1 Customer Churn Prediction Using Machine Learning

1.1 Importing the dependencies

[6]:
```
import pandas as pd
import numpy as np
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
```

[ ]:
```

```

1.2 2. Data Loading and Understanding

[33]:
```
# load teh csv data to a pandas dataframe
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

[9]:
```
df.shape
```

[9]:
```
(7043, 21)
```

[12]:
```
pd.set_option("display.max_columns", None)
```

[13]:
```
df.head()
```

[13]:
```
   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 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
4                No     Fiber optic             No           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
4               No          No          No              No  Month-to-month

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

  Churn
0    No
1    No
2   Yes
3    No
4   Yes
```

[ ]:

[14]:
```
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
```

[ ]:

[15]:
```python
# dropping customerID column as this is not required for modelling
df = df.drop(columns=["customerID"])
```

[18]:
```python
df.head(2)
```

[18]:
```
   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
```

[ ]:

[20]:
```python
df.columns
```

[20]:
```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
```

```
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

[22]:
```
print(df['gender'].unique())
```

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

[23]:
```
print(df['SeniorCitizen'].unique())
```

```
[0 1]
```

[ ]:
```

```

[25]:
```
# 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']
-------------------------------------------------
```

[ ]:

[26]:
```
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
```

[ ]:

[57]:
```
#df["TotalCharges"] = df["TotalCharges"].astype(float)
```

[58]:
```
df[df["TotalCharges"]==" "]
```

[58]:
```
      customerID  gender  SeniorCitizen Partner Dependents  tenure  \
488   4472-LVYGI  Female              0     Yes        Yes       0
753   3115-CZMZD    Male              0      No        Yes       0
936   5709-LVOEQ  Female              0     Yes        Yes       0
```

1082	4367-NUYAO	Male	0	Yes	Yes	0
1340	1371-DWPAZ	Female	0	Yes	Yes	0
3331	7644-OMVMY	Male	0	Yes	Yes	0
3826	3213-VVOLG	Male	0	Yes	Yes	0
4380	2520-SGTTA	Female	0	Yes	Yes	0
5218	2923-ARZLG	Male	0	Yes	Yes	0
6670	4075-WKNIU	Female	0	Yes	Yes	0
6754	2775-SEFEE	Male	0	No	Yes	0

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

[ ]:

```

```

[59]:

```python
len(df[df["TotalCharges"]==" "])
```

[59]: 11

[60]:

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

[61]:

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

[62]:

```python
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   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

[ ]:

```

```

[63]:

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

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

[ ]:

```

```

1.3 Insights:

1. Customer ID removed as it is not required for modelling
2. No mmissing values in the dataset
3. Missing values in the TotalCharges column were replaced with 0
4. Class imbalance identified in the target

[ ]:

```

```

1.4 3. Exploratory Data Analysis (EDA)

[64]:

```
df.shape
```

[64]:

```
(7043, 21)
```

[65]:

```
df.columns
```

[65]:

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

[66]:

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

[66]:

```
     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

      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       1889.50    No
```

[67]:

```python
df.describe()
```

[67]:

	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

[ ]:

```python

```

1.4.1 Numerical Features - Analysis

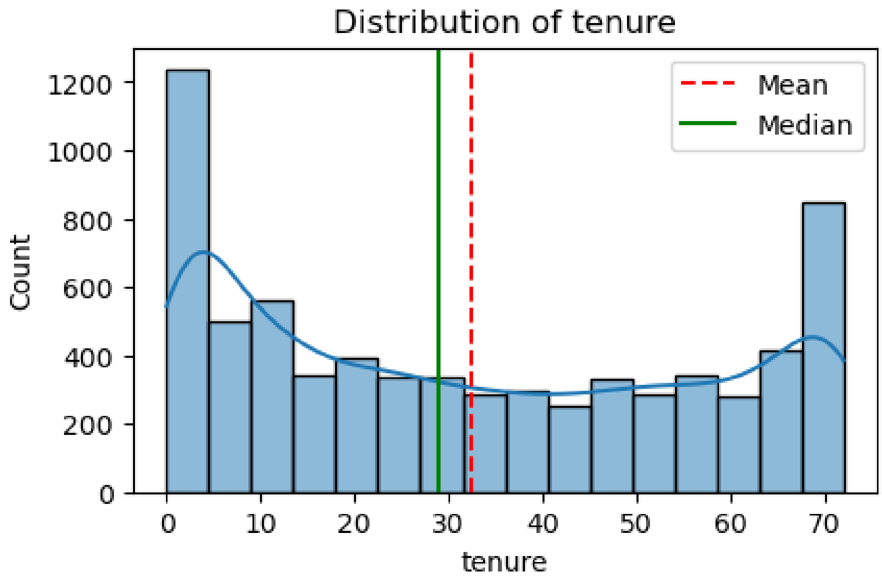
Understand the distribution of teh numerical features

[68]:

```python
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")
```

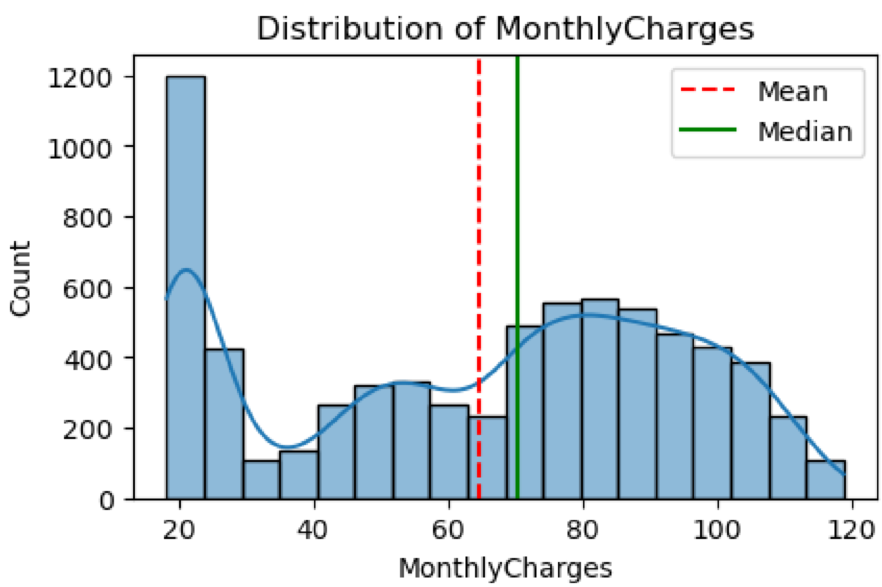
```
    plt.legend()

    plt.show()
```

[43]:
```
plot_histogram(df, "tenure")
```



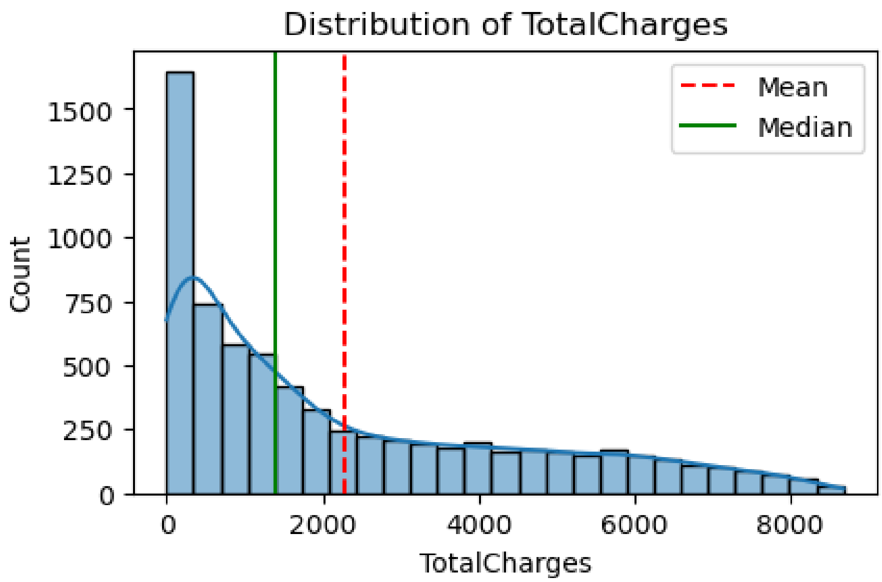
[ ]:

[44]:
```
plot_histogram(df, "MonthlyCharges")
```



[ ]:

[73]:
```
plot_histogram(df, "TotalCharges")
```

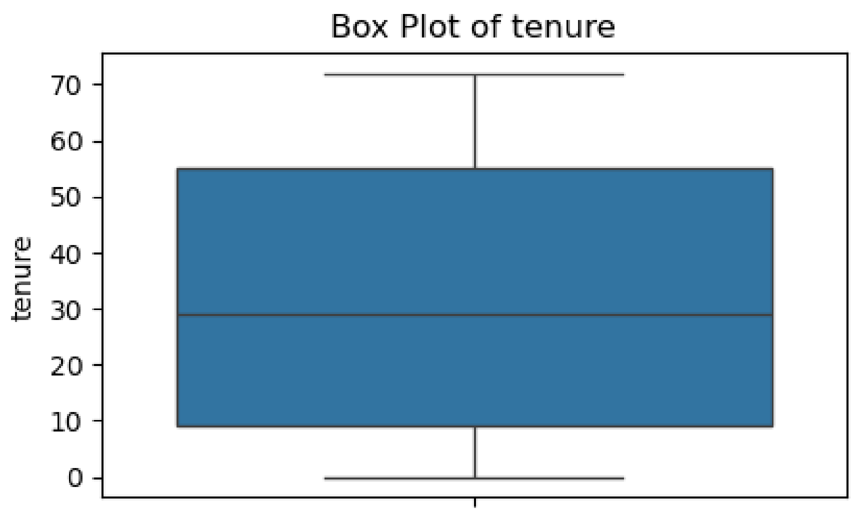


[ ]:

1.4.2 Box plot for numerical features

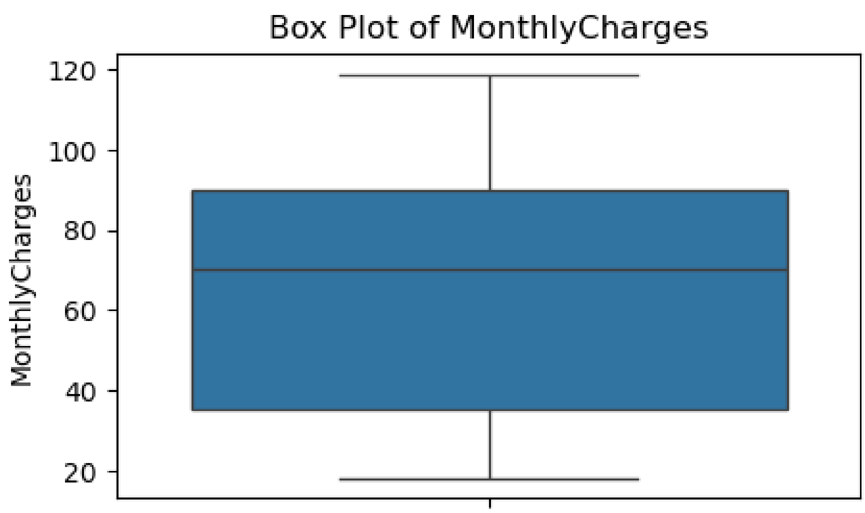
[49]:
```
def plot_boxplot(df, column_name):

  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
```

[50]:
```
plot_boxplot(df, "tenure")
```



[ ]:

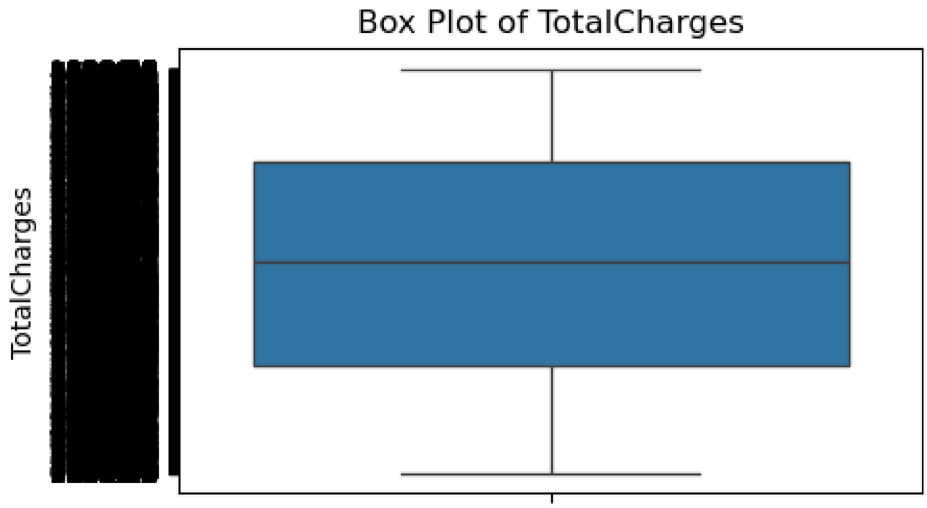
```
plot_boxplot(df, "MonthlyCharges")
```



[ ]:

[51]:

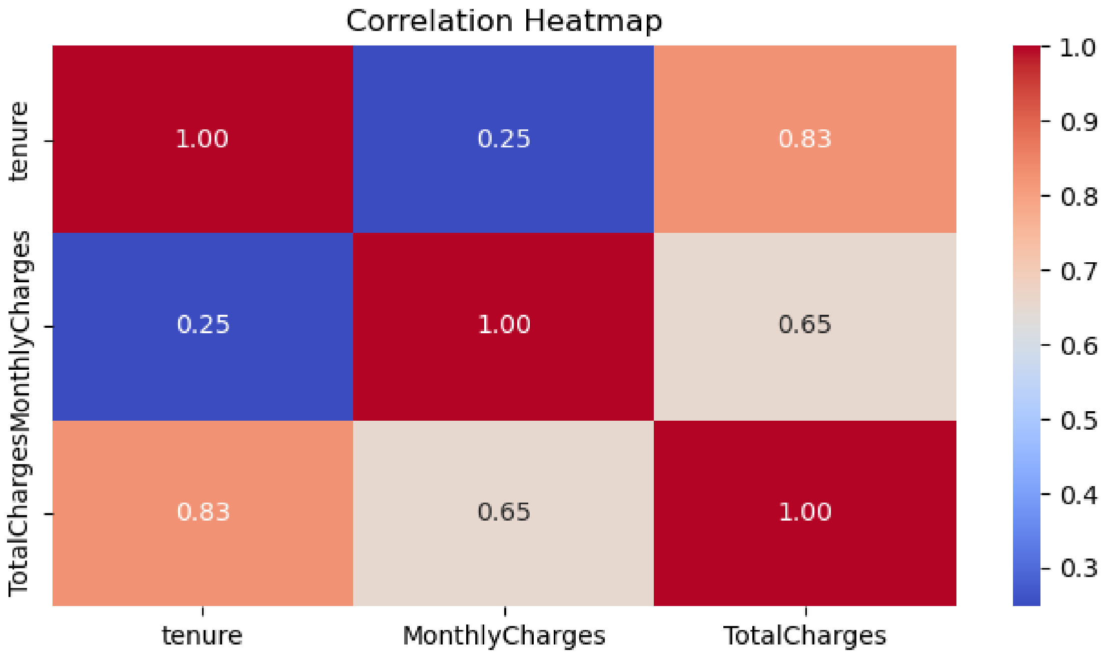
```
plot_boxplot(df, "TotalCharges")
```



[ ]:

1.4.3 Correlation Heatmap for numerical columns

[69]:
```
# 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()
```



[ ]:

```

```

1.4.4 Categorical features - Analysis

[70]:
```
df.columns
```

[70]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

[71]:
```
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
```

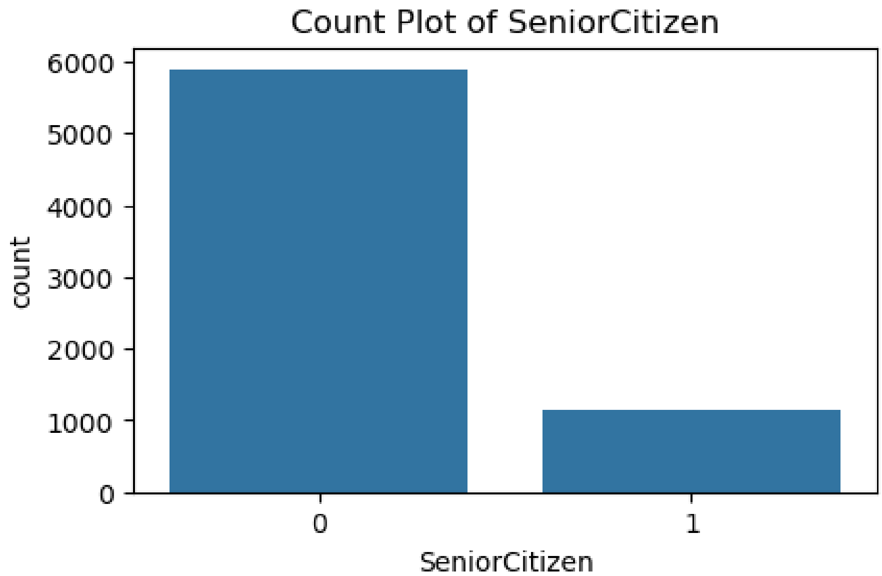
```
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  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   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

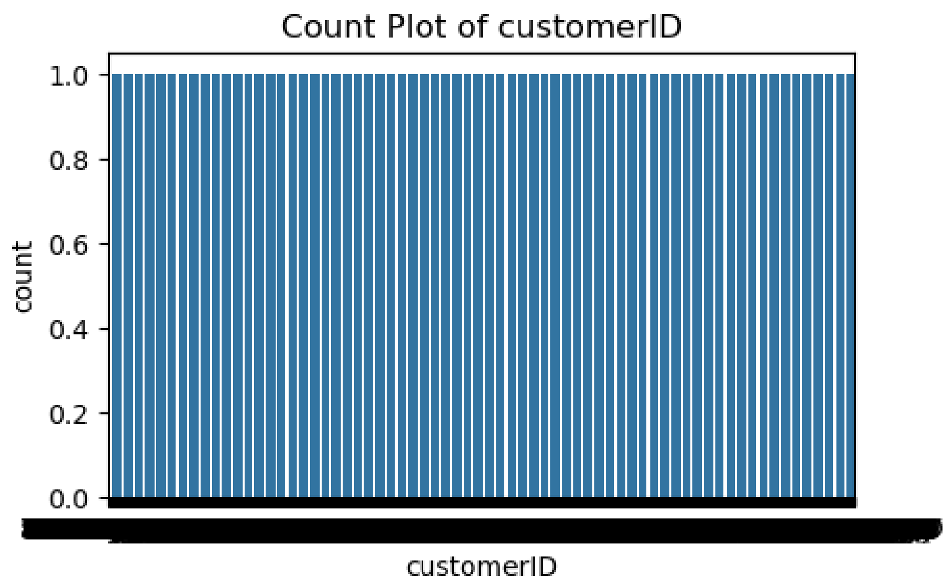
[ ]:

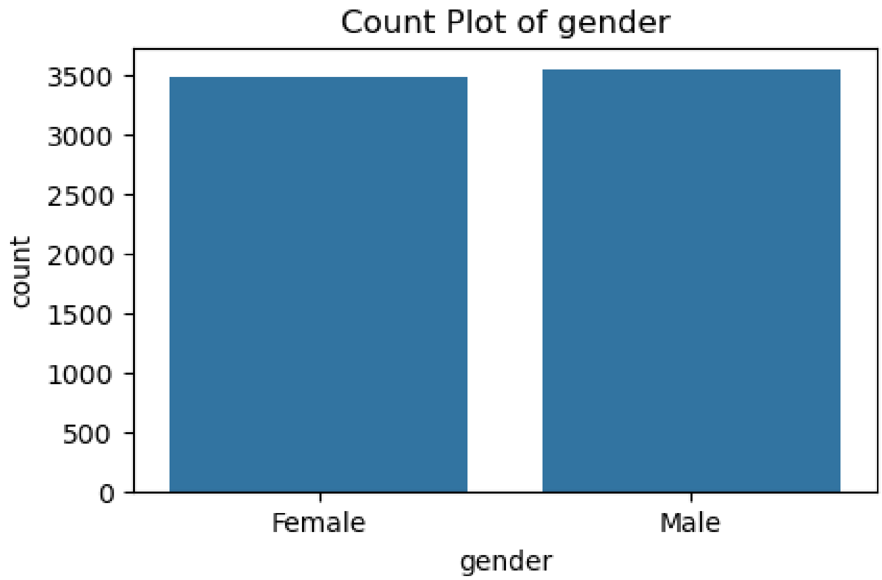
1.4.5 Countplot for categorical columns

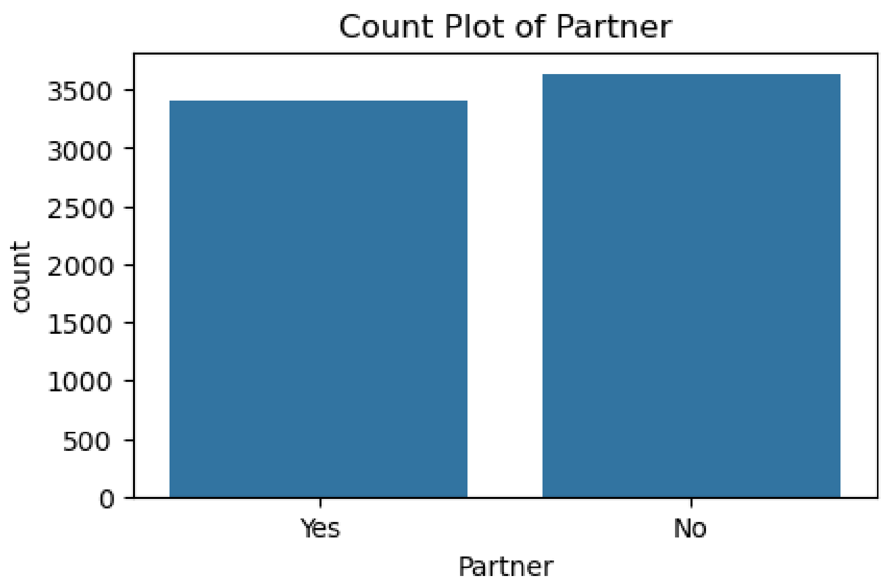
[72]:

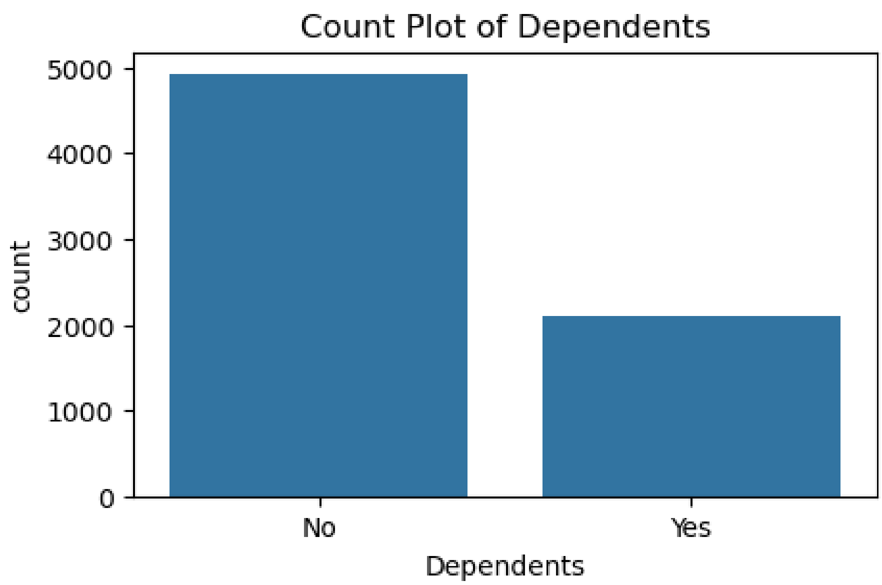
```
object_cols = df.select_dtypes(include="object").columns.to_list()

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()
```

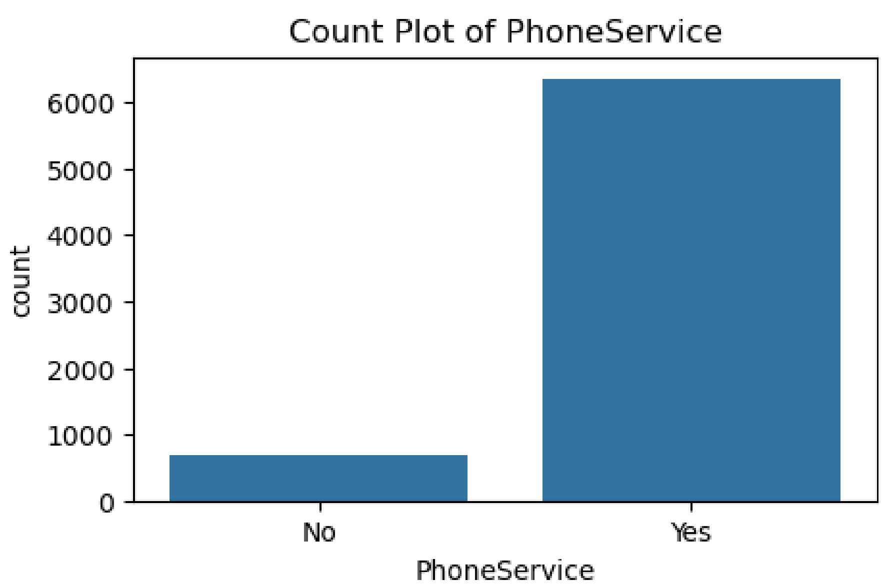

Count Plot of SeniorCitizen
6000
5000
4000
3000
2000
1000
0
count
0
1
SeniorCitizen

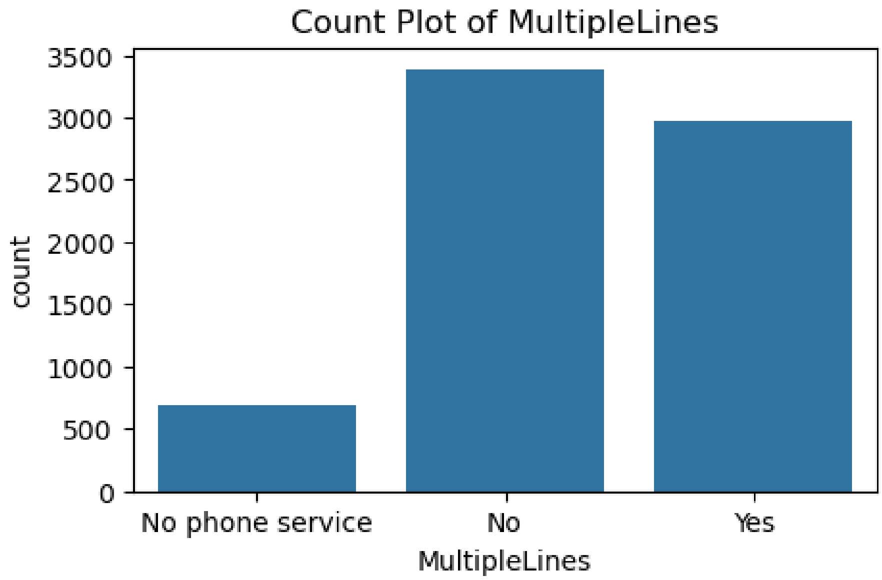

Count Plot of customerID
1.0
0.8
0.6
0.4
0.2
0.0
count
customerID

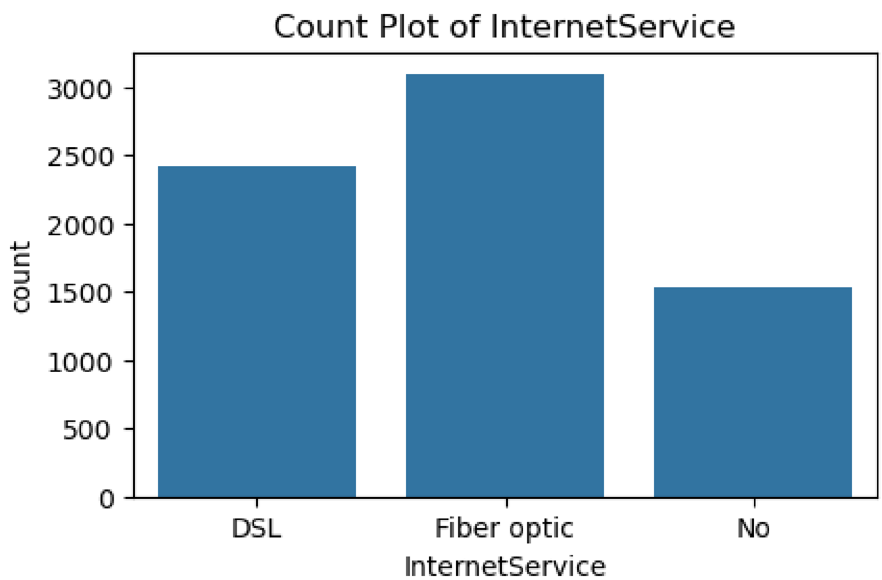

Count Plot of gender
3500
3000
2500
2000
1500
1000
500
0
count
Female
Male
gender

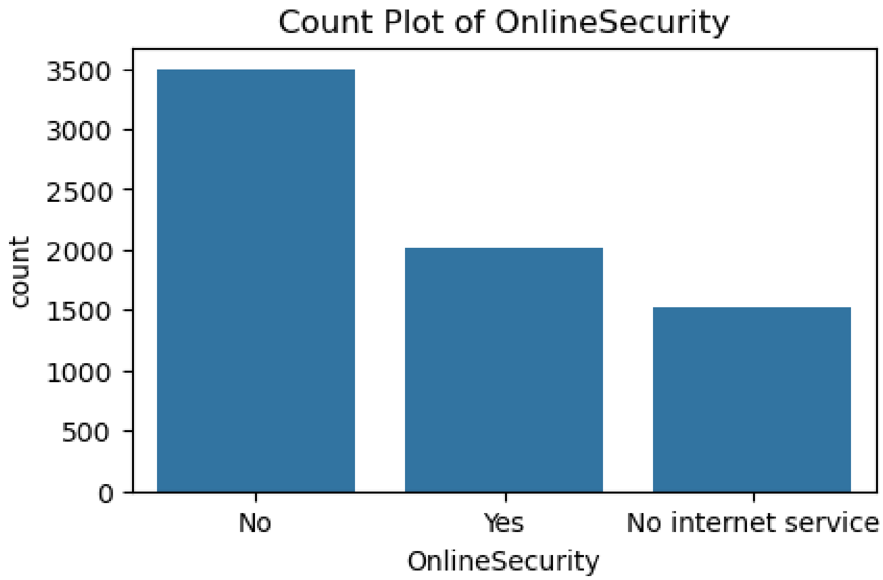

Count Plot of Partner
3500
3000
2500
2000
1500
1000
500
0
count
Yes
No
Partner

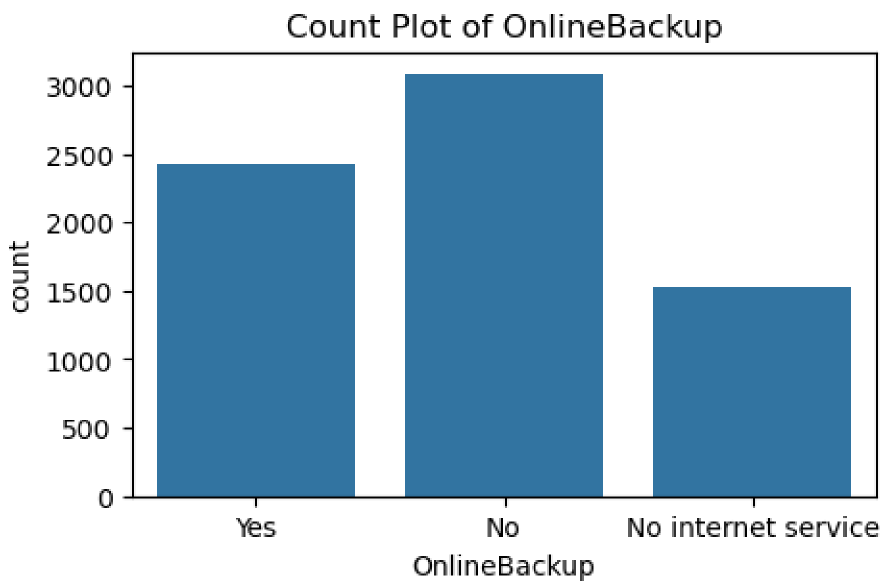

Count Plot of Dependents
5000
4000
3000
2000
1000
0
count
No
Yes
Dependents

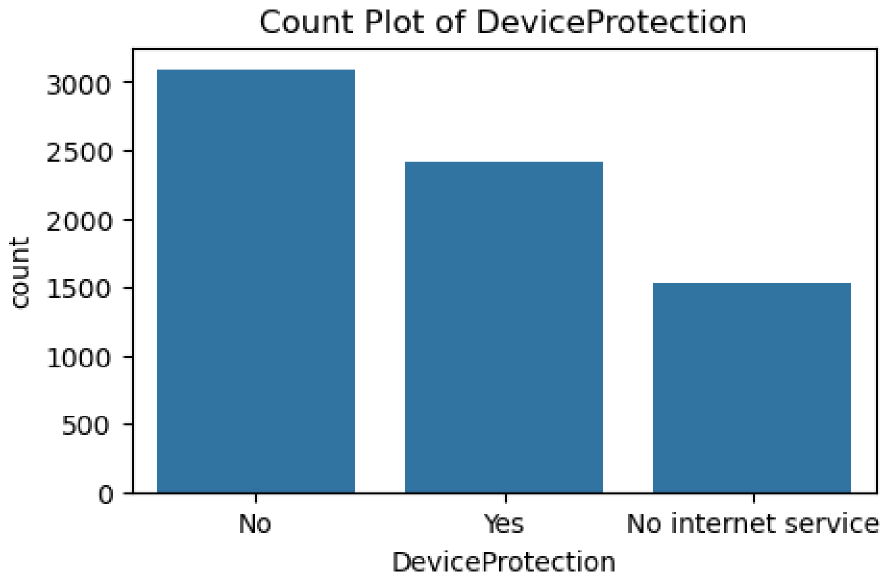

Count Plot of PhoneService
6000
5000
4000
3000
2000
1000
0
count
No
Yes
PhoneService

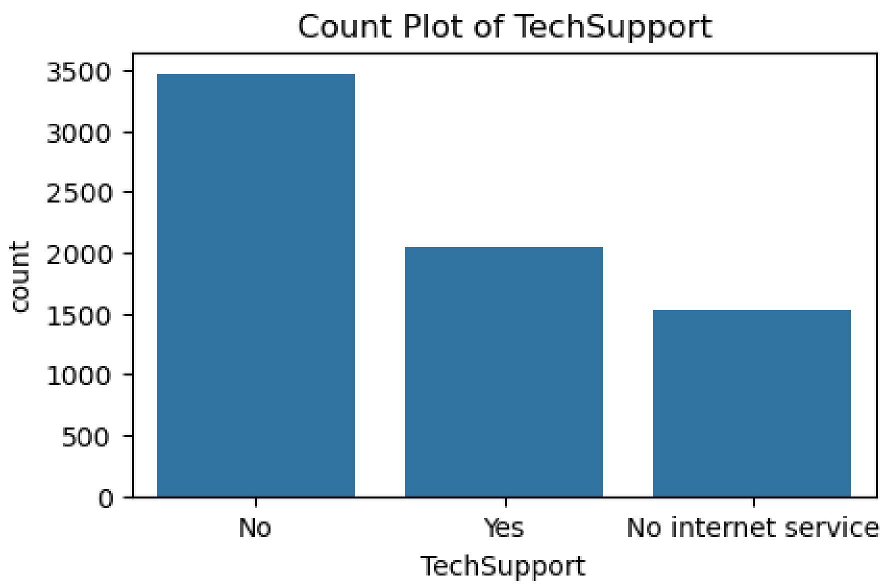

Count Plot of MultipleLines
count
3500
3000
2500
2000
1500
1000
500
0
No phone service
No
Yes
MultipleLines

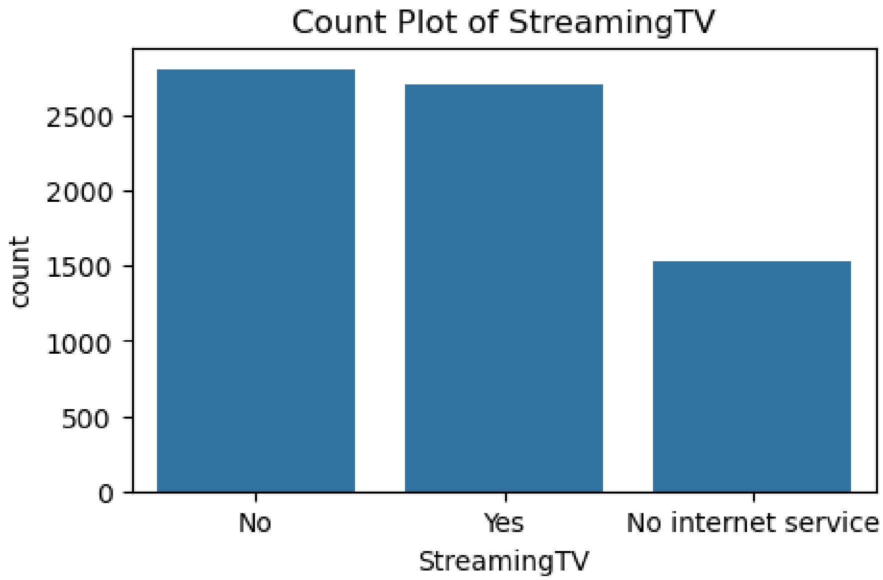

Count Plot of InternetService
count
3000
2500
2000
1500
1000
500
0
DSL
Fiber optic
No
InternetService

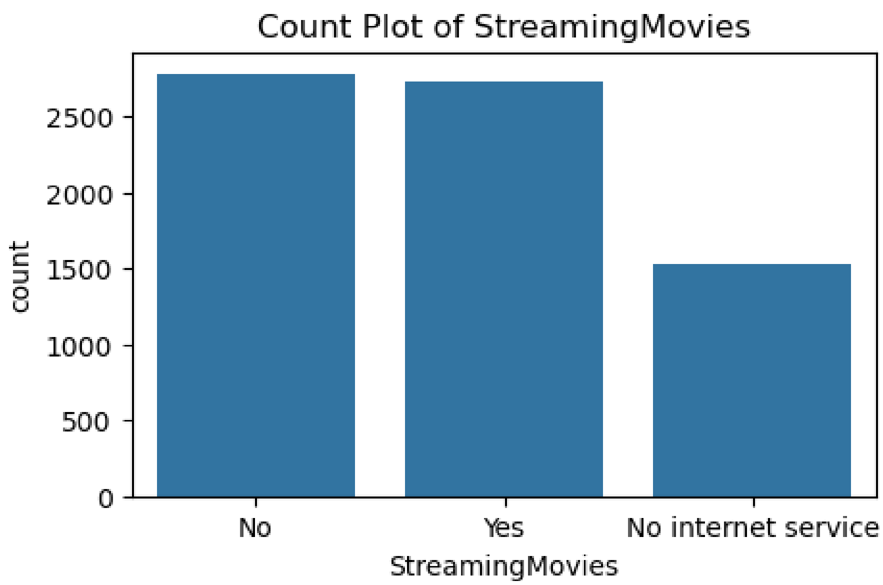

Count Plot of OnlineSecurity
3500
3000
2500
2000
1500
1000
500
0
count
No
Yes
No internet service
OnlineSecurity

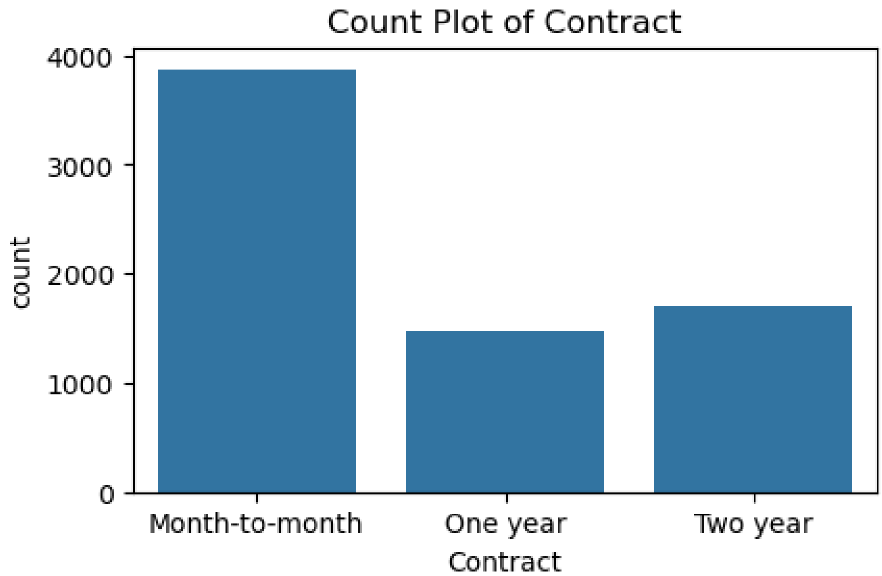

Count Plot of OnlineBackup
3000
2500
2000
1500
1000
500
0
count
Yes
No
No internet service
OnlineBackup

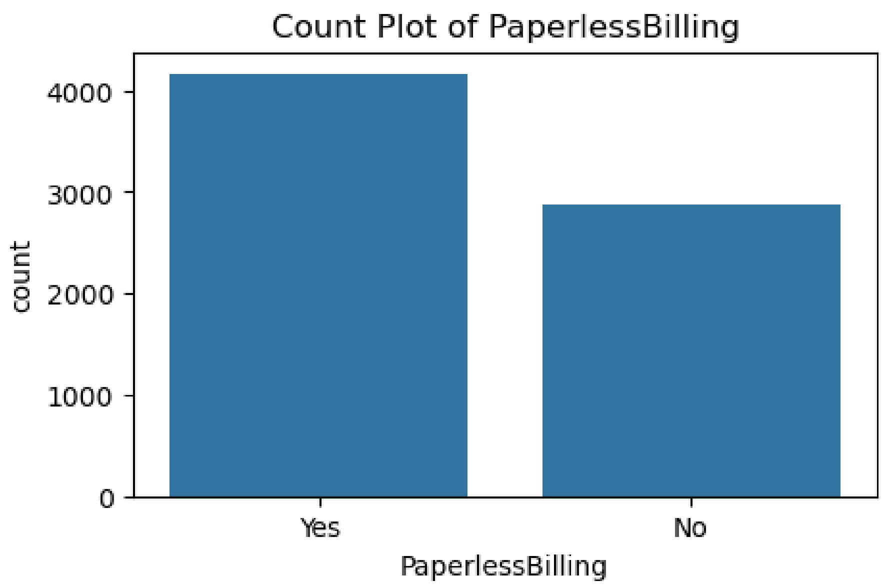

Count Plot of DeviceProtection
count
3000
2500
2000
1500
1000
500
0
No
Yes
No internet service
DeviceProtection

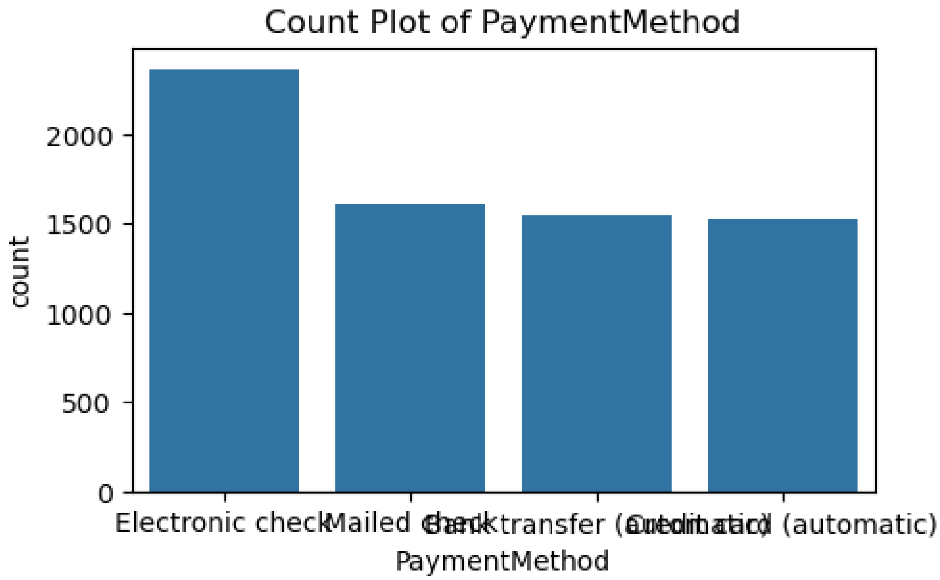

Count Plot of TechSupport
count
3500
3000
2500
2000
1500
1000
500
0
No
Yes
No internet service
TechSupport

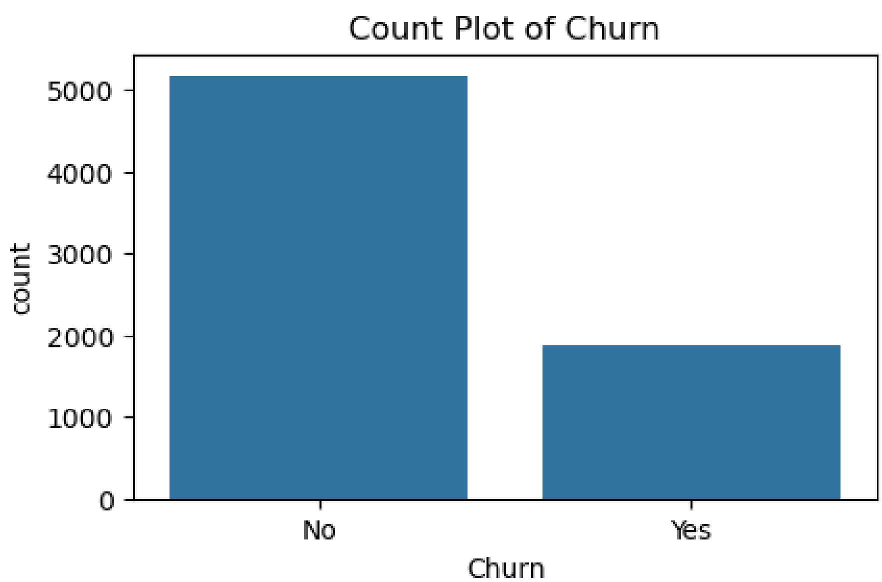

Count Plot of StreamingTV
count
2500
2000
1500
1000
500
0
No
Yes
No internet service
StreamingTV


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


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


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





[ ]:

```

```

2 4. Data Preprocessing

[75]:

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

[75]:
```
   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

      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       1889.50    No
```

[ ]:

2.0.1 Label encoding of target column

[78]:
```python
df["Churn"] = df["Churn"].replace({"Yes": 1, "No": 0})
```

[ ]:

[80]:
```python
df.head(2)
```

[80]:
```
   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

      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      0
1               No      Mailed check           56.95       1889.50      0
```

[ ]:

[81]:
```python
print(df["Churn"].value_counts())
```

```
Churn
```

```
0    5174
1    1869
Name: count, dtype: int64
```

[ ]:

2.0.2 Label encoding of categorical fetaures

[82]:
```
# identifying columns with object data type
object_columns = df.select_dtypes(include="object").columns
```

[83]:
```
print(object_columns)
```

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

[ ]:

[84]:
```
# 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)
```

[85]:
```
encoders
```

[85]:
```
{'customerID': LabelEncoder(),
 '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()}
```

[87]:
```
df.head(2)
```

[87]:
```
   customerID  gender  SeniorCitizen  Partner  Dependents  tenure  \
0        5375       0              0        1           0       1
1        3962       1              0        0           0      34

   PhoneService  MultipleLines  InternetService  OnlineSecurity  OnlineBackup  \
0             0              1                0               0             2
1             1              0                0               2             0

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

   PaperlessBilling  PaymentMethod  MonthlyCharges  TotalCharges  Churn
0                 1              2           29.85         29.85      0
1                 0              3           56.95       1889.50      0
```

[ ]:
```

```

2.0.3 Traianing and test data split

[88]:
```
# splitting the features and target
X = df.drop(columns=["Churn"])
y = df["Churn"]
```

[89]:
```
# 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)
```

[90]:
```
print(y_train.shape)
```

```
(5634,)
```

[91]:
```
print(y_train.value_counts())
```

```
Churn
0    4138
1    1496
Name: count, dtype: int64
```

[ ]:

```

```

Synthetic Minority Oversampling TEchnique (SMOTE)

[92]:
```
smote = SMOTE(random_state=42)
```

[93]:
```
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

[94]:
```
print(y_train_smote.shape)
```

```
(8276,)
```

[95]:
```
print(y_train_smote.value_counts())
```

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

[ ]:

```

```

3 5. Model Training

Training with default hyperparameters

[96]:
```
# dictionary of models
models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42)
}
```

[97]:
```
# 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: 0.84
----------------------------------------------------------------------
```

[98]:
```
cv_scores
```

[98]:
```
{'Decision Tree': array([0.68297101, 0.7081571 , 0.81873112, 0.82779456,
0.83987915]),
 'Random Forest': array([0.73248792, 0.76495468, 0.90755287, 0.89848943,
0.90574018]),
 'XGBoost': array([0.70833333, 0.74682779, 0.91359517, 0.90090634, 0.91359517])}
```

[ ]:

3.0.1 Random Forest gives the highest accuracy compared to other models with default parameters

[99]:
```
rfc = RandomForestClassifier(random_state=42)
```

[100]:
```
rfc.fit(X_train_smote, y_train_smote)
```

[100]:
```
RandomForestClassifier(random_state=42)
```

[101]:
```
print(y_test.value_counts())
```

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

[ ]:

4 6. Model Evaluation

[102]:
```
# 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:
 0.7792760823278921
Confsuion Matrix:
 [[875 161]
```

```
 [150 223]]
Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.84      0.85      1036
           1       0.58      0.60      0.59       373

    accuracy                           0.78      1409
   macro avg       0.72      0.72      0.72      1409
weighted avg       0.78      0.78      0.78      1409
```

[103]:
```
# 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)
```

[ ]:

4.1 7. Load the saved model and build a Predictive System

[104]:
```
# 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"]
```

[105]:
```
print(loaded_model)
```

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

[106]:
```
print(feature_names)
```

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

[ ]:

[117]:
```
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
}


encoders.pop("customerID", None)  # Remove customerID encoder


# Check for missing or extra columns
expected_features = set(loaded_model.feature_names_in_)  # Features expected by␣
 ↪the model
current_features = set(input_data_df.columns)

missing_features = expected_features - current_features
extra_features = current_features - expected_features

print(f"Missing features: {missing_features}")
print(f"Extra features: {extra_features}")
```

```
Missing features: {'customerID'}
Extra features: set()
```

[118]:
```
encoders
```

[118]:
```
{'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()}
```

[ ]:

```
```

[ ]:

```
```

[ ]:

```
```