# Customer churn Prediction using ML

June 2, 2025

[3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
import pickle
```

[4]:
```
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

[6]:
```
df.shape
```

[6]:
```
(7043, 21)
```

[7]:
```
df.head()
```

[7]:
```
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes
3  7795-CFOCW    Male              0      No         No      45           No
4  9237-HQITU  Female              0      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity  … DeviceProtection  \
0  No phone service             DSL             No  …               No
1                No             DSL            Yes  …              Yes
2                No             DSL            Yes  …               No
3  No phone service             DSL            Yes  …              Yes
4                No     Fiber optic             No  …               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
```

```
1          No          No              No        One year              No
2          No          No              No  Month-to-month             Yes
3         Yes          No              No        One year              No
4          No          No              No  Month-to-month             Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

[8]:

```
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
```

[9]:

```
# dropping customerID column as this is not required for modelling
df = df.drop(columns=["customerID"])
```

[10]:

```
df.head(2)
```

[10]:

```
   gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  Female              0     Yes         No       1           No
1    Male              0      No         No      34          Yes

      MultipleLines InternetService OnlineSecurity OnlineBackup  \
0  No phone service             DSL             No          Yes
1                No             DSL            Yes           No

  DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0               No          No          No              No  Month-to-month
1              Yes          No          No              No        One year

  PaperlessBilling     PaymentMethod  MonthlyCharges TotalCharges Churn
0              Yes  Electronic check           29.85        29.85    No
1               No      Mailed check           56.95       1889.5    No
```

[11]:

```
df.columns
```

[11]:

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

[12]:

```
print(df["gender"].unique())
```

```
['Female' 'Male']
```

[13]:

```
print(df["SeniorCitizen"].unique())
```

```
[0 1]
```

[14]:

```
# printing the unique values in all the columns

numerical_features_list = ["tenure", "MonthlyCharges", "TotalCharges"]

for col in df.columns:
  if col not in numerical_features_list:
    print(col, df[col].unique())
    print("-"*50)
```

```
gender ['Female' 'Male']
--------------------------------------------------
SeniorCitizen [0 1]
--------------------------------------------------
Partner ['Yes' 'No']
```

```
--------------------------------------------------
Dependents ['No' 'Yes']
--------------------------------------------------
PhoneService ['No' 'Yes']
--------------------------------------------------
MultipleLines ['No phone service' 'No' 'Yes']
--------------------------------------------------
InternetService ['DSL' 'Fiber optic' 'No']
--------------------------------------------------
OnlineSecurity ['No' 'Yes' 'No internet service']
--------------------------------------------------
OnlineBackup ['Yes' 'No' 'No internet service']
--------------------------------------------------
DeviceProtection ['No' 'Yes' 'No internet service']
--------------------------------------------------
TechSupport ['No' 'Yes' 'No internet service']
--------------------------------------------------
StreamingTV ['No' 'Yes' 'No internet service']
--------------------------------------------------
StreamingMovies ['No' 'Yes' 'No internet service']
--------------------------------------------------
Contract ['Month-to-month' 'One year' 'Two year']
--------------------------------------------------
PaperlessBilling ['Yes' 'No']
--------------------------------------------------
PaymentMethod ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
--------------------------------------------------
Churn ['No' 'Yes']
--------------------------------------------------
```

[15]:

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

```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
```

```
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

[16]:
```
#df["TotalCharges"] = df["TotalCharges"].astype(float)
df[df["TotalCharges"]==" "]
```

[16]:
```
      gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
488   Female              0     Yes        Yes       0           No
753     Male              0      No        Yes       0          Yes
936   Female              0     Yes        Yes       0          Yes
1082    Male              0     Yes        Yes       0          Yes
1340  Female              0     Yes        Yes       0           No
3331    Male              0     Yes        Yes       0          Yes
3826    Male              0     Yes        Yes       0          Yes
4380  Female              0     Yes        Yes       0          Yes
5218    Male              0     Yes        Yes       0          Yes
6670  Female              0     Yes        Yes       0          Yes
6754    Male              0      No        Yes       0          Yes

         MultipleLines InternetService       OnlineSecurity  \
488   No phone service             DSL                  Yes
753                 No              No  No internet service
936                 No             DSL                  Yes
1082               Yes              No  No internet service
1340  No phone service             DSL                  Yes
3331                No              No  No internet service
3826               Yes              No  No internet service
4380                No              No  No internet service
5218                No              No  No internet service
6670               Yes             DSL                   No
6754               Yes             DSL                  Yes

             OnlineBackup     DeviceProtection          TechSupport  \
488                    No                  Yes                  Yes
753   No internet service  No internet service  No internet service
936                   Yes                  Yes                   No
1082  No internet service  No internet service  No internet service
1340                  Yes                  Yes                  Yes
3331  No internet service  No internet service  No internet service
3826  No internet service  No internet service  No internet service
4380  No internet service  No internet service  No internet service
5218  No internet service  No internet service  No internet service
6670                  Yes                  Yes                  Yes
```

```
6754                  Yes                   No                  Yes

              StreamingTV      StreamingMovies  Contract PaperlessBilling  \
488                   Yes                   No  Two year              Yes
753   No internet service  No internet service  Two year               No
936                   Yes                  Yes  Two year               No
1082  No internet service  No internet service  Two year               No
1340                  Yes                   No  Two year               No
3331  No internet service  No internet service  Two year               No
3826  No internet service  No internet service  Two year               No
4380  No internet service  No internet service  Two year               No
5218  No internet service  No internet service  One year              Yes
6670                  Yes                   No  Two year               No
6754                   No                   No  Two year              Yes

                  PaymentMethod  MonthlyCharges TotalCharges Churn
488   Bank transfer (automatic)           52.55                 No
753                Mailed check           20.25                 No
936                Mailed check           80.85                 No
1082               Mailed check           25.75                 No
1340    Credit card (automatic)           56.05                 No
3331               Mailed check           19.85                 No
3826               Mailed check           25.35                 No
4380               Mailed check           20.00                 No
5218               Mailed check           19.70                 No
6670               Mailed check           73.35                 No
6754  Bank transfer (automatic)           61.90                 No
```

[17]:
```
len(df[df["TotalCharges"]==" "])
```

[17]:
```
11
```

[19]:
```
df["TotalCharges"] = df["TotalCharges"].replace({" ": "0.0"})
```

[20]:
```
df["TotalCharges"] = df["TotalCharges"].astype(float)
```

[21]:
```
df.info()
```

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

[22]:

```
# checking the class distribution of target column
print(df["Churn"].value_counts())
```

```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
```

## 1 Insights:

# 1)Customer Id removed as it in not requied for modelling. # 2)No missing values in the dataset. # 3)Missing values in the TotalCharges column were replaced with 0. # 4)Class imbalance identified in the target .

## 2 Exploratory Data Analysis (EDA)

[23]:

```
df.shape
```

[23]:

```
(7043, 20)
```

[24]:

```
df.columns
```

[24]:

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

[25]:

```
df.describe()
```

[25]:

```
       SeniorCitizen       tenure  MonthlyCharges  TotalCharges
count    7043.000000  7043.000000     7043.000000   7043.000000
mean        0.162147    32.371149       64.761692   2279.734304
std         0.368612    24.559481       30.090047   2266.794470
min         0.000000     0.000000       18.250000      0.000000
25%         0.000000     9.000000       35.500000    398.550000
50%         0.000000    29.000000       70.350000   1394.550000
75%         0.000000    55.000000       89.850000   3786.600000
max         1.000000    72.000000      118.750000   8684.800000
```

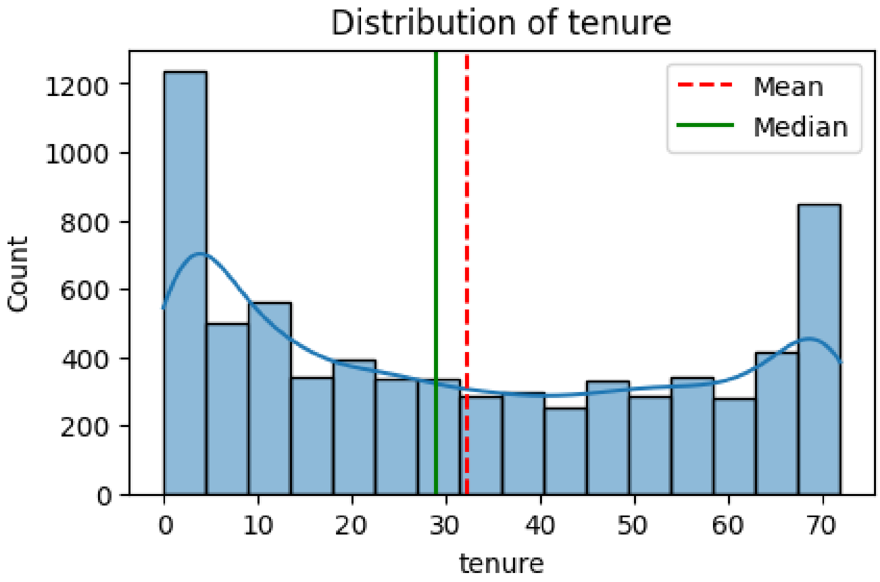
# 3 Numerical Feature - Analysis

[26]:

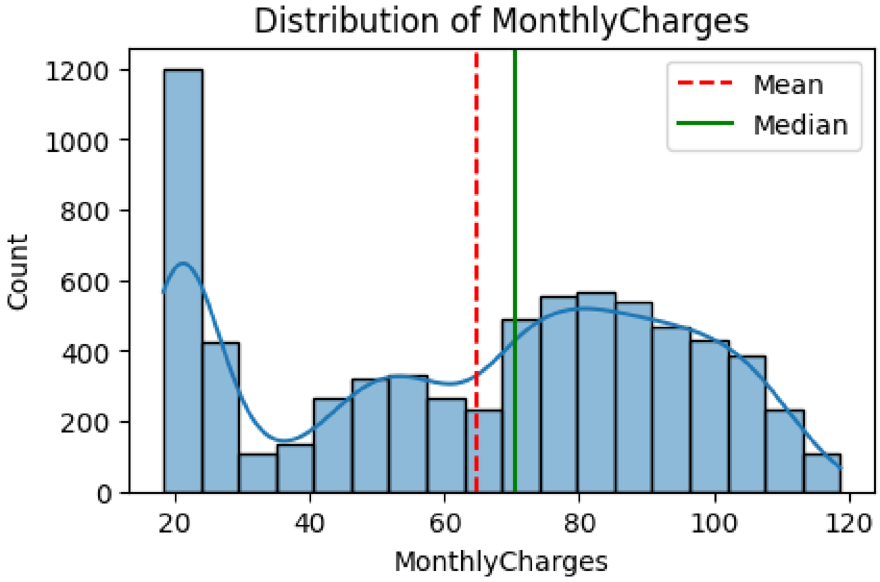
```
def plot_histogram(df, column_name):

    plt.figure(figsize=(5, 3))
    sns.histplot(df[column_name], kde=True)
    plt.title(f"Distribution of {column_name}")

    # calculate the mean and median values for the columns
    col_mean = df[column_name].mean()
    col_median = df[column_name].median()

    # add vertical lines for mean and median
    plt.axvline(col_mean, color="red", linestyle="--", label="Mean")
    plt.axvline(col_median, color="green", linestyle="-", label="Median")

    plt.legend()

    plt.show()
```

[27]:

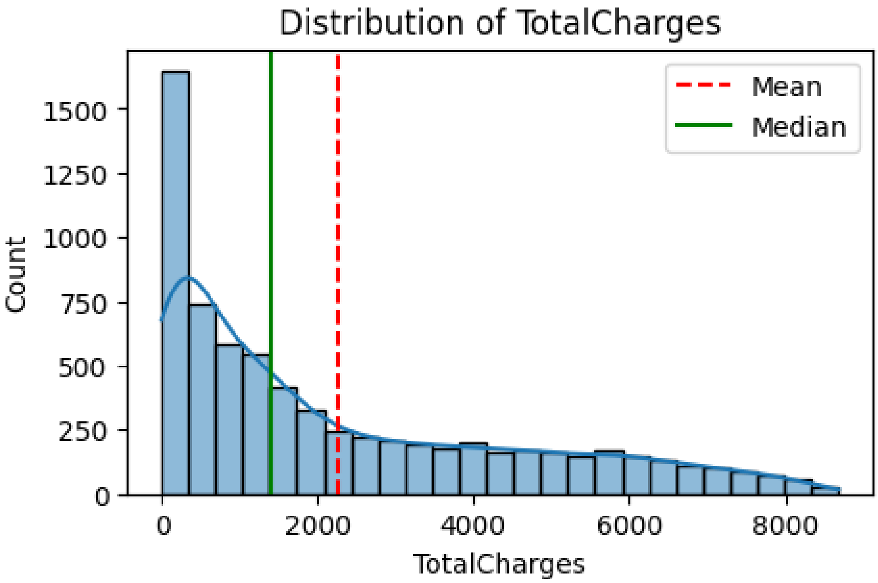
```
plot_histogram(df, "tenure")
```



[28]:
```
plot_histogram(df, "MonthlyCharges")
```



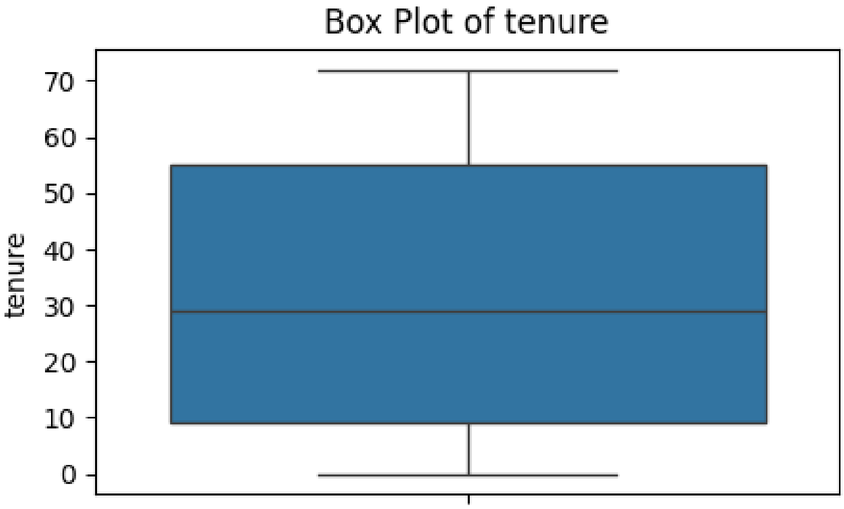
[29]:
```
plot_histogram(df, "TotalCharges")
```



## 4 Box plot for numerical feature

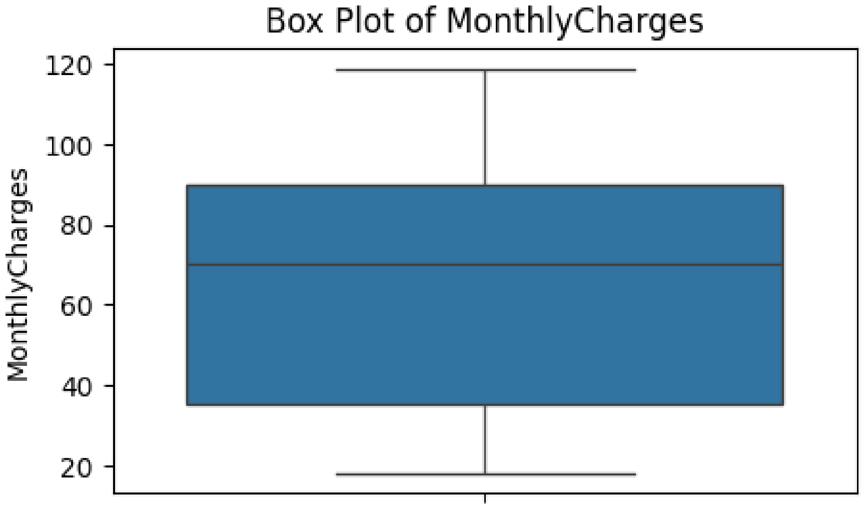
[30]:
```
def plot_boxplot(df, column_name):

  plt.figure(figsize=(5, 3))
  sns.boxplot(y=df[column_name])
  plt.title(f"Box Plot of {column_name}")
  plt.ylabel(column_name)
  plt.show
```

[31]:
```
plot_boxplot(df, "tenure")
```



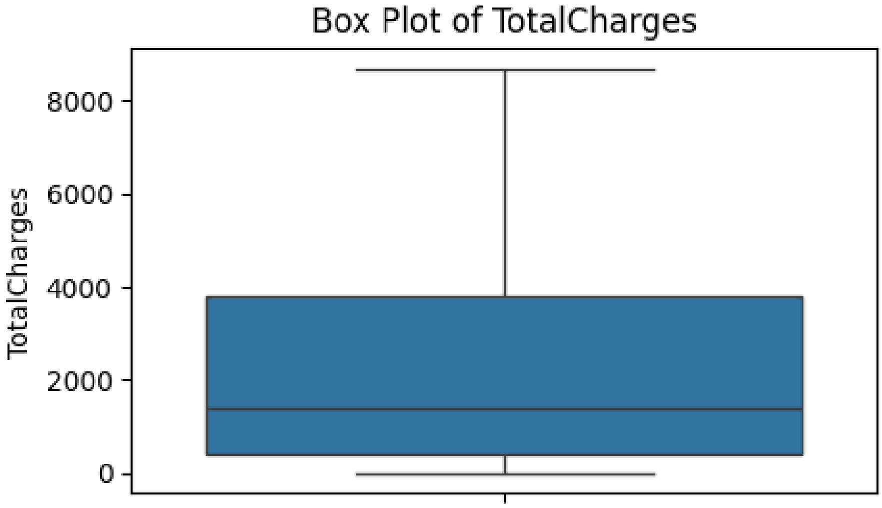
[33]:

```
plot_boxplot(df, "MonthlyCharges")
```



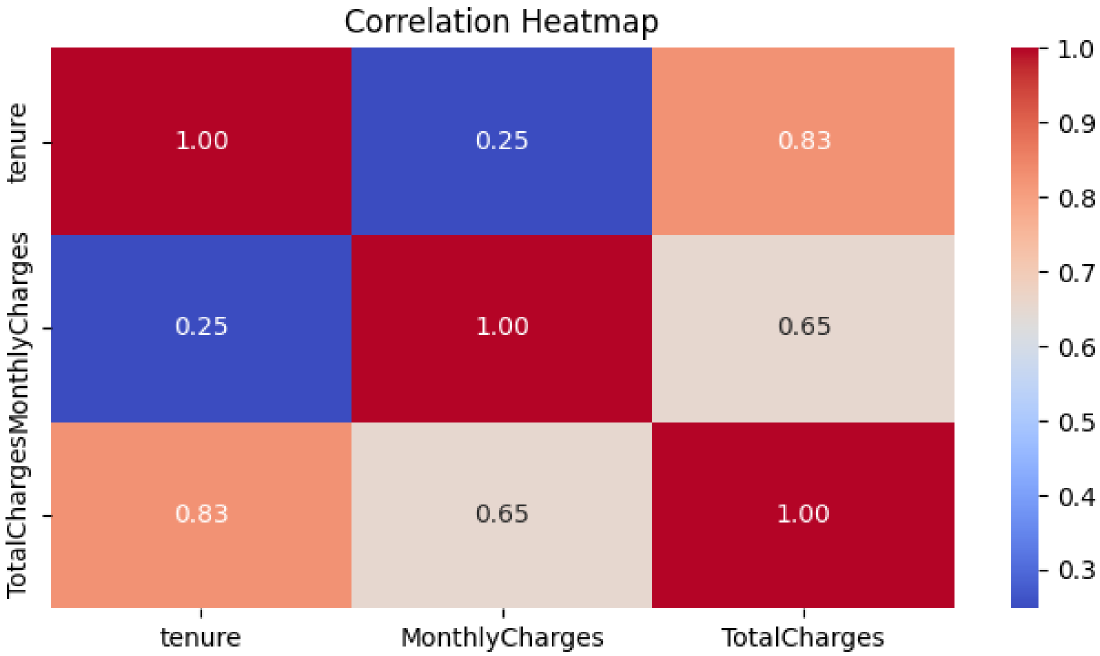
[34]:

```
plot_boxplot(df, "TotalCharges")
```



## 5 Correlation Heatmap for numerical columns

[35]:
```
# correlation matrix - heatmap
plt.figure(figsize=(8, 4))
sns.heatmap(df[["tenure", "MonthlyCharges", "TotalCharges"]].corr(),␣
  ↪annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



## 6 Categorical feature - Analysis

[37]:
```
df.columns
```

[37]:
```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

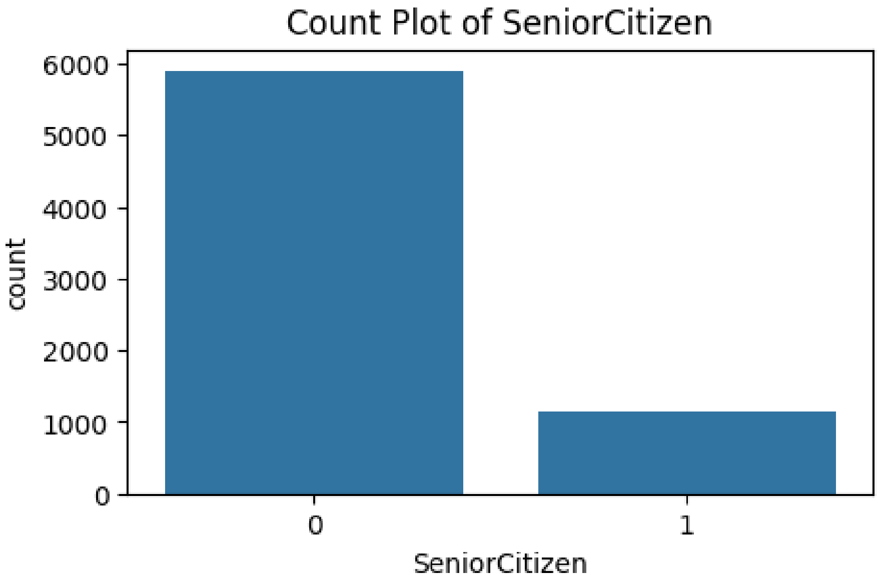
[38]:
```
df.info()
```

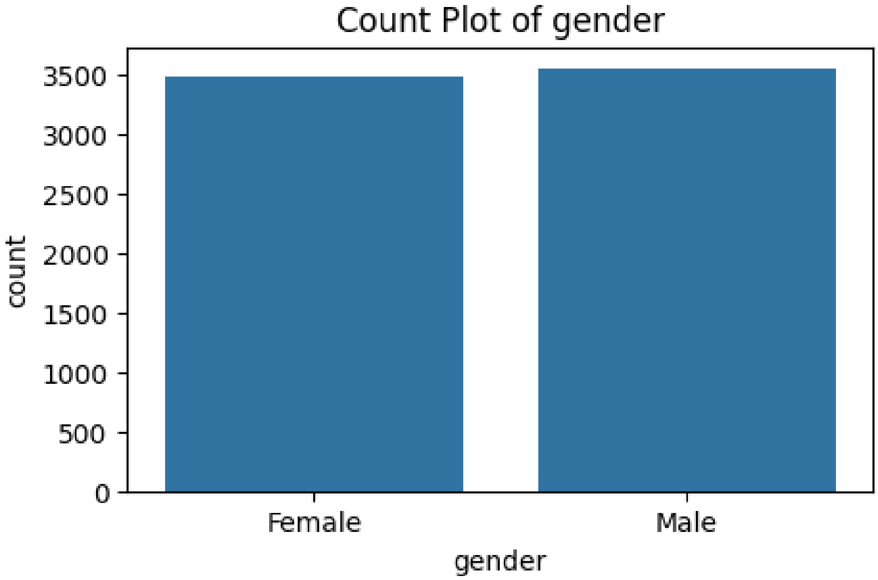
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7043 non-null   object
 1   SeniorCitizen     7043 non-null   int64
 2   Partner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

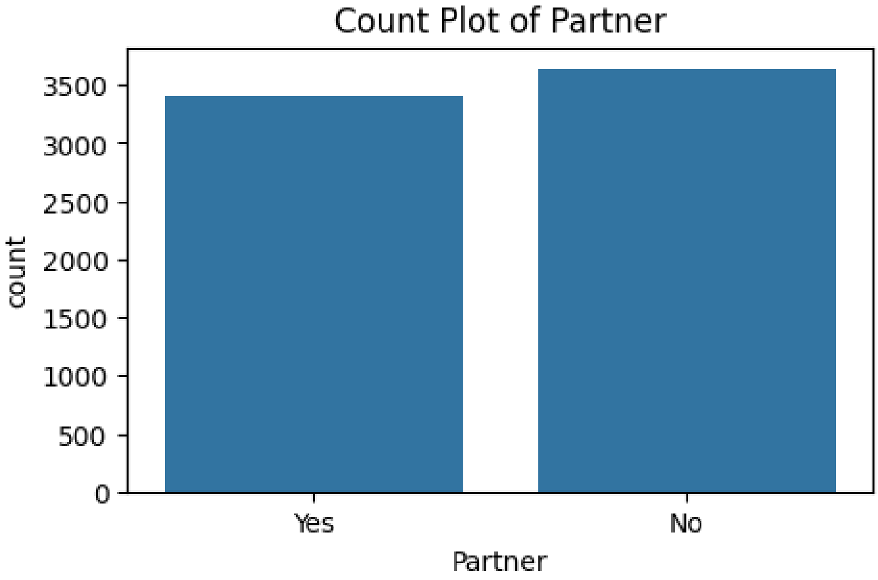
## 7 Count plot categorical columns

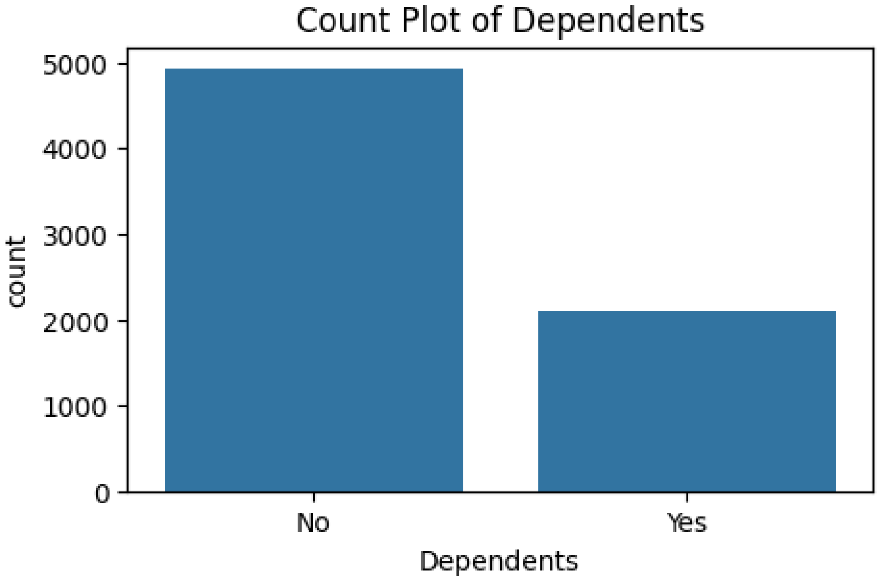
[39]:

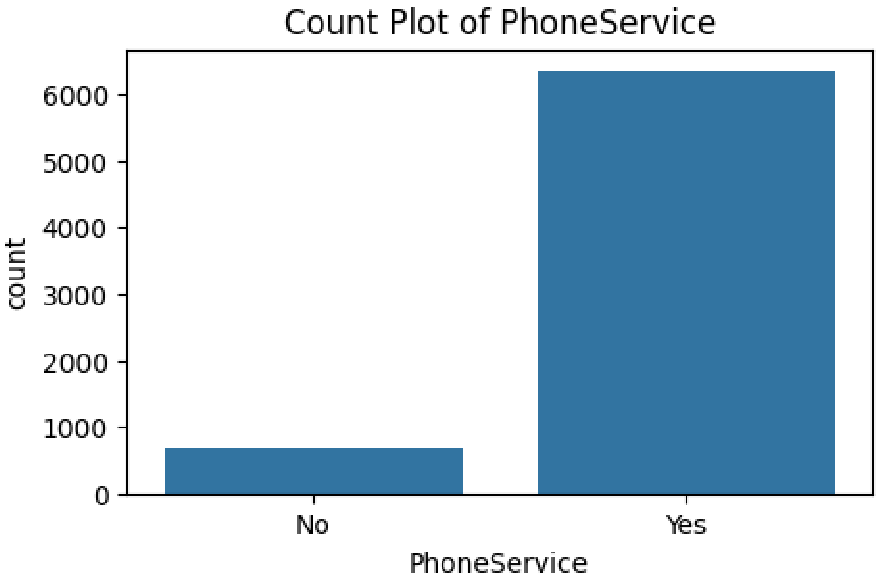
```python
object_cols = df.select_dtypes(include="object").columns.to_list()

object_cols = ["SeniorCitizen"] + object_cols

for col in object_cols:
  plt.figure(figsize=(5, 3))
  sns.countplot(x=df[col])
  plt.title(f"Count Plot of {col}")
  plt.show()
```

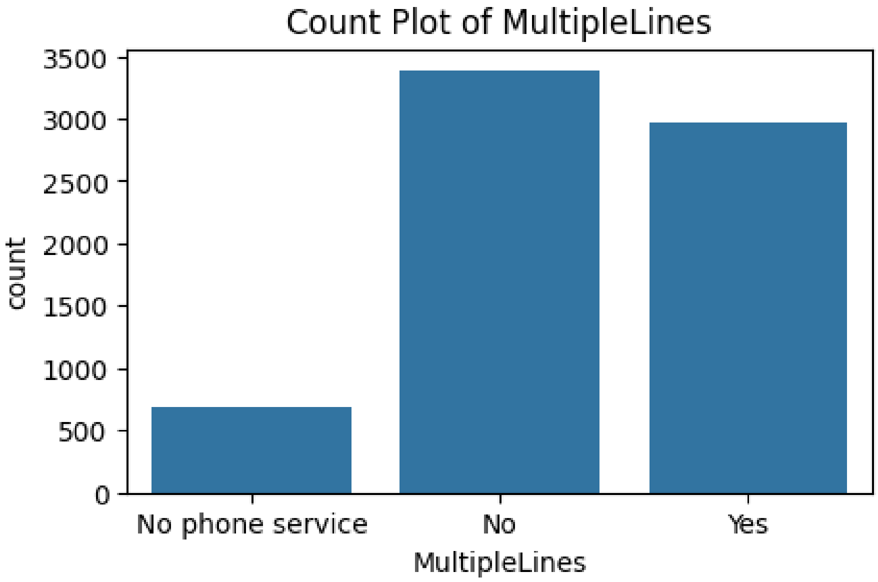

Count Plot of SeniorCitizen
count
6000
5000
4000
3000
2000
1000
0
0
1
SeniorCitizen

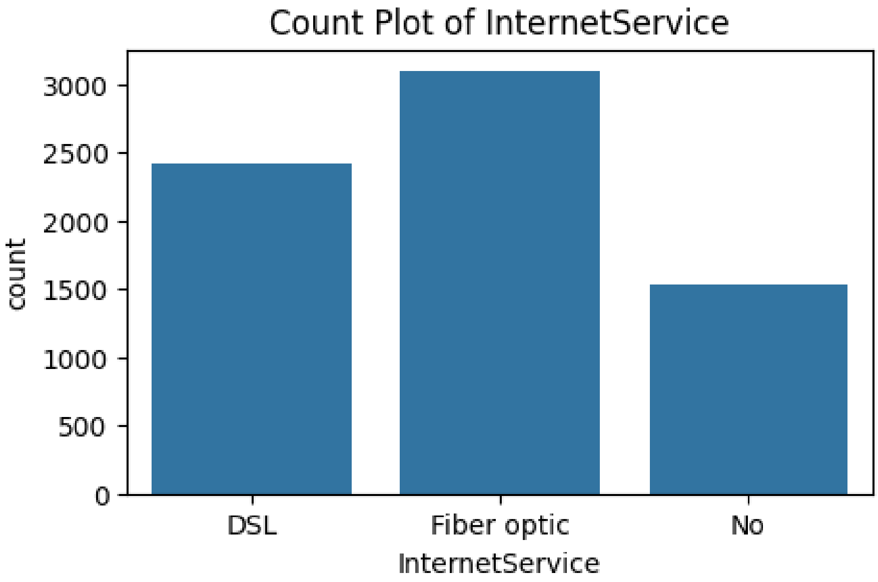

Count Plot of gender
count
3500
3000
2500
2000
1500
1000
500
0
Female
Male
gender

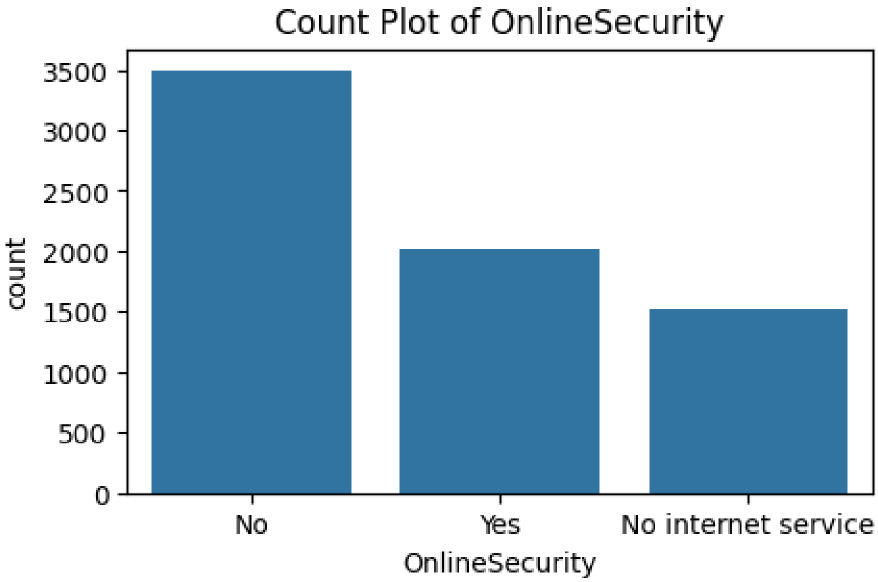

Count Plot of Partner
count
0
500
1000
1500
2000
2500
3000
3500
Yes
No
Partner

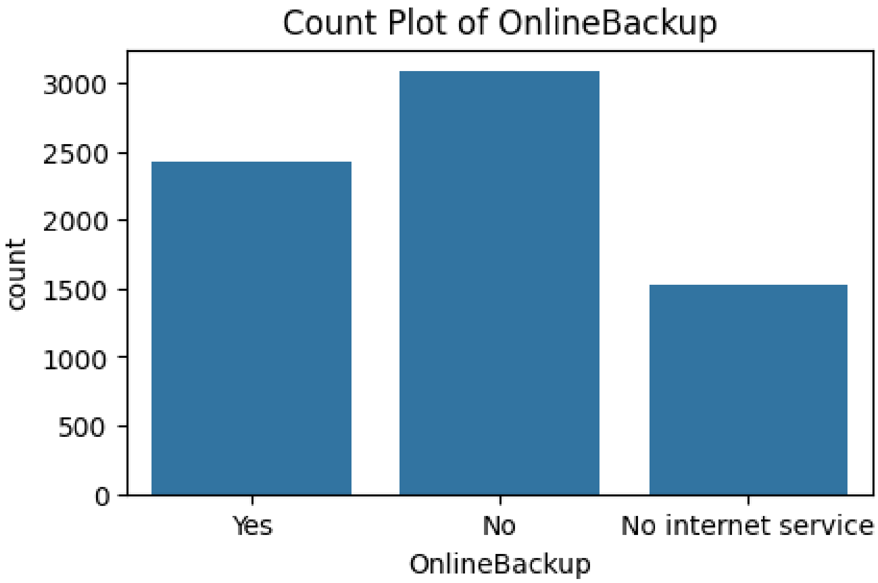

Count Plot of Dependents
count
0
1000
2000
3000
4000
5000
No
Yes
Dependents

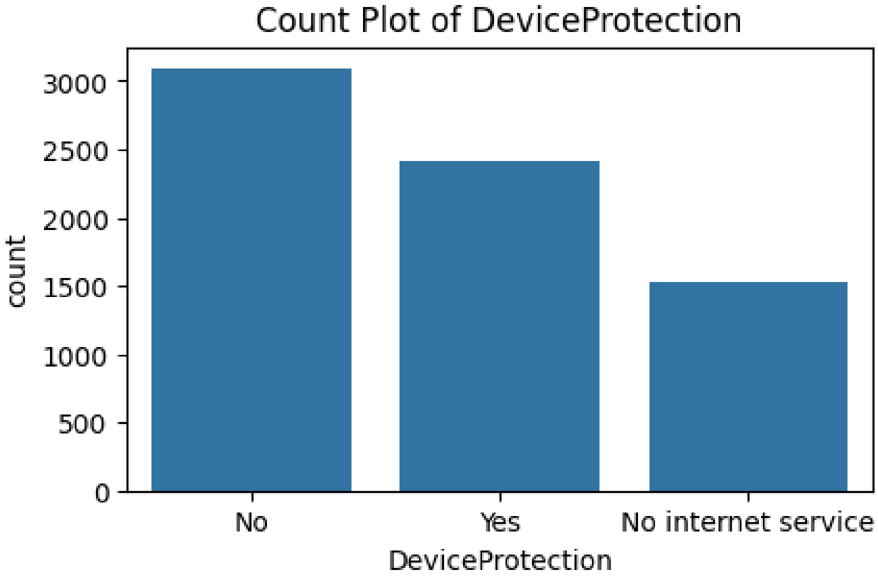

Count Plot of PhoneService
6000
5000
4000
3000
2000
1000
0
count
No
Yes
PhoneService

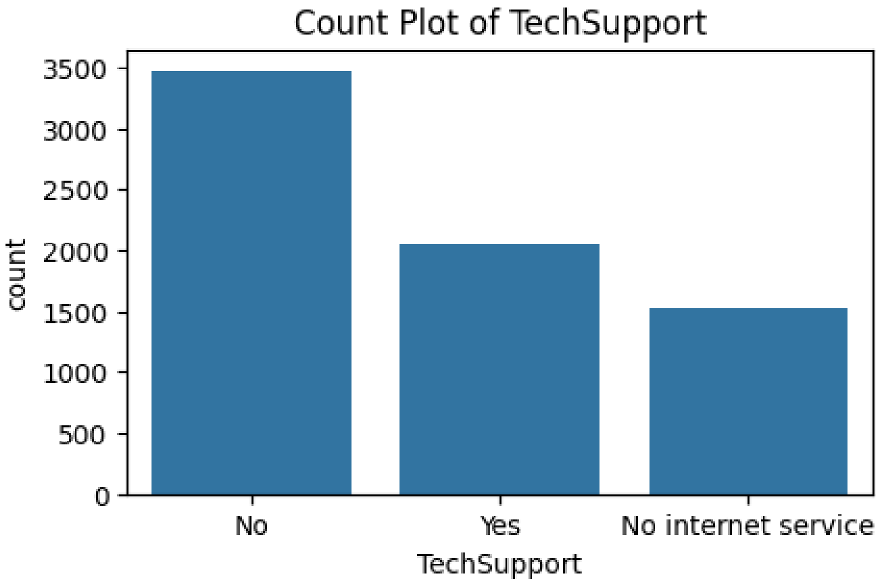

Count Plot of MultipleLines
3500
3000
2500
2000
1500
1000
500
0
count
No phone service
No
Yes
MultipleLines

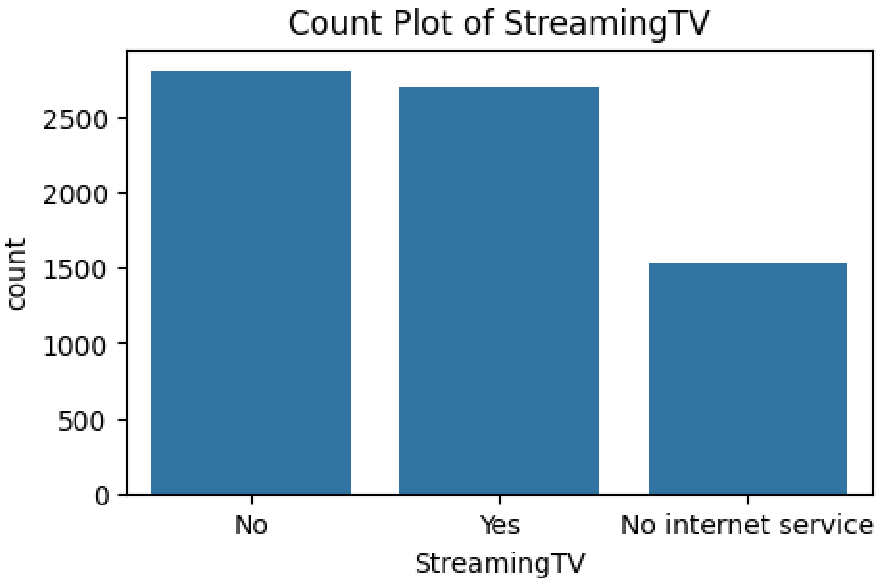

Count Plot of InternetService
count
3000
2500
2000
1500
1000
500
0
DSL
Fiber optic
No
InternetService

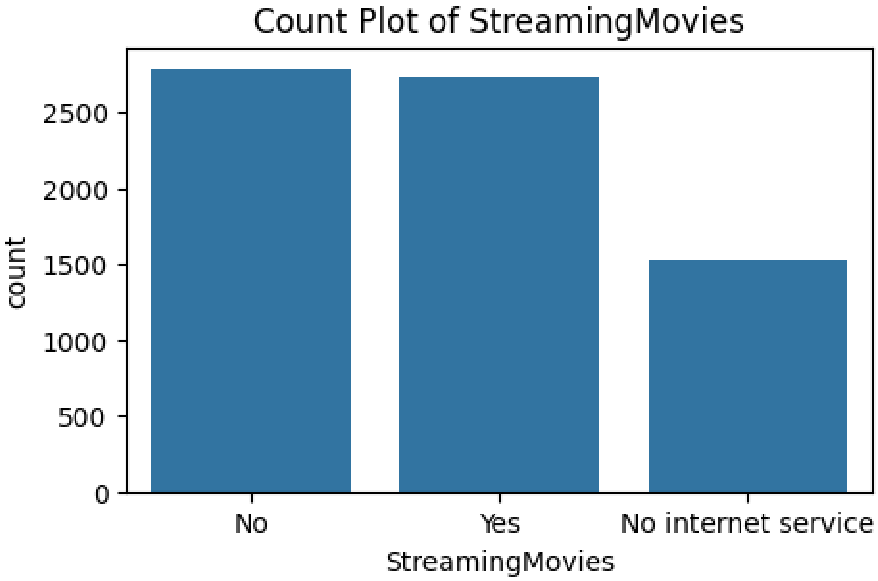

Count Plot of OnlineSecurity
count
3500
3000
2500
2000
1500
1000
500
0
No
Yes
No internet service
OnlineSecurity

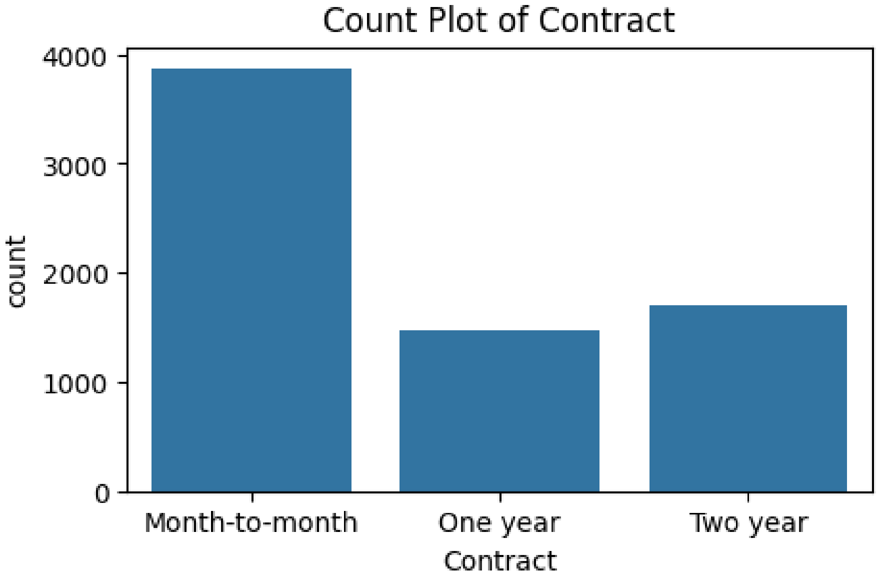

Count Plot of OnlineBackup
count
3000
2500
2000
1500
1000
500
0
Yes
No
No internet service
OnlineBackup

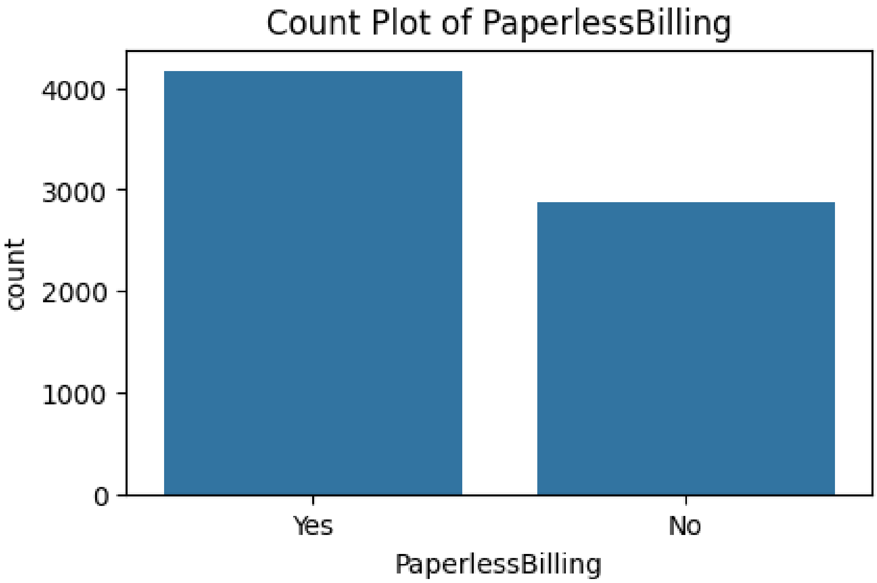

Count Plot of DeviceProtection
count
3000
2500
2000
1500
1000
500
0
No
Yes
No internet service
DeviceProtection

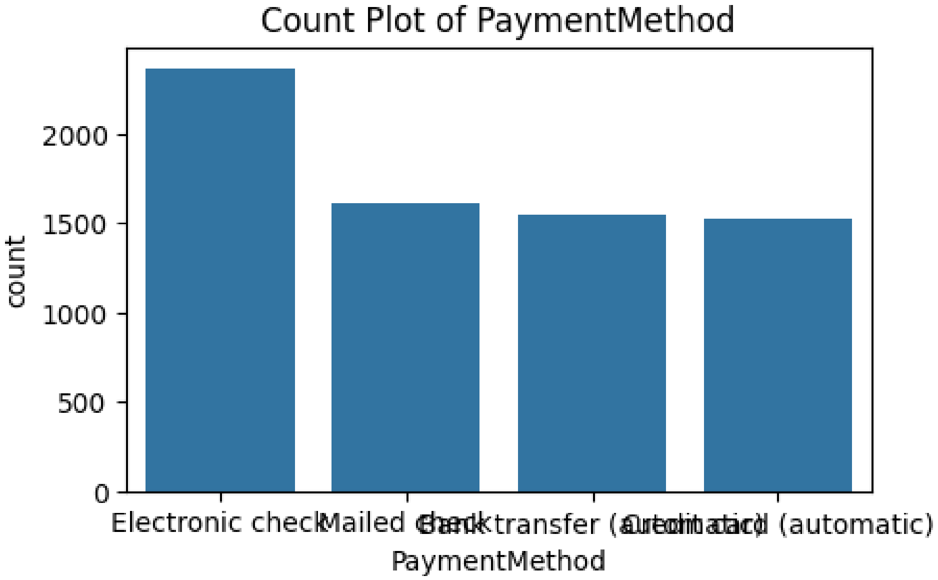

Count Plot of TechSupport
count
3500
3000
2500
2000
1500
1000
500
0
No
Yes
No internet service
TechSupport

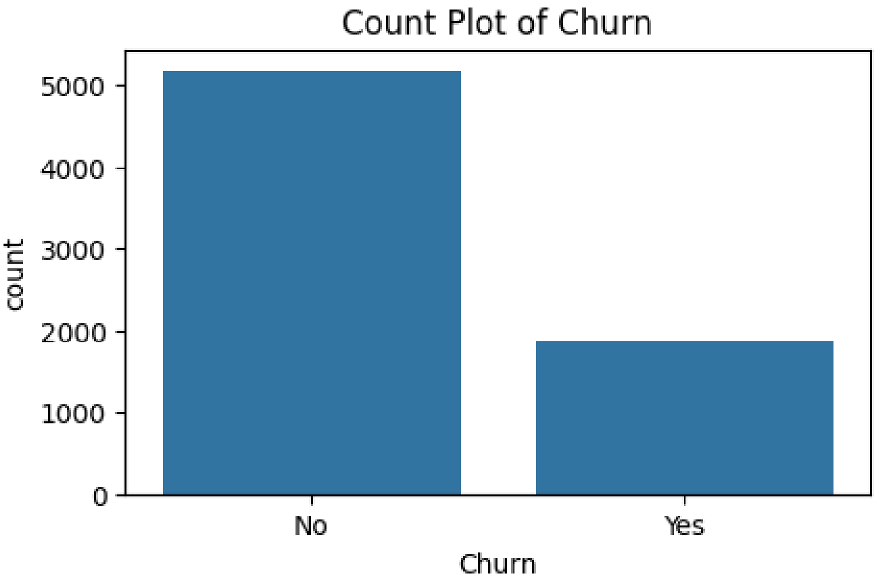

Count Plot of StreamingTV
count
2500
2000
1500
1000
500
0
No
Yes
No internet service
StreamingTV


Count Plot of StreamingMovies
count
0
500
1000
1500
2000
2500
No
Yes
No internet service
StreamingMovies


Count Plot of Contract
count
0
1000
2000
3000
4000
Month-to-month
One year
Two year
Contract


Count Plot of PaperlessBilling
count
4000
3000
2000
1000
0
Yes
No
PaperlessBilling


Count Plot of PaymentMethod
count
2000
1500
1000
500
0
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod



# 8 Data Processing

```
[40]: df.head(3)
```

```
[40]:    gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
      0  Female              0     Yes         No       1           No
      1    Male              0      No         No      34          Yes
      2    Male              0      No         No       2          Yes

            MultipleLines InternetService OnlineSecurity OnlineBackup  \
      0  No phone service             DSL             No          Yes
      1                No             DSL            Yes           No
      2                No             DSL            Yes          Yes

        DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
      0               No          No          No              No  Month-to-month
      1              Yes          No          No              No        One year
      2               No          No          No              No  Month-to-month

        PaperlessBilling     PaymentMethod  MonthlyCharges  TotalCharges Churn
      0              Yes  Electronic check           29.85         29.85    No
      1               No      Mailed check           56.95       1889.50    No
      2              Yes      Mailed check           53.85        108.15   Yes
```

## 9 Label encoding of target column

[41]:

```
df["Churn"] = df["Churn"].replace({"Yes": 1, "No": 0})
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_30384\2364848822.py:1:
FutureWarning: Downcasting behavior in `replace` is deprecated and will be
removed in a future version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  df["Churn"] = df["Churn"].replace({"Yes": 1, "No": 0})
```

[42]:

```
df.head(4)
```

[42]:

```
   gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  Female              0     Yes         No       1           No
1    Male              0      No         No      34          Yes
2    Male              0      No         No       2          Yes
3    Male              0      No         No      45           No

      MultipleLines InternetService OnlineSecurity OnlineBackup  \
0  No phone service             DSL             No          Yes
1                No             DSL            Yes           No
2                No             DSL            Yes          Yes
3  No phone service             DSL            Yes           No

  DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0               No          No          No              No  Month-to-month
1              Yes          No          No              No        One year
2               No          No          No              No  Month-to-month
3              Yes         Yes          No              No        One year

  PaperlessBilling              PaymentMethod  MonthlyCharges  TotalCharges  \
0              Yes           Electronic check           29.85         29.85
1               No               Mailed check           56.95       1889.50
2              Yes               Mailed check           53.85        108.15
3               No  Bank transfer (automatic)           42.30       1840.75

   Churn
0      0
1      0
2      1
3      0
```

[43]:

```
print(df["Churn"].value_counts())
```

```
Churn
0    5174
1    1869
```

```
Name: count, dtype: int64
```

## 10 Label encoding

[79]:

```
# identifying columns with object data type
object_columns = df.select_dtypes(include="object").columns
```

[80]:

```
print(object_columns)
```

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

[81]:

```
# initialize a dictionary to save the encoders
encoders = {}

# apply label encoding and store the encoders
for column in object_columns:
  label_encoder = LabelEncoder()
  df[column] = label_encoder.fit_transform(df[column])
  encoders[column] = label_encoder


# save the encoders to a pickle file
with open("encoders.pkl", "wb") as f:
  pickle.dump(encoders, f)
```

[82]:

```
encoders
```

[82]:

```
{'gender': LabelEncoder(),
 'Partner': LabelEncoder(),
 'Dependents': LabelEncoder(),
 'PhoneService': LabelEncoder(),
 'MultipleLines': LabelEncoder(),
 'InternetService': LabelEncoder(),
 'OnlineSecurity': LabelEncoder(),
 'OnlineBackup': LabelEncoder(),
 'DeviceProtection': LabelEncoder(),
 'TechSupport': LabelEncoder(),
 'StreamingTV': LabelEncoder(),
 'StreamingMovies': LabelEncoder(),
 'Contract': LabelEncoder(),
 'PaperlessBilling': LabelEncoder(),
 'PaymentMethod': LabelEncoder()}
```

```
[84]: df.head()
```

```
[84]:    gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  \
      0       0              0        1           0       1             0
      1       1              0        0           0      34             1
      2       1              0        0           0       2             1
      3       1              0        0           0      45             0
      4       0              0        0           0       2             1

         MultipleLines  InternetService  OnlineSecurity  OnlineBackup  \
      0              1                0               0             2
      1              0                0               2             0
      2              0                0               2             2
      3              1                0               2             0
      4              0                1               0             0

         DeviceProtection  TechSupport  StreamingTV  StreamingMovies  Contract  \
      0                 0            0            0                0         0
      1                 2            0            0                0         1
      2                 0            0            0                0         0
      3                 2            2            0                0         1
      4                 0            0            0                0         0

         PaperlessBilling  PaymentMethod  MonthlyCharges  TotalCharges  Churn
      0                 1              2           29.85         29.85      0
      1                 0              3           56.95       1889.50      0
      2                 1              3           53.85        108.15      1
      3                 0              0           42.30       1840.75      0
      4                 1              2           70.70        151.65      1
```

## 11 Training and Test data split

```
[44]: # splitting the features and target
      X = df.drop(columns=["Churn"])
      y = df["Churn"]
```

```
[45]: # split training and test data
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
       ↪random_state=42)
```

```
[46]: print(y_train.shape)
```

```
(5634,)
```

```
[47]: print(y_train.value_counts())
```

```
Churn
0    4138
```

```
1    1496
Name: count, dtype: int64
```

## 12 (SMOTS)

[58]:

```python
smote = SMOTE(random_state=42)
```

[60]:

```python
from sklearn.preprocessing import LabelEncoder

# Encode categorical columns
categorical_cols = X.select_dtypes(include='object').columns
label_encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
    label_encoders[col] = le  # save encoders in case you need to decode later

# Re-split the data after encoding
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply SMOTE
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

[61]:

```python
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

[62]:

```python
print(y_train_smote.shape)
```

```
(8276,)
```

[63]:

```python
print(y_train_smote.value_counts())
```

```
Churn
0    4138
1    4138
Name: count, dtype: int64
```

## 13 Data Modelling

[64]:

```python
# dictionary of models
models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42)
}
```

[65]:

```
# dictionary to store the cross validation results
cv_scores = {}

# perform 5-fold cross validation for each model
for model_name, model in models.items():
  print(f"Training {model_name} with default parameters")
  scores = cross_val_score(model, X_train_smote, y_train_smote, cv=5,␣
 ↪scoring="accuracy")
  cv_scores[model_name] = scores
  print(f"{model_name} cross-validation accuracy: {np.mean(scores):.2f}")
  print("-"*70)
```

```
Training Decision Tree with default parameters
Decision Tree cross-validation accuracy: 0.78
----------------------------------------------------------------------
Training Random Forest with default parameters
Random Forest cross-validation accuracy: 0.84
----------------------------------------------------------------------
Training XGBoost with default parameters
XGBoost cross-validation accuracy: 0.83
----------------------------------------------------------------------
```

[66]:

```
cv_scores
```

[66]:

```
{'Decision Tree': array([0.68297101, 0.71601208, 0.81993958, 0.83564955,
0.83746224]),
 'Random Forest': array([0.72826087, 0.7734139 , 0.90332326, 0.89969789,
0.8978852 ]),
 'XGBoost': array([0.71135266, 0.74864048, 0.91178248, 0.88640483, 0.91117825])}
```

## 14 Random Forest give highest accuracy

[67]:

```
rfc = RandomForestClassifier(random_state=42)
```

[68]:

```
rfc.fit(X_train_smote, y_train_smote)
```

[68]:

```
RandomForestClassifier(random_state=42)
```

[69]:

```
print(y_test.value_counts())
```

```
Churn
0    1036
1     373
Name: count, dtype: int64
```

## 15 Model Evaluation

[71]:

```
# evaluate on test data
y_test_pred = rfc.predict(X_test)

print("Accuracy Score:\n", accuracy_score(y_test, y_test_pred))
print("Confsuion Matrix:\n", confusion_matrix(y_test, y_test_pred))
print("Classification Report:\n", classification_report(y_test, y_test_pred))
```

```
Accuracy Score:
 0.7771469127040455
Confsuion Matrix:
 [[879 157]
 [157 216]]
Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.85      0.85      1036
           1       0.58      0.58      0.58       373

    accuracy                           0.78      1409
   macro avg       0.71      0.71      0.71      1409
weighted avg       0.78      0.78      0.78      1409
```

[72]:

```
# save the trained model as a pickle file
model_data = {"model": rfc, "features_names": X.columns.tolist()}


with open("customer_churn_model.pkl", "wb") as f:
  pickle.dump(model_data, f)
```

## 16 Load the saved model and build a pridictive system

[73]:

```
# load teh saved model and the feature names

with open("customer_churn_model.pkl", "rb") as f:
  model_data = pickle.load(f)

loaded_model = model_data["model"]
feature_names = model_data["features_names"]
```

[74]:

```
print(loaded_model)
```

```
RandomForestClassifier(random_state=42)
```

[75]:

```
print(feature_names)
```

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

[85]:

```
input_data = {
    'gender': 'Female',
    'SeniorCitizen': 0,
    'Partner': 'Yes',
    'Dependents': 'No',
    'tenure': 1,
    'PhoneService': 'No',
    'MultipleLines': 'No phone service',
    'InternetService': 'DSL',
    'OnlineSecurity': 'No',
    'OnlineBackup': 'Yes',
    'DeviceProtection': 'No',
    'TechSupport': 'No',
    'StreamingTV': 'No',
    'StreamingMovies': 'No',
    'Contract': 'Month-to-month',
    'PaperlessBilling': 'Yes',
    'PaymentMethod': 'Electronic check',
    'MonthlyCharges': 29.85,
    'TotalCharges': 29.85
}


input_data_df = pd.DataFrame([input_data])

with open("encoders.pkl", "rb") as f:
  encoders = pickle.load(f)


# encode categorical featires using teh saved encoders
for column, encoder in encoders.items():
  input_data_df[column] = encoder.transform(input_data_df[column])

# make a prediction
prediction = loaded_model.predict(input_data_df)
pred_prob = loaded_model.predict_proba(input_data_df)

print(prediction)

# results
print(f"Prediction: {'Churn' if prediction[0] == 1 else 'No Churn'}")
print(f"Prediciton Probability: {pred_prob}")
```

```
[0]
Prediction: No Churn
Prediciton Probability: [[0.79 0.21]]
```

[86]:

```
encoders
```

[86]:

```
{'gender': LabelEncoder(),
 'Partner': LabelEncoder(),
 'Dependents': LabelEncoder(),
 'PhoneService': LabelEncoder(),
 'MultipleLines': LabelEncoder(),
 'InternetService': LabelEncoder(),
 'OnlineSecurity': LabelEncoder(),
 'OnlineBackup': LabelEncoder(),
 'DeviceProtection': LabelEncoder(),
 'TechSupport': LabelEncoder(),
 'StreamingTV': LabelEncoder(),
 'StreamingMovies': LabelEncoder(),
 'Contract': LabelEncoder(),
 'PaperlessBilling': LabelEncoder(),
 'PaymentMethod': LabelEncoder()}
```

# 17 To do:

1. Implement Hyperparameter Tuining
2. Try Model Selection
3. Try downsampling
4. Try to address teh overfitting
5. Try Startified k fold CV

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: