Hyper Parameter Tuning for Best Mode

It is important to bear in mind that we have trained all the algorithms using the default hyperparameters. The accuracy of many machine learning algorithms is highly sensitive to the hyperparameters chosen for training the model. A more in-depth analysis will include an evaluation of a wider range of hyperparameters (not only default values) before choosing a model (or models) for hyperparameter tuning.

- Choosing best model as Logistic Regression" Model
- minimum difference between cross validation and Accuracy score is 0
- Roc_Auc score is maximum

```
#Hyper Parameter Tuning for Logistic Regression using grid search

param_grid={'penalty':['l1', 'l2','elasticnet','none'],
           'dual':[True,False],
           'tol':[0.75,1],
           'C':[2,3,4,7],
           'intercept_scaling':[1.2,2.1],
           'class_weight':['dict','balanced'],
           'random_state':[2,3],
          'solver':['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
           }
```

```
# Instantiate the grid search model
grid_search = GridSearchCV(estimator =LogisticRegression() , param_grid = param_grid,cv = 5)

# Fit the grid search to the data
grid_search.fit(x_train,y_train)
print(grid_search.best_params_)
```

```
{'C': 2, 'class_weight': 'dict', 'dual': False, 'intercept_scaling': 1.2, 'penalty': 'none', 'random_state': 2, 'solver': 'lbfgs', 'tol': 1}
```

Applying Best Parameters for our best model

The next step in the machine learning process is to perform hyperparameter tuning.
The selection of hyperparameters consists of testing the performance of the model against different combinations of hyperparameters, selecting those that perform best according to a chosen metric and a validation method.

There are multiple techniques to find the best hyperparameters for a model. The most popular methods are (1) grid search, (2) random search, and (3) Bayesian optimization. Grid search test all combinations of hyperparameters and select the best performing one. It is a really time-consuming method, particularly when the number of hyperparameters and values to try are really high. Now we are using grid search for hyperparameter tuning.

```
Final_Model=LogisticRegression(random_state=2,C=2,dual=False,intercept_scaling=1.2,penalty='none',tol=1,class_weight='dict',solver='lbfgs')
Final_Model.fit(x_train,y_train)
pred=Final_Model.predict(x_test)
print("Accuracy score of final model is=  ",accuracy_score(y_test,pred)*100)
```

```
Accuracy score of final model is=   80.69552874378992
```

```
print("classification_report for final Model",classification_report(y_test,pred))
```

```
classification_report for final Model              precision    recall  f1-score   support

           0       0.86      0.89      0.87      1049
           1       0.64      0.56      0.60       360

    accuracy                           0.81      1409
   macro avg       0.75      0.73      0.74      1409
weighted avg       0.80      0.81      0.80      1409
```

Confusion Matrix of final model:

The confusion matrix, also known as the error matrix, is used to evaluate the performance of a machine learning model by examining the number of observations that are correctly and incorrectly classified. Each column of the matrix contains the predicted classes while each row represents the actual classes or vice versa. In a perfect classification, the confusion matrix will be all zeros except for the diagonal. All the elements out of the main diagonal represent misclassifications. It is important to bear in mind that the confusion matrix allows us to observe patterns of misclassification (which classes and to which extend they were incorrectly classified).

In binary classification problems, the confusion matrix is a 2-by-2 matrix composed of 4 elements:

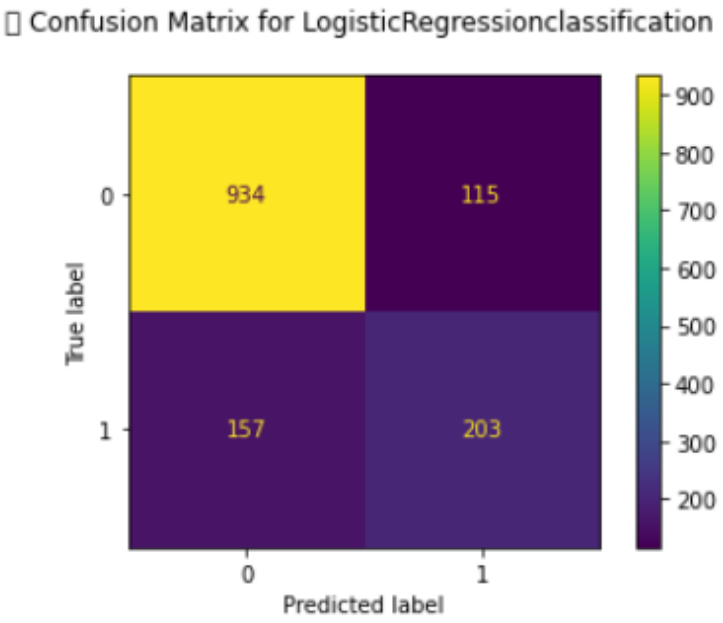
TP (True Positive): number of patients with spine problems that are correctly classified as sick.

TN (True Negative): number of patients without pathologies who are correctly classified as healthy.

FP (False Positive): number of healthy patients that are wrongly classified as sick.

FN (False Negative): number of patients with spine diseases that are misclassified as healthy.

```
class_names = df.columns
metrics.plot_confusion_matrix(Final_Model, x_test, y_test)
plt.title('\t Confusion Matrix for LogisticRegressionclassification \n')
plt.show()
```



Saving, Loading and Prediction of the best Classification ML model:

```
import pickle
filename_1="Customer_Churn.pkl"
pickle.dump(Final_Model,open(filename_1,"wb"))
```

```
loaded_model=pickle.load(open("Customer_Churn.pkl","rb"))
result=loaded_model.score(x_test,y_test)
print(result*100)
```

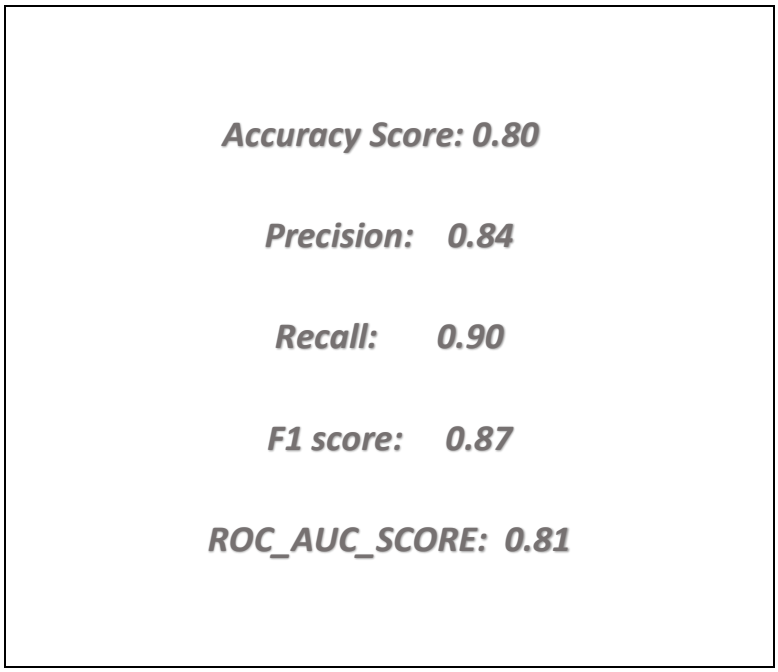
```
80.69552874378992
```

```
conclusion=pd.DataFrame([loaded_model.predict(x_test)[:],pred[:]],index=["predicted","original"])
```

```
conclusion.T
```

	predicted	original
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
...	...	...
1404	0	0
1405	0	0
1406	1	1

Our model is now ready to predict customer churn or not



Conclusion and Remarked

In this project, we have walked through a complete end-to-end machine learning project using the telecommunications customers dataset. We started by cleaning the data and analyzing it with visualization. Then, to be able to build a machine learning model, we transformed the categorical data into numeric variables. After transforming the data, we tried 7 different machine learning algorithms using default parameters. Finally, we tuned the hyperparameters of the **Logistic Regression** (best performance model) for model optimization, obtaining an **accuracy of nearly 80%** .

For more and clear steps please use the below links to access the codes and results:

https://github.com/raganeeverma/Blog/blob/main/Telecom_customer_churn-checkpoint.ipynb