In [28]:
```
from google.colab import files
uploaded=files.upload()
```

Choose Files No file selected Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Customer_data.csv to Customer_data (1).csv
```

In [256…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [257…
```
df=pd.read_csv('Customer_data.csv')
df
```

Out[257…

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No
...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes

7043 rows × 23 columns

In [258…
```
#head
df.head()
```

Out[258…

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

5 rows × 23 columns

In [259…
```
#information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 23 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      7032 non-null   float64
 20  Churn             7043 non-null   object 
 21  Unnamed: 21       0 non-null      float64
 22  Unnamed: 22       0 non-null      float64
dtypes: float64(4), int64(2), object(17)
memory usage: 1.2+ MB
```

In [260…

```
#null values
df.isnull().sum()
```

Out[260…

	0
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
Unnamed: 21	7043
Unnamed: 22	7043

dtype: int64

In [261…

```
#Insights
#1. Here our target variable is churn
#2. The data has almost no missing values only the column 'TotalCharges' contains some missing values
#3. There is two column which are completely empty and have to drop to clean the data
#4. Customer id is a unique identifier but it is of no use in data prediction so we will drop it
```

In [262…
```
#Data cleaning
#Drop the fully empty columns
df1=df.drop(['Unnamed: 21','Unnamed: 22','customerID'],axis=1)
df1
```

Out[262…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBack
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Y
1	Male	0	No	No	34	Yes	No	DSL	Yes	
2	Male	0	No	No	2	Yes	No	DSL	Yes	Y
3	Male	0	No	No	45	No	No phone service	DSL	Yes	
4	Female	0	No	No	2	Yes	No	Fiber optic	No	
...	...	...	...	...	...	...	...	...	...	
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	Y
7040	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	
7041	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	
7042	Male	0	No	No	66	Yes	No	Fiber optic	Yes	

7043 rows × 20 columns

In [263…
```
#valuecount
df1['Churn'].value_counts()
```

Out[263…

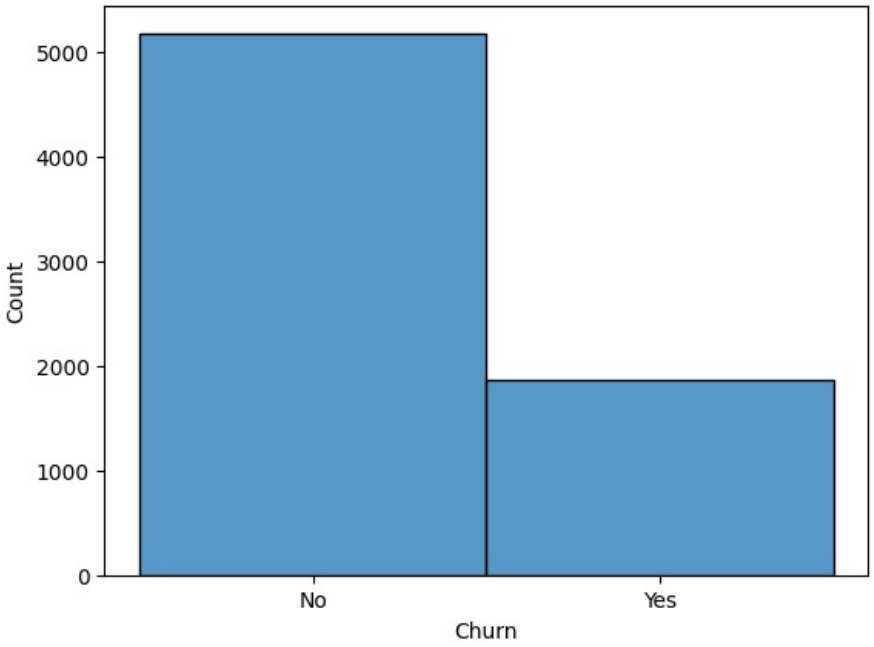
Churn	count
No	5174
Yes	1869

dtype: int64

In [265…
```
#Insights
#1.The majority of the customer did not churn
#2. This imbalance could lead to a model baised towards predicting'No Churn'
```

In [264…
```
#Histogram
sns.histplot(df1['Churn'])
```

Out[264… `<Axes: xlabel='Churn', ylabel='Count'>`



In [266…
```
#valuecount
df1['gender'].value_counts()
```

Out[266…

gender	count
Male	3555
Female	3488

dtype: int64

In [267…
```
#column having null value
df1.isnull().sum()[df1.isnull().sum()>0]
```

Out[267…

	0
TotalCharges	11

dtype: int64

In [268…
```
#Convert'Totalcharge' to numeric(error='coerce')
df1['TotalCharges']=pd.to_numeric(df1['TotalCharges'],errors='coerce')
```

In [269…
```
#recheck null values
df1.isnull().sum()
```

Out[269…

	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

In [270…
```
# drop rows where total charge is still nun
df1.dropna(subset=['TotalCharges'],inplace=True)
```

In [271…
```
#information
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7032 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7032 non-null   object 
 1   SeniorCitizen     7032 non-null   int64  
 2   Partner           7032 non-null   object 
 3   Dependents        7032 non-null   object 
 4   tenure            7032 non-null   int64  
 5   PhoneService      7032 non-null   object 
 6   MultipleLines     7032 non-null   object 
 7   InternetService   7032 non-null   object 
 8   OnlineSecurity    7032 non-null   object 
 9   OnlineBackup      7032 non-null   object 
 10  DeviceProtection  7032 non-null   object 
 11  TechSupport       7032 non-null   object 
 12  StreamingTV       7032 non-null   object 
 13  StreamingMovies   7032 non-null   object 
 14  Contract          7032 non-null   object 
 15  PaperlessBilling  7032 non-null   object 
 16  PaymentMethod     7032 non-null   object 
 17  MonthlyCharges    7032 non-null   float64
 18  TotalCharges      7032 non-null   float64
 19  Churn             7032 non-null   object 
dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
```

In [272…
```
#value count
df1['PaymentMethod'].value_counts()
```

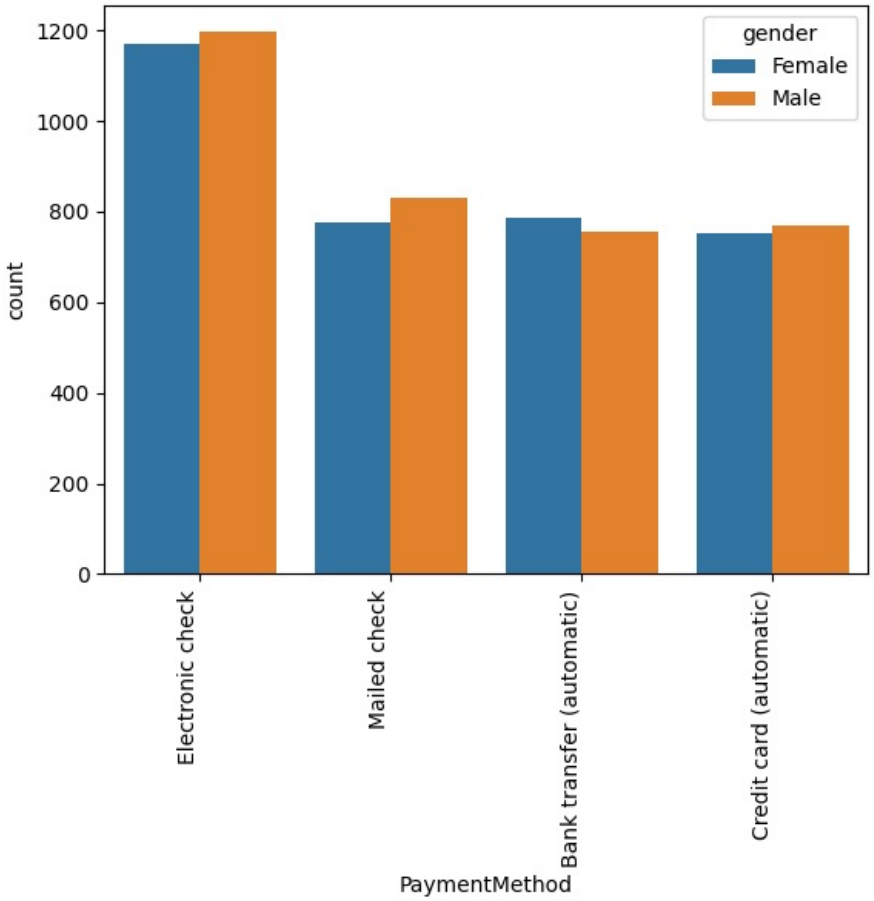
Out[272…

PaymentMethod	count
Electronic check	2365
Mailed check	1604
Bank transfer (automatic)	1542
Credit card (automatic)	1521

dtype: int64

In [273…
```
#Insights
#1.As we can see that most of the payment has been done by 'Electronic check'.
```

In [274…
```
#Histogram
sns.countplot(data=df1, x='PaymentMethod', hue='gender')
plt.xticks(rotation=90, horizontalalignment='right') #rotate and align
plt.show()
```



In [275…
```
#Insights
#1.Both male and female are most frequently use Electronic check as there payment method
```

In [276…
```
#for 'gender'
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['gender']=LE.fit_transform(df1['gender'])
df1['gender']
```

Out[276…

	gender
0	0
1	1
2	1
3	1
4	0
...	...
7038	1
7039	0
7040	0
7041	1
7042	1

7032 rows × 1 columns

dtype: int64

In [277…
```
#binary columns
binary_cols=[col for col in df1.columns if df1[col].dtype=='object' and df1[col].nunique()==2]
binary_cols
```

Out[277… ['Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'Churn']

In [278…
```
#Unique value in binary column
for col in binary_cols:
  print(f'{col}:{df1[col].unique()}')
```

```
Partner:['Yes' 'No']
Dependents:['No' 'Yes']
PhoneService:['No' 'Yes']
PaperlessBilling:['Yes' 'No']
Churn:['No' 'Yes']
```

In [279…
```
#we use label encode for binary Columns
from sklearn.preprocessing import LabelEncoder
for col in binary_cols:
  df1[col]=df1[col].map({'Yes':1,'No':0})
```

In [280…
```
#head
df1.head()
```

Out[280…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
0	0	0	1	0	1	0	No phone service	DSL	No	Yes
1	1	0	0	0	34	1	No	DSL	Yes	No
2	1	0	0	0	2	1	No	DSL	Yes	Yes
3	1	0	0	0	45	0	No phone service	DSL	Yes	No
4	0	0	0	0	2	1	No	Fiber optic	No	No

In [281…
```
#value_count
df1['gender'].value_counts()
```

Out[281…

	count
gender	
1	3549
0	3483

dtype: int64

In [282…
```
#Categorical columns
cat_cols=[col for col in df1.columns if df1[col].dtype=='object' and df1[col].nunique()>2]
```

```
cat_cols
```

Out[282…
```
['MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaymentMethod']
```

In [283…
```
#we will use one hot encoding for categorical columns
df1=pd.get_dummies(df1,columns=cat_cols,drop_first=True)
df1
```

Out[283…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	PaperlessBilling	MonthlyCharges	TotalCharges	Churn
0	0	0	1	0	1	0	1	29.85	29.85	0
1	1	0	0	0	34	1	0	56.95	1889.50	0
2	1	0	0	0	2	1	1	53.85	108.15	1
3	1	0	0	0	45	0	0	42.30	1840.75	0
4	0	0	0	0	2	1	1	70.70	151.65	1
...	...	...	...	...	...	...	...	...	...	...
7038	1	0	1	1	24	1	1	84.80	1990.50	0
7039	0	0	1	1	72	1	1	103.20	7362.90	0
7040	0	0	1	1	11	0	1	29.60	346.45	0
7041	1	1	1	0	4	1	1	74.40	306.60	1
7042	1	0	0	0	66	1	1	105.65	6844.50	0

7032 rows × 31 columns

In [284…
```
#head
df1.head()
```

Out[284…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	PaperlessBilling	MonthlyCharges	TotalCharges	Churn	...
0	0	0	1	0	1	0	1	29.85	29.85	0	...
1	1	0	0	0	34	1	0	56.95	1889.50	0	...
2	1	0	0	0	2	1	1	53.85	108.15	1	...
3	1	0	0	0	45	0	0	42.30	1840.75	0	...
4	0	0	0	0	2	1	1	70.70	151.65	1	...

5 rows × 31 columns

In [285…
```
#Fix all boolean columns columns to integer
for col in df1.columns:
  if df1[col].dtype=='bool':
    df1[col]=df1[col].astype(int)
```

In [287…
```
#head
df1.head()
```

Out[287…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	PaperlessBilling	MonthlyCharges	TotalCharges	Churn	...
0	0	0	1	0	1	0	1	29.85	29.85	0	...
1	1	0	0	0	34	1	0	56.95	1889.50	0	...
2	1	0	0	0	2	1	1	53.85	108.15	1	...
3	1	0	0	0	45	0	0	42.30	1840.75	0	...
4	0	0	0	0	2	1	1	70.70	151.65	1	...

5 rows × 31 columns

In [288…
```
#Correlation
corr_matrix=df1.corr()
churn_corr=corr_matrix['Churn'].sort_values(ascending=False)
print(churn_corr)
```

```
Churn                                    1.000000
InternetService_Fiber optic              0.307463
PaymentMethod_Electronic check           0.301455
MonthlyCharges                           0.192858
PaperlessBilling                         0.191454
SeniorCitizen                            0.150541
StreamingTV_Yes                          0.063254
StreamingMovies_Yes                      0.060860
MultipleLines_Yes                        0.040033
PhoneService                             0.011691
gender                                  -0.008545
MultipleLines_No phone service          -0.011691
DeviceProtection_Yes                    -0.066193
OnlineBackup_Yes                        -0.082307
PaymentMethod_Mailed check              -0.090773
PaymentMethod_Credit card (automatic)   -0.134687
Partner                                 -0.149982
Dependents                              -0.163128
TechSupport_Yes                         -0.164716
OnlineSecurity_Yes                      -0.171270
Contract_One year                       -0.178225
TotalCharges                            -0.199484
InternetService_No                      -0.227578
StreamingTV_No internet service         -0.227578
OnlineSecurity_No internet service      -0.227578
OnlineBackup_No internet service        -0.227578
DeviceProtection_No internet service    -0.227578
StreamingMovies_No internet service     -0.227578
TechSupport_No internet service         -0.227578
Contract_Two year                       -0.301552
tenure                                  -0.354049
Name: Churn, dtype: float64
```

In [289…
```
#Insights
#1.Tenure is negatively correlated, means long term customer are less likely to churn
#2.month to month contracts,no online security and fiber optic service are highly associated with churn
```

In [291…
```
#train_test
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.utils.class_weight import compute_class_weight
```

In [292…
```
#Separate feature and target variables
X=df1.drop('Churn',axis=1)
y=df1['Churn']
```

In [293…
```
#Stratified train_test split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42,stratify=y)
```

In [295…
```
#Compute the class for imbalance handling
class_weights=compute_class_weight(class_weight='balanced',classes=np.array([0,1]),y=y_train)
class_weights_dict={0:class_weights[0],1:class_weights[1]}
```

In [296…
```
#Train random forest with class weight
rf=RandomForestClassifier(n_estimators=100,random_state=42,class_weight=class_weights_dict)
rf.fit(X_train,y_train)
```

Out[296…
RandomForestClassifier

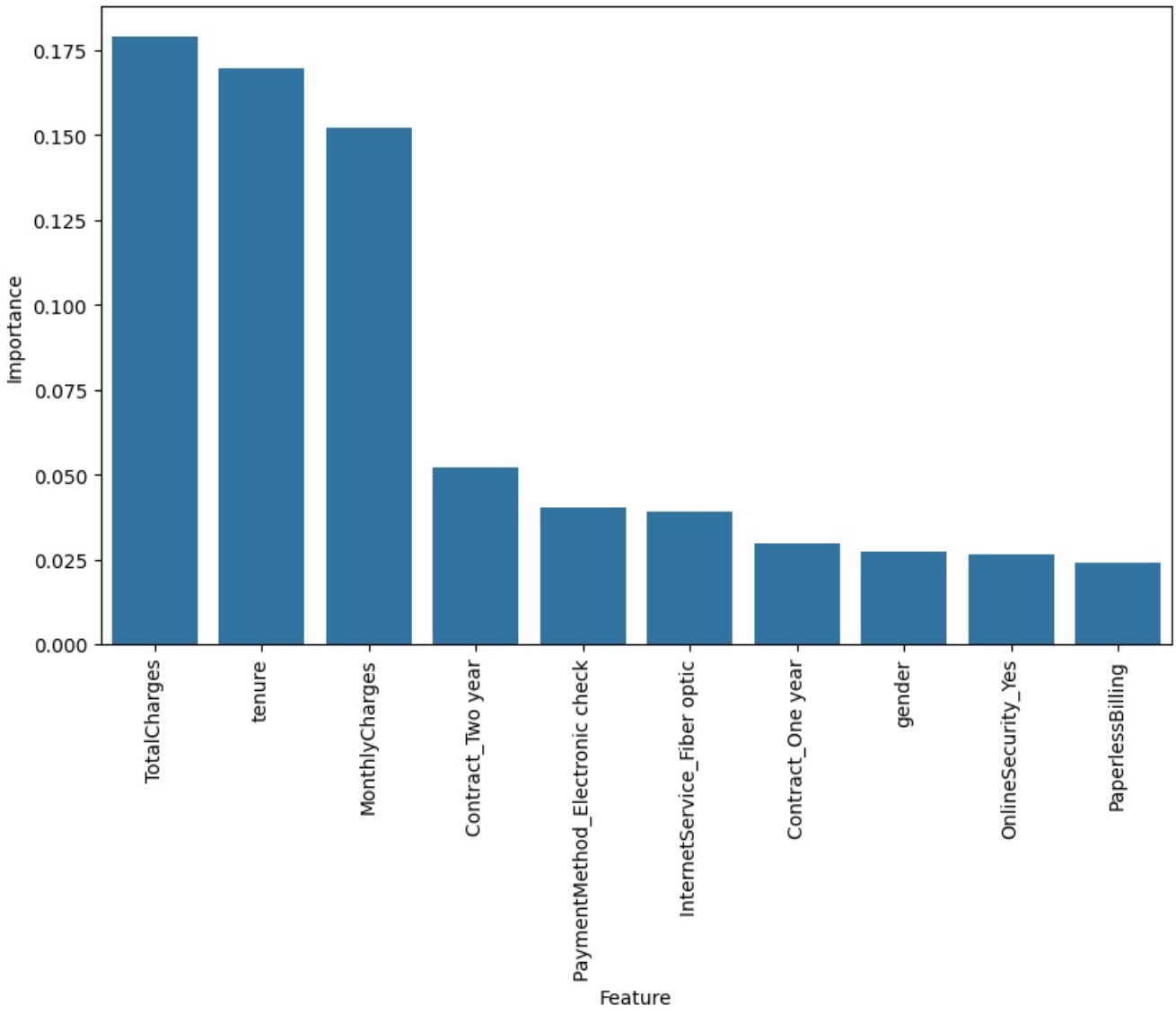
```
RandomForestClassifier(class_weight={0: np.float64(0.6809927360774818),
                                     1: np.float64(1.8812709030100334)},
                       random_state=42)
```

In [297…
```
#Predict and evaluate
y_pred=rf.predict(X_test)
conf_matrix=confusion_matrix(y_test,y_pred)
class_report=classification_report(y_test,y_pred)
print(conf_matrix)
print(class_report)
```

```
[[927 106]
 [191 183]]
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      1033
           1       0.63      0.49      0.55       374

    accuracy                           0.79      1407
   macro avg       0.73      0.69      0.71      1407
weighted avg       0.78      0.79      0.78      1407
```

In [298…
```
#Insights
#1.The model performing well on "No Churn" but struggles more with "Yes Churn"
#2. It still identifies 49% of churn cases, which is a good base line
```

In [301…
```
#Get feature importance from the random forest model
importance= rf.feature_importances_
features= X.columns
feature_importance_df1=pd.DataFrame({'Feature':features,'Importance':importance})
feature_importance_df1=feature_importance_df1.sort_values(by='Importance',ascending=False)

#plot top 10 features
plt.figure(figsize=(10,6))
sns.barplot(x='Feature',y='Importance',data=feature_importance_df1.head(10))
plt.xticks(rotation=90)
plt.show()
```



In [302…
```
#Insights
#1.Higher total spending is strongly related to churn
#2. shorter tenure customer are more likely to churn
#3.Higher monthly bills may be dissatisfication
#4.Customer with longer contracts are less likely to churn
```

In [ ]:

In [ ]: