tomer-churn-prediction-using-ml-1

December 18, 2025

0.1 1.Importing the dependencies

[104]:
```
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
```

0.2 2. Data Loading and Understanding

[107]:
```
# load thecsv data to a pandas dataframe
df = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
```

[109]:
```
df.shape
```

[109]:
```
(7043, 21)
```

[110]:
```
df.head()
```

[110]:
```
   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
```

[111]:
```
df.tail()
```

[111]:
```
      customerID  gender  SeniorCitizen Partner Dependents  tenure  \
7038  6840-RESVB    Male              0     Yes        Yes      24
7039  2234-XADUH  Female              0     Yes        Yes      72
7040  4801-JZAZL  Female              0     Yes        Yes      11
7041  8361-LTMKD    Male              1     Yes         No       4
7042  3186-AJIEK    Male              0      No         No      66

     PhoneService     MultipleLines InternetService OnlineSecurity  \
7038          Yes               Yes             DSL            Yes
7039          Yes               Yes     Fiber optic             No
7040           No  No phone service             DSL            Yes
7041          Yes               Yes     Fiber optic             No
7042          Yes                No     Fiber optic            Yes

     OnlineBackup DeviceProtection TechSupport StreamingTV StreamingMovies  \
7038           No              Yes         Yes         Yes             Yes
7039          Yes              Yes          No         Yes             Yes
7040           No               No          No          No              No
7041           No               No          No          No              No
7042           No              Yes         Yes         Yes             Yes
```

```
            Contract PaperlessBilling              PaymentMethod  \
7038        One year              Yes               Mailed check
7039        One year              Yes    Credit card (automatic)
7040  Month-to-month              Yes           Electronic check
7041  Month-to-month              Yes               Mailed check
7042        Two year              Yes  Bank transfer (automatic)

      MonthlyCharges TotalCharges Churn
7038           84.80       1990.5    No
7039          103.20       7362.9    No
7040           29.60       346.45    No
7041           74.40        306.6   Yes
7042          105.65       6844.5    No
```

[112]:
```python
pd.set_option("display.max_columns", None)
```

[113]:
```python
df.head(2)
```

[113]:
```
   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       1889.5    No
```

[114]:
```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   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

[115]:
```
# dropping customerID column as this is not required for modelling
df = df.drop(columns=["customerID"])
```

[116]:
```
df.head(2)
```

[116]:
```
   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
```

[117]:
```
df.columns
```

[117]:
```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges', 'Churn'],
```

```
      dtype='object')
```

[118]:
```python
print(df["gender"].unique())
```

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

[119]:
```python
print(df["SeniorCitizen"].unique())
```

```
[0 1]
```

[120]:
```python
# 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']
--------------------------------------------------
```

[121]:
```
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
```

[122]:
```
#df["TotalCharges"] = df["TotalCharges"].astype(float)
```

[123]:
```
df[df["TotalCharges"]==" "]
```

[123]:

	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
```

[124]:
```python
len(df[df["TotalCharges"]==" "])
```

[124]:
```
11
```

[125]:
```python
df["TotalCharges"] = df["TotalCharges"].replace({" ": "0.0"})
```

[126]:
```python
df["TotalCharges"] = df["TotalCharges"].astype(float)
```

[127]:
```python
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
```

[128]:
```python
# checking the class distribution of target column
print(df["Churn"].value_counts())
```

```
Churn
```

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

0.2.1 Insights:

0.2.2 Customer ID removed as it is not required for modelling

0.2.3 No mmissing values in the dataset

0.2.4 Missing values in the TotalCharges column were replaced with 0

0.2.5 Class imbalance identified in the target

0.3 3.. Exploratory Data Analysis (EDA)

[129]:
```
df.shape
```

[129]:
```
(7043, 20)
```

[130]:
```
df.columns
```

[130]:
```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

[131]:
```
df.head(2)
```

[131]:
```
   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       1889.50    No
```

[133]:
```
df.describe()
```

[133]:

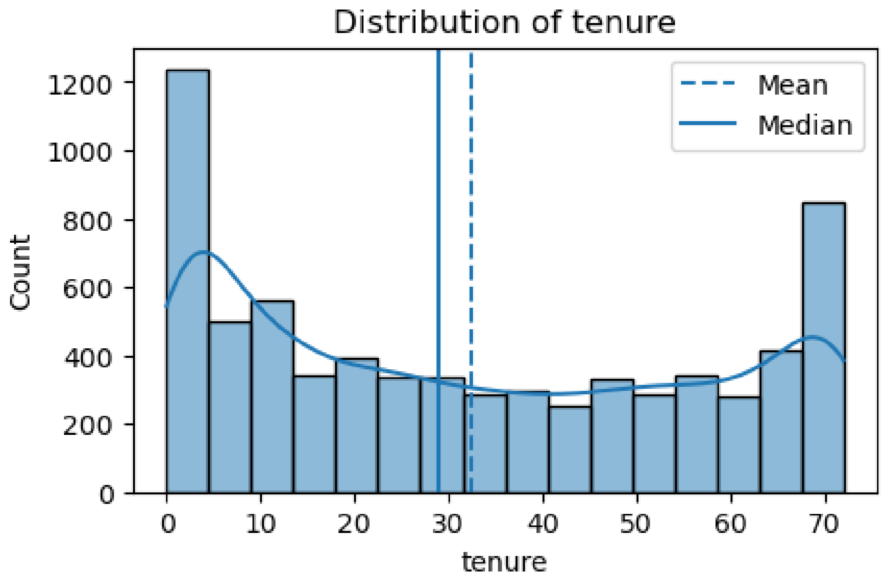
	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692	2279.734304
std	0.368612	24.559481	30.090047	2266.794470
min	0.000000	0.000000	18.250000	0.000000
25%	0.000000	9.000000	35.500000	398.550000
50%	0.000000	29.000000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

0.3.1 Numerical Features - Analysis

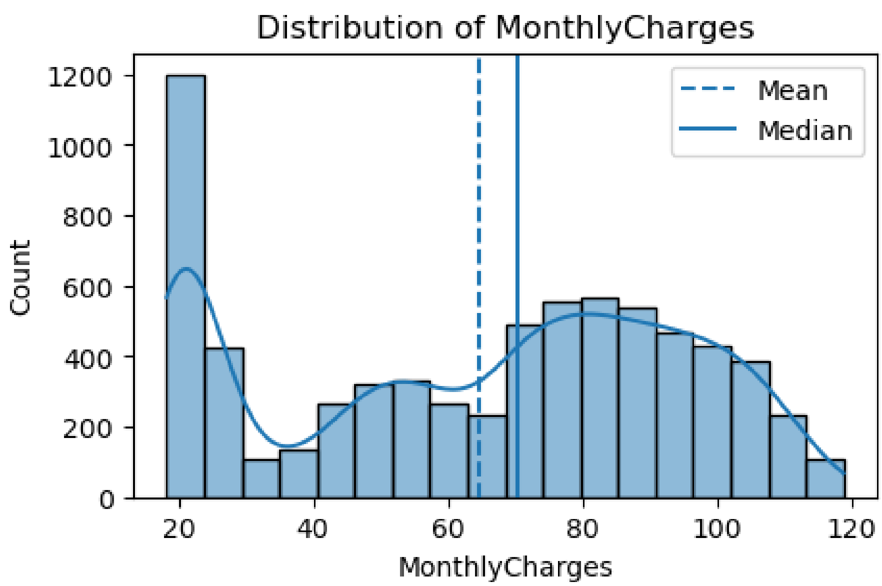
0.3.2 Understand the distribution of teh numerical features

[135]:
```
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 mean and median
    col_mean = df[column_name].mean()
    col_median = df[column_name].median()

    # add vertical lines
    plt.axvline(col_mean, linestyle="--", label="Mean")
    plt.axvline(col_median, linestyle="-", label="Median")

    plt.legend()
    plt.show()
```

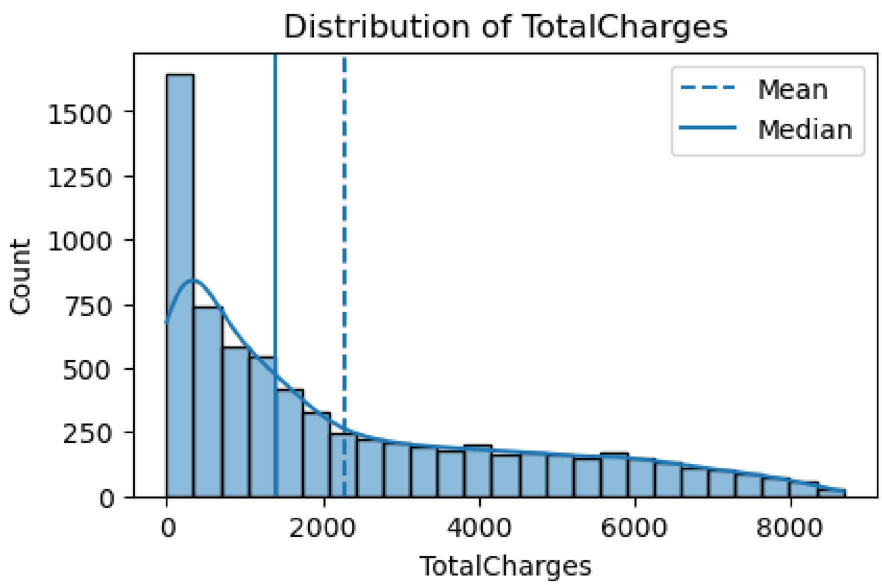
[136]:
```
plot_histogram(df, "tenure")
```



[137]:
```
plot_histogram(df, "MonthlyCharges")
```



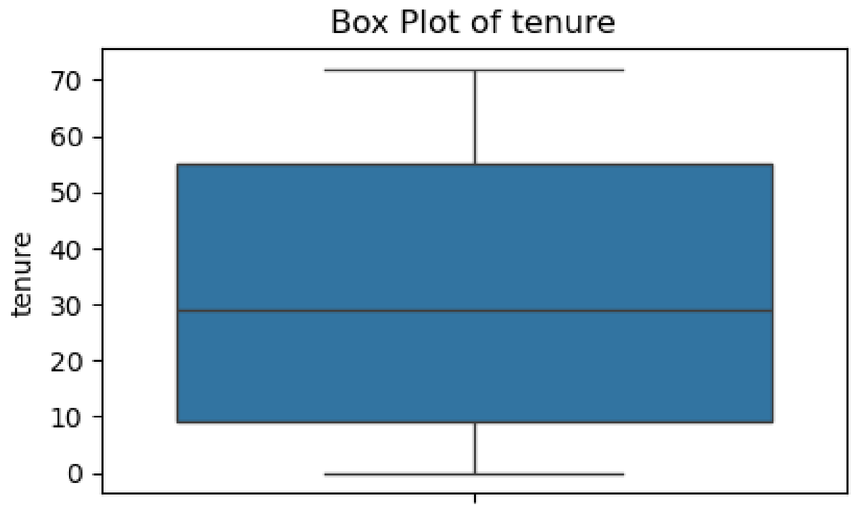
[138]:
```
plot_histogram(df, "TotalCharges")
```



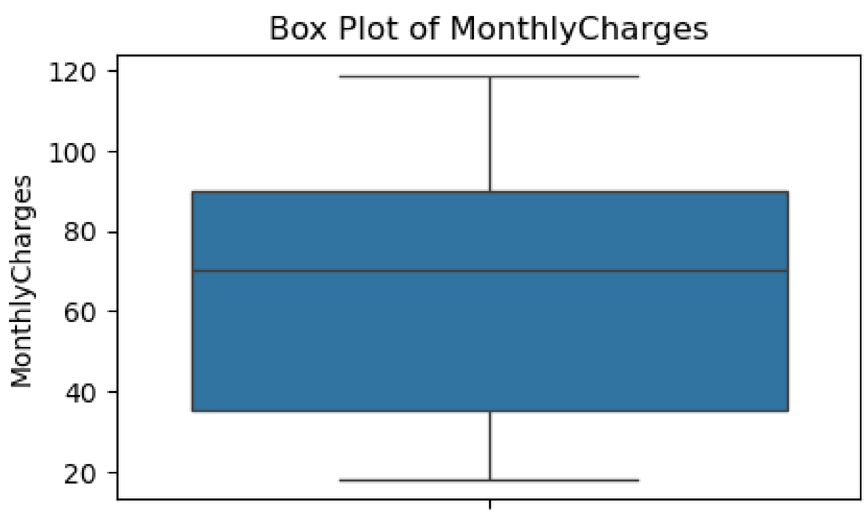
0.3.3 Box plot for numerical features

[141]:
```
import matplotlib.pyplot as plt

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()
```

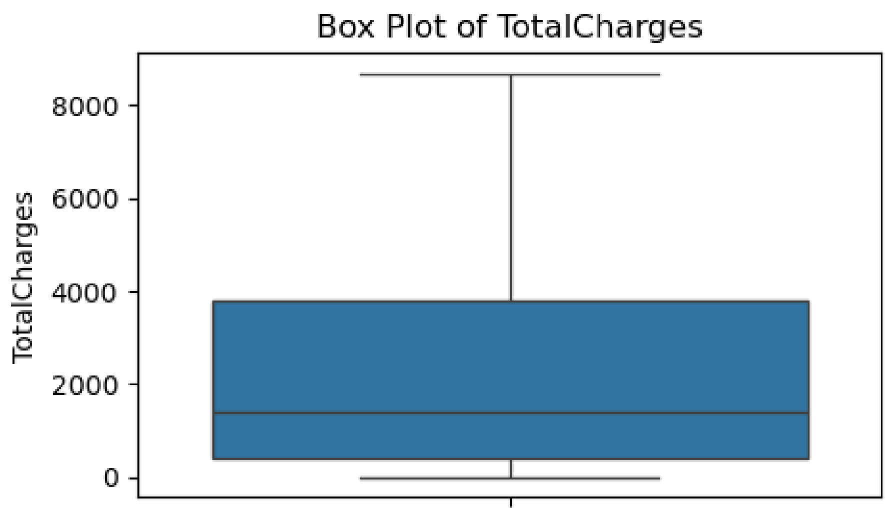
[142]:
```
plot_boxplot(df, "tenure")
```



[143]:
```
plot_boxplot(df, "MonthlyCharges")
```

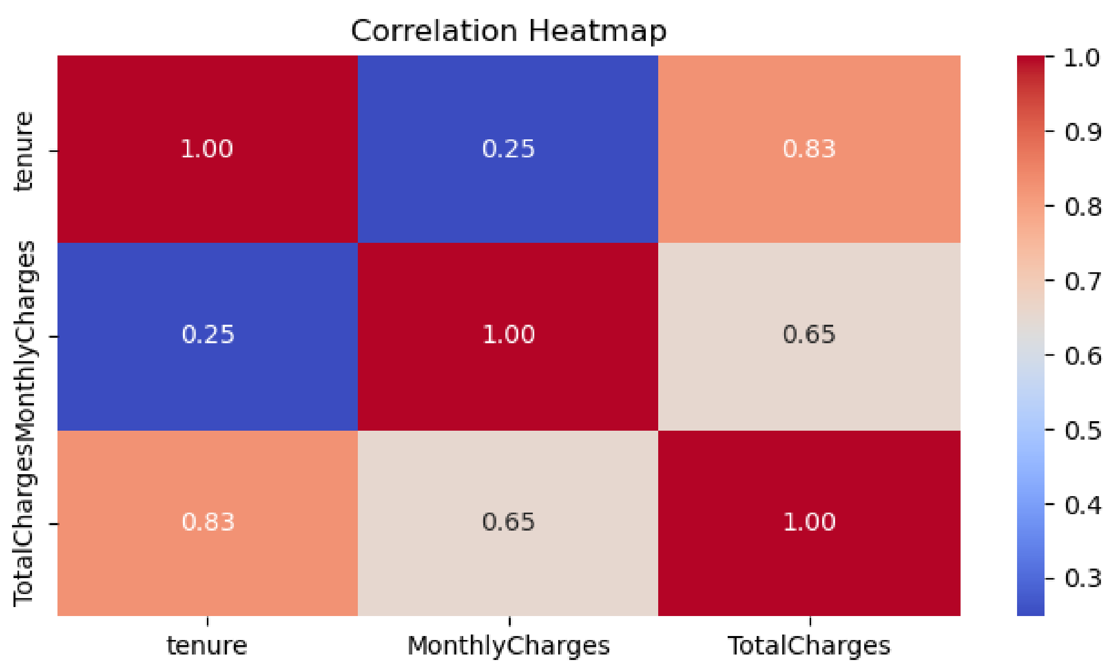


[144]:
```
plot_boxplot(df, "TotalCharges")
```



0.3.4 Correlation Heatmap for numerical columns

[145]:
```
# 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()
```



0.3.5 Categorical features - Analysis

[147]:
```
df.columns
```

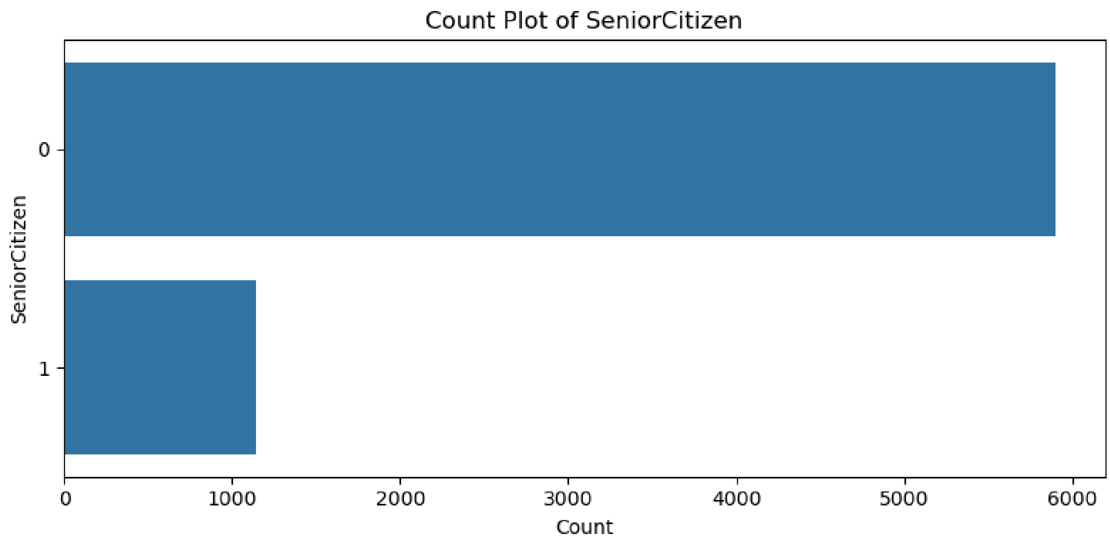
[147]:
```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
       'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

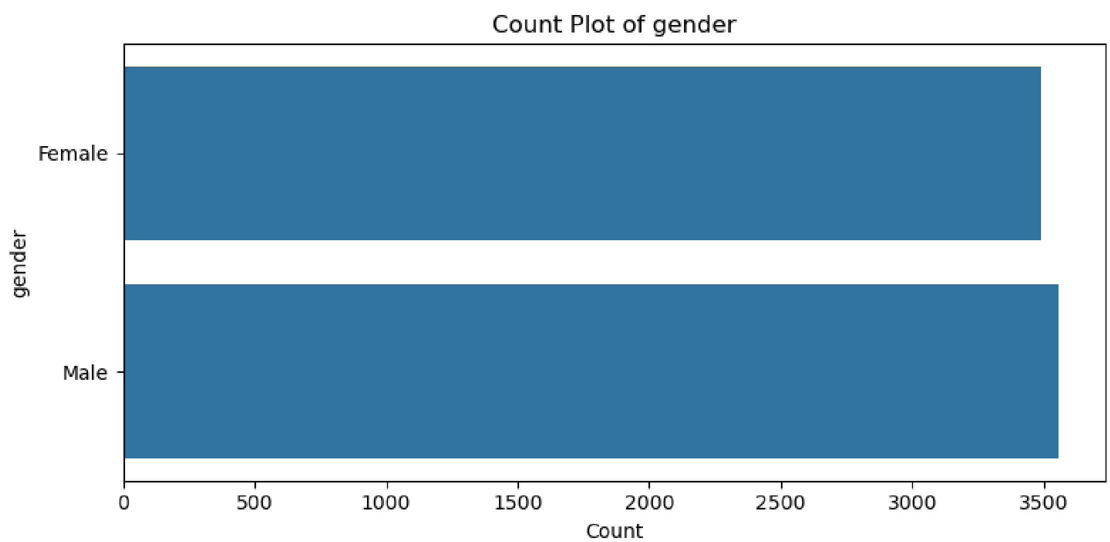
[148]:
```
df.info()
```

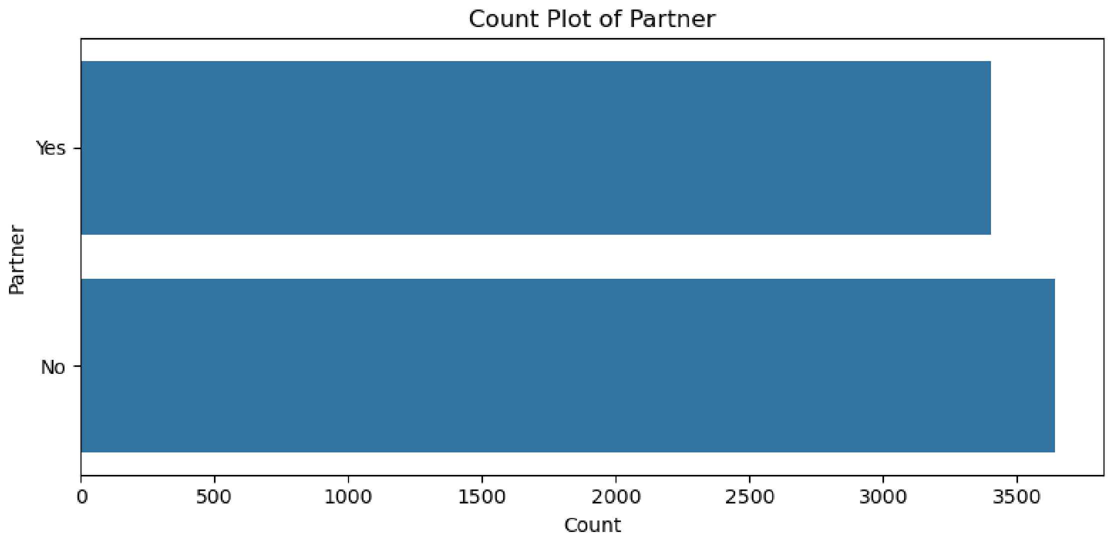
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7043 non-null   object
 1   SeniorCitizen     7043 non-null   int64
 2   Partner           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
```

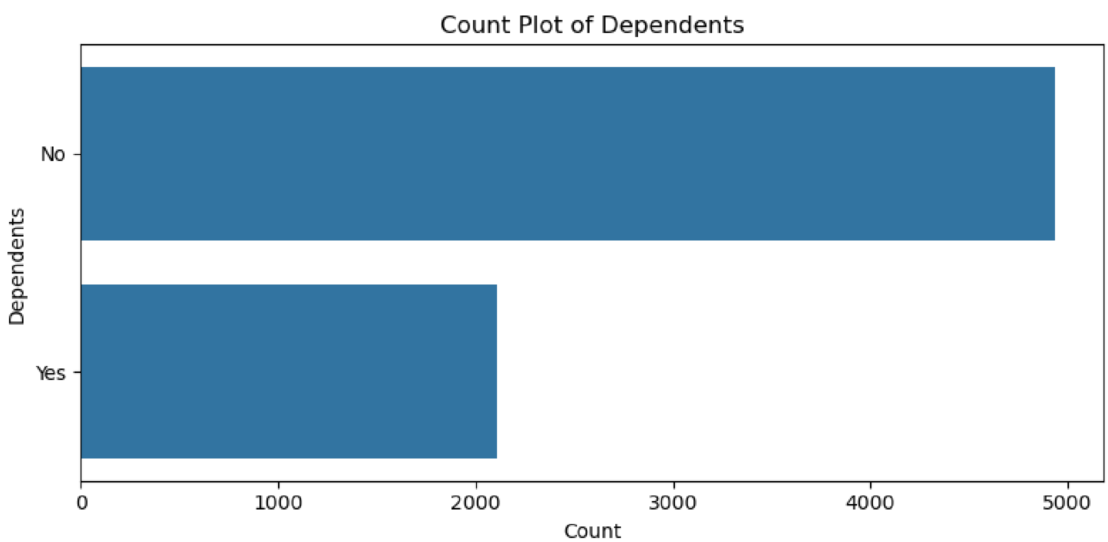
0.3.6 Countplot for categorical columns

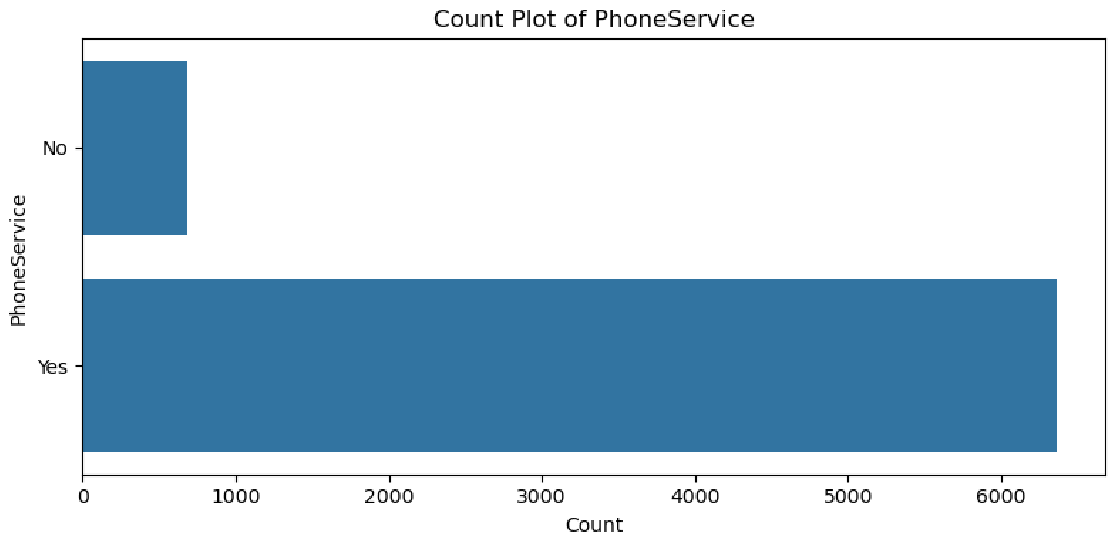
[151]:
```
for col in object_cols:
    plt.figure(figsize=(8, 4))
    sns.countplot(y=df[col])
    plt.title(f"Count Plot of {col}")
    plt.xlabel("Count")
    plt.ylabel(col)
    plt.tight_layout()
    plt.show()
```

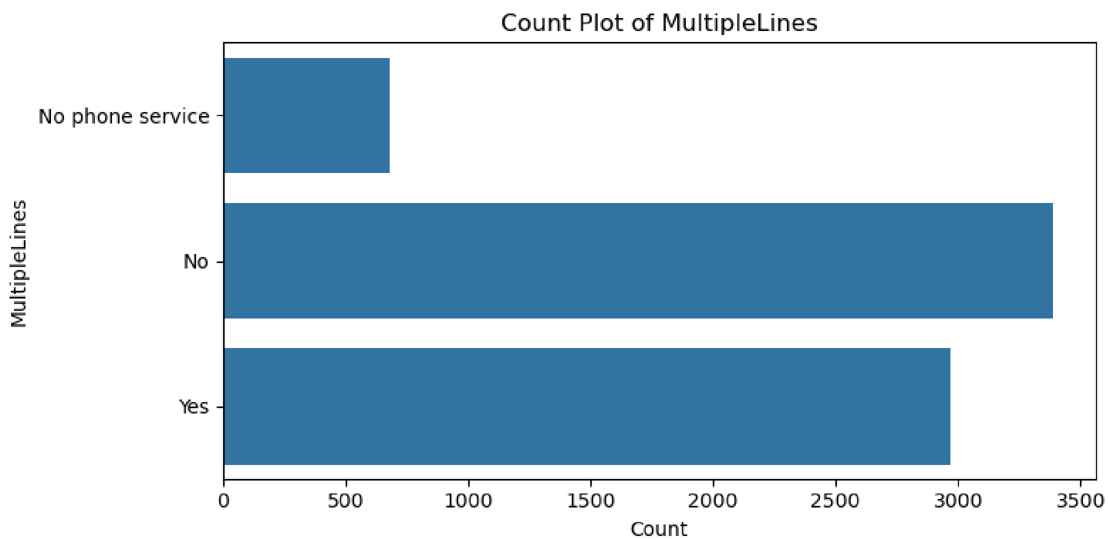


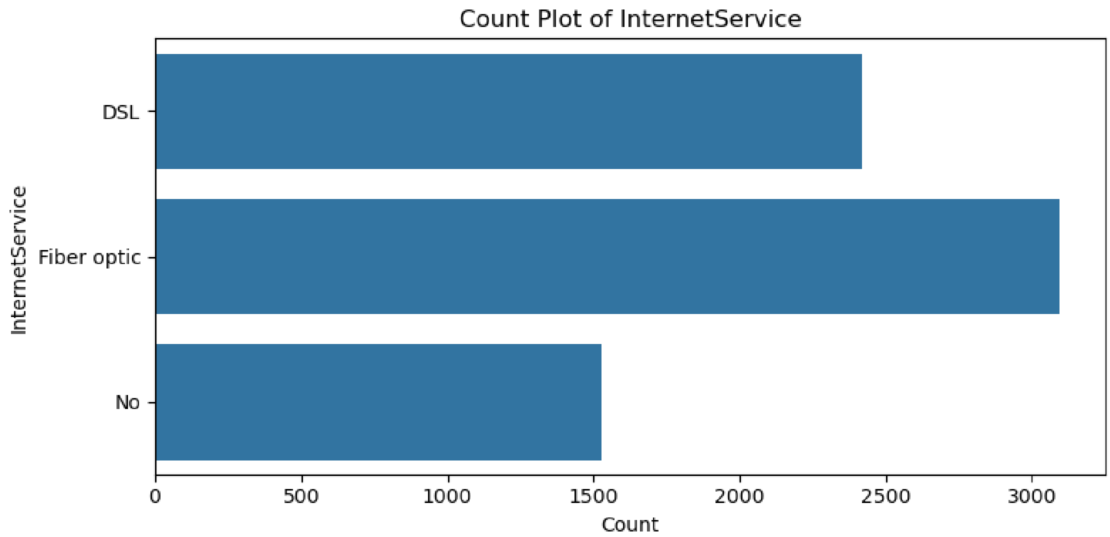



Count Plot of Partner
Yes
No
Partner
0
500
1000
1500
2000
2500
3000
3500
Count

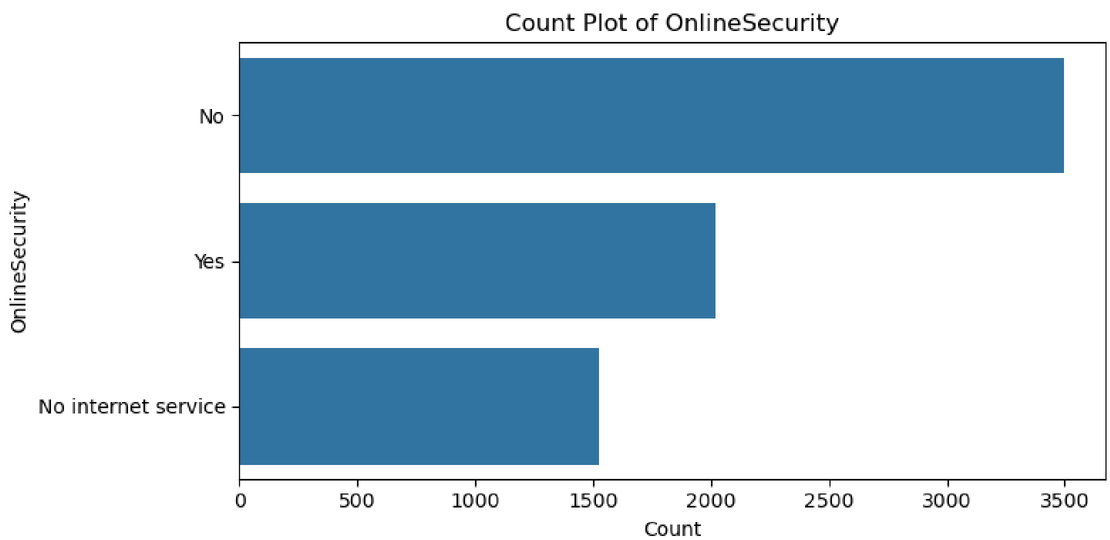

Count Plot of Dependents
No
Yes
Dependents
0
1000
2000
3000
4000
5000
Count


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

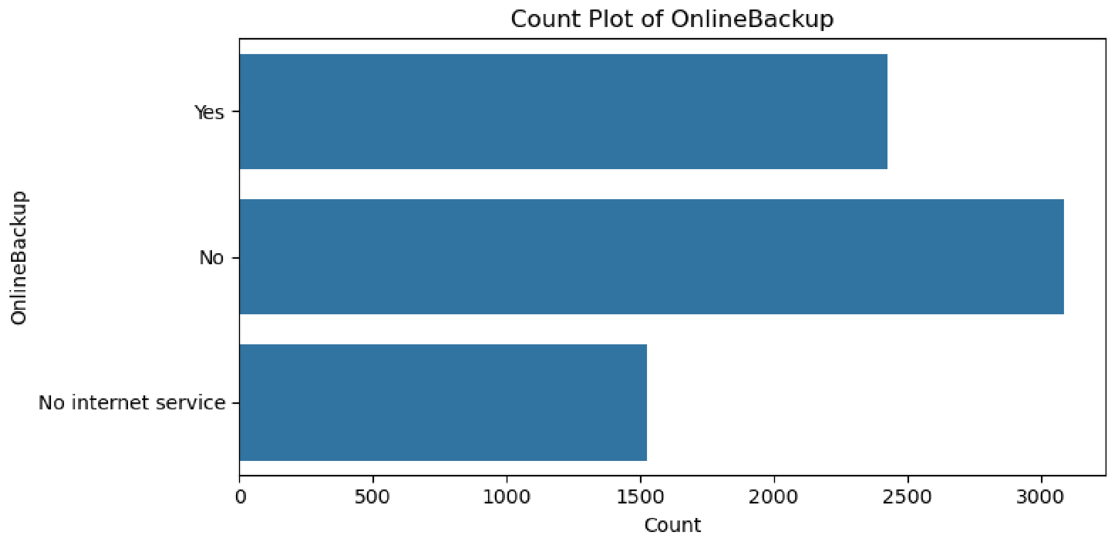

Count Plot of MultipleLines
No phone service
No
Yes
MultipleLines
0
500
1000
1500
2000
2500
3000
3500
Count

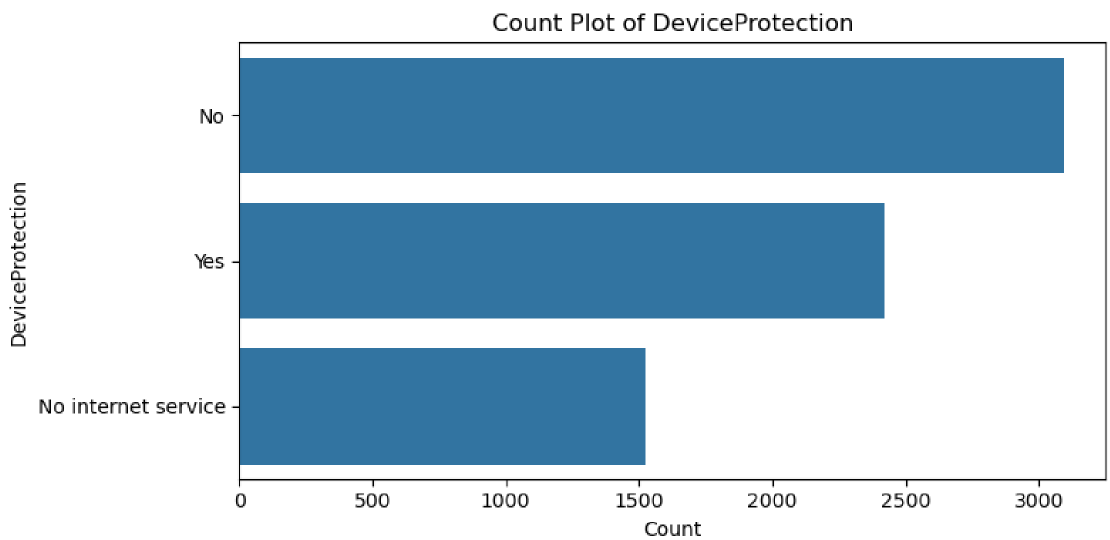


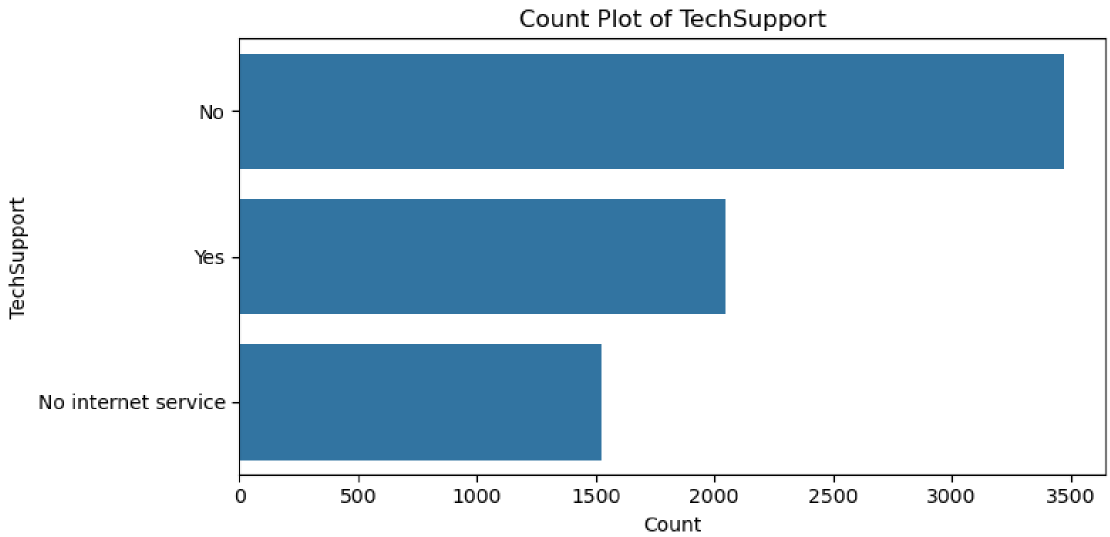
Count Plot of InternetService
DSL
Fiber optic
No
InternetService
0
500
1000
1500
2000
2500
3000
Count

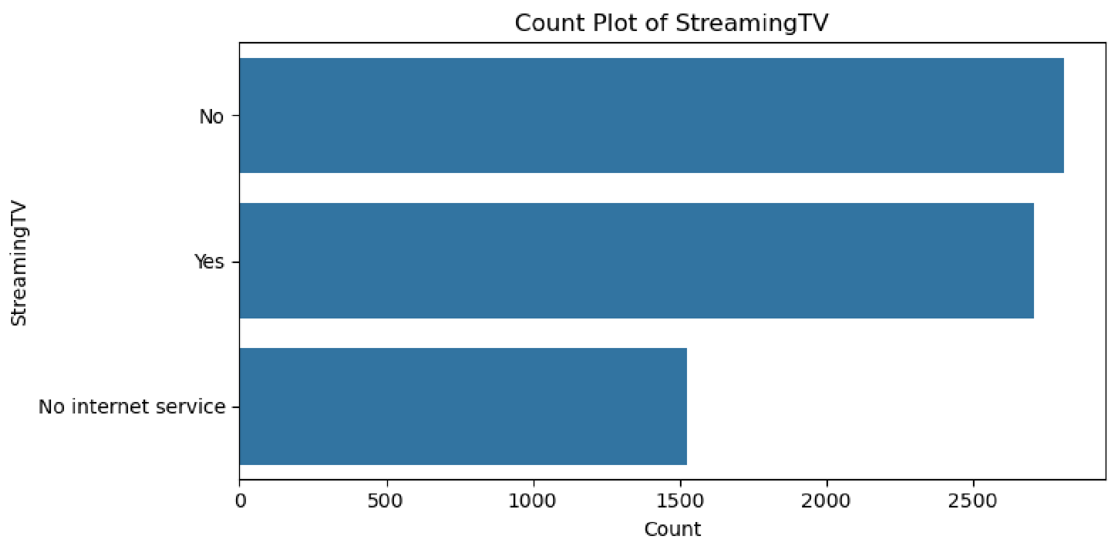


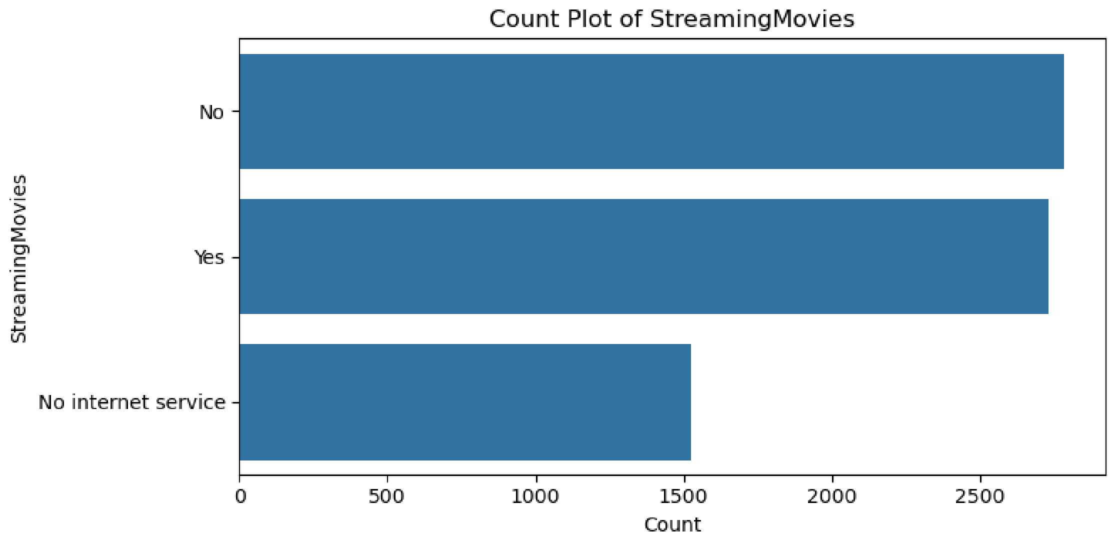
Count Plot of OnlineSecurity
No
Yes
No internet service
OnlineSecurity
0
500
1000
1500
2000
2500
3000
3500
Count

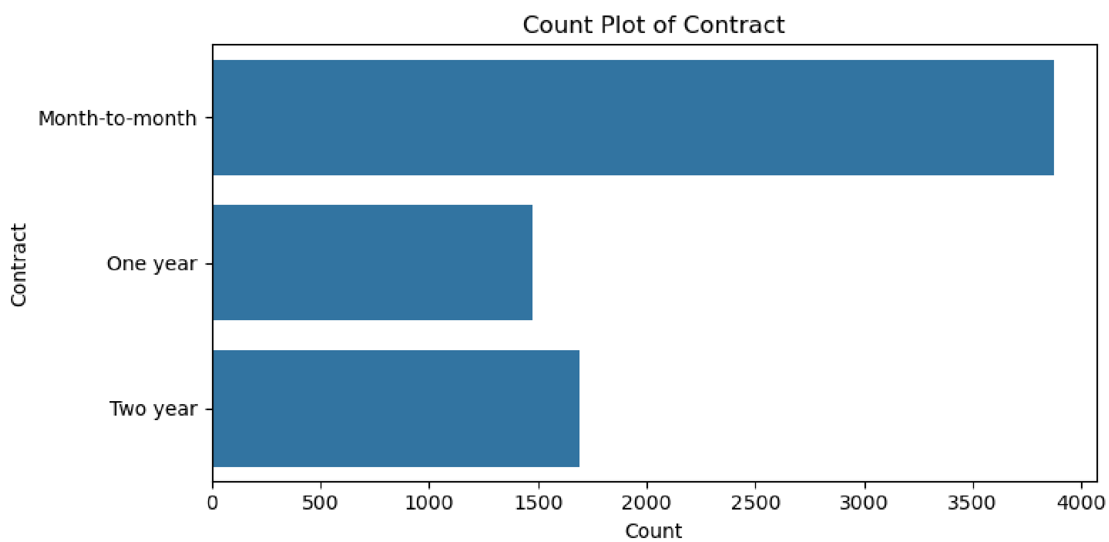

Count Plot of OnlineBackup
Yes
No
No internet service
OnlineBackup
0
500
1000
1500
2000
2500
3000
Count

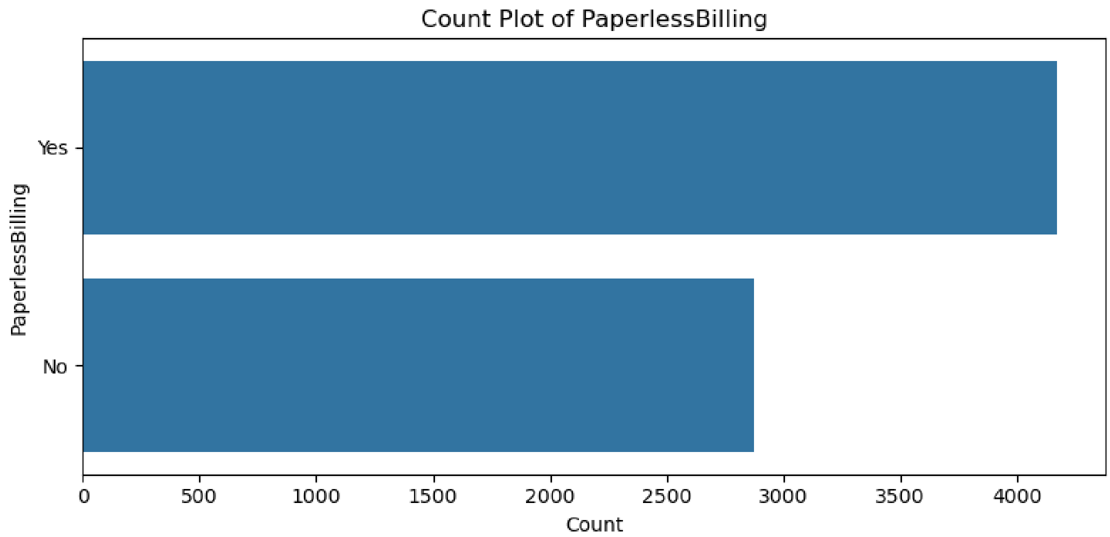

Count Plot of DeviceProtection
No
Yes
No internet service
DeviceProtection
0
500
1000
1500
2000
2500
3000
Count

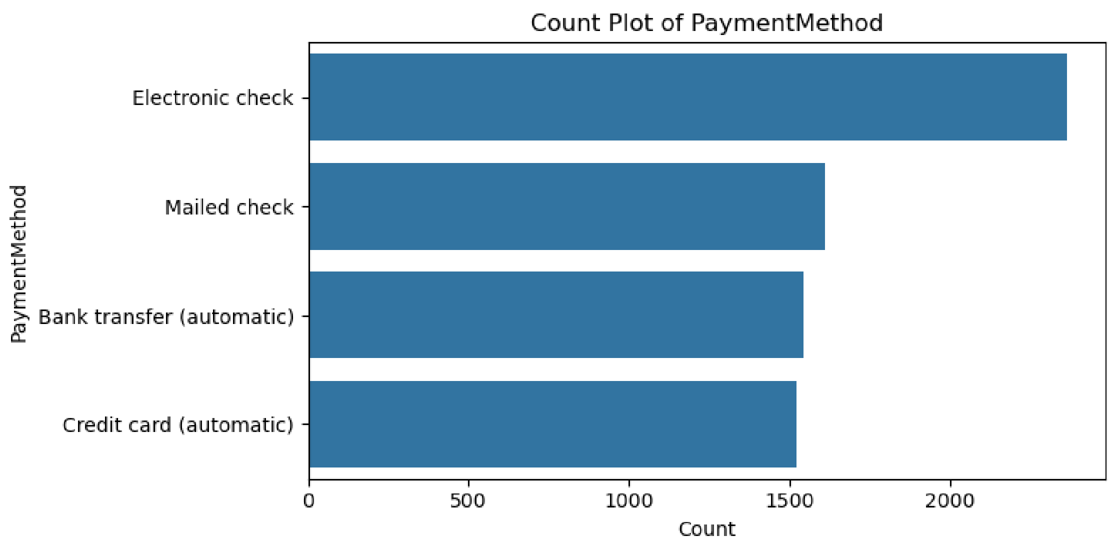

Count Plot of TechSupport
TechSupport
No
Yes
No internet service
0
500
1000
1500
2000
2500
3000
3500
Count

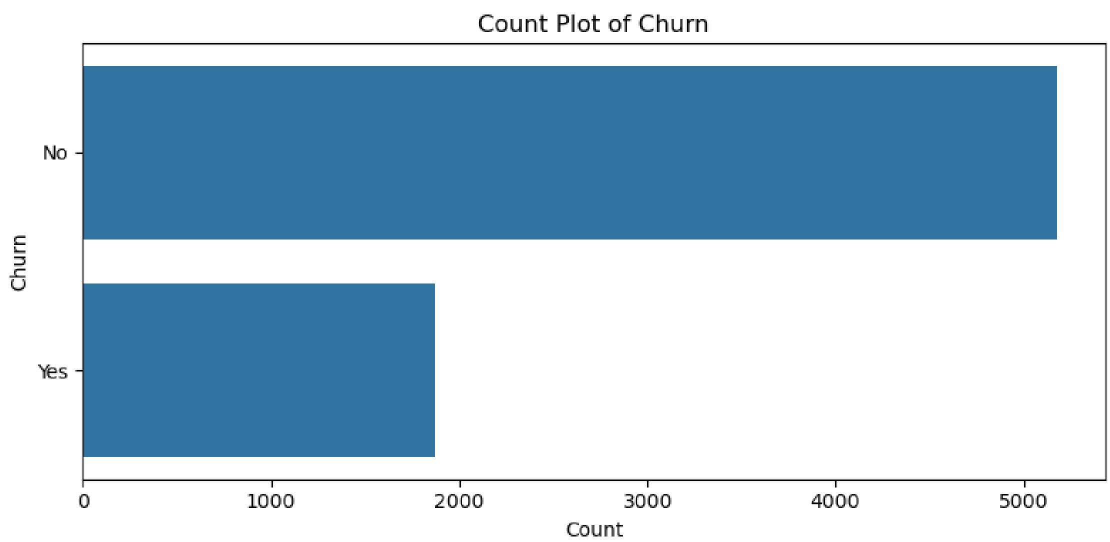

Count Plot of StreamingTV
StreamingTV
No
Yes
No internet service
0
500
1000
1500
2000
2500
Count


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


Count Plot of Contract
Contract
Month-to-month
One year
Two year
0
500
1000
1500
2000
2500
3000
3500
4000
Count


Count Plot of PaperlessBilling
Yes
No
PaperlessBilling
0
500
1000
1500
2000
2500
3000
3500
4000
Count


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



0.4 4. Data Preprocessing

[152]:
```
df.head(3)
```

[152]:
```
   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
```

0.4.1 Label encoding of target column

[153]:
```python
df["Churn"] = df["Churn"].replace({"Yes": 1, "No": 0})
```

```
C:\Users\asusl\AppData\Local\Temp\ipykernel_27480\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})
```

[154]:
```python
df.head()
```

[154]:
```
   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
4  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       1889.50
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      0
1      0
2      1
3      0
4      1
```

[155]:
```
print(df["Churn"].value_counts())
```

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

0.4.2 Label encoding of categorical fetaures

[156]:
```
# identifying columns with object data type
object_columns = df.select_dtypes(include="object").columns
```

[157]:
```
print(object_columns)
```

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

[158]:
```
# 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)
```

[159]:
```
encoders
```

[159]:
```
{'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()}
```

[160]:
```python
df.head()
```

[160]:
```
   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
```

[161]:
```python
df.tail()
```

[161]:
```
      gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  \
7038       1              0        1           1      24             1
7039       0              0        1           1      72             1
7040       0              0        1           1      11             0
7041       1              1        1           0       4             1
7042       1              0        0           0      66             1

      MultipleLines  InternetService  OnlineSecurity  OnlineBackup  \
7038              2                0               2             0
7039              2                1               0             2
```

```
7040                1                 0                2              0
7041                2                 1                0              0
7042                0                 1                2              0

      DeviceProtection  TechSupport  StreamingTV  StreamingMovies  Contract  \
7038                 2            2            2                2         1
7039                 2            0            2                2         1
7040                 0            0            0                0         0
7041                 0            0            0                0         0
7042                 2            2            2                2         2

      PaperlessBilling  PaymentMethod  MonthlyCharges  TotalCharges  Churn
7038                 1              3           84.80       1990.50      0
7039                 1              1          103.20       7362.90      0
7040                 1              2           29.60        346.45      0
7041                 1              3           74.40        306.60      1
7042                 1              0          105.65       6844.50      0
```

0.4.3 Traianing and test data split

[162]:
```python
# splitting the features and target
X = df.drop(columns=["Churn"])
y = df["Churn"]
```

[163]:
```python
# 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)
```

[164]:
```python
print(y_train.shape)
```

```
(5634,)
```

[165]:
```python
print(y_train.value_counts())
```

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

0.4.4 Synthetic Minority Oversampling TEchnique (SMOTE)

[166]:
```python
smote = SMOTE(random_state=42)
```

[167]:
```python
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

[168]:
```python
print(y_train_smote.shape)
```

```
(8276,)
```

[169]:
```
print(y_train_smote.value_counts())
```

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

0.5 5. Model Training

0.5.1 Training with default hyperparameters

[170]:
```
# dictionary of models
models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42)
}
```

[171]:
```
# 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
----------------------------------------------------------------------
```

[172]:
```
cv_scores
```

[172]:
```
{'Decision Tree': array([0.68115942, 0.71903323, 0.81752266, 0.84350453,
0.84350453]),
 'Random Forest': array([0.72705314, 0.76676737, 0.90453172, 0.89244713,
0.89848943]),
```

```
'XGBoost': array([0.71074879, 0.75226586, 0.90271903, 0.89123867, 0.89909366])}
```

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

[173]:
```
rfc = RandomForestClassifier(random_state=42)
```

[174]:
```
rfc.fit(X_train_smote, y_train_smote)
```

[174]:
```
RandomForestClassifier(random_state=42)
```

[175]:
```
print(y_test.value_counts())
```

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

0.6 6. Model Evaluation

[176]:
```
# 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:
 [[880 156]
 [158 215]]
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
```

[177]:
```
# save the trained model as a pickle file
model_data = {"model": rfc, "features_names": X.columns.tolist()}
```

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

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

[178]:
```python
# 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"]
```

[179]:
```python
print(loaded_model)
```

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

[180]:
```python
print(feature_names)
```

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

[181]:
```python
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: [[0.83 0.17]]
```

0.8 Conclusion

This project demonstrates an end-to-end machine learning workflow for predicting customer churn.

Through effective data analysis, preprocessing, and model building, meaningful churn patterns were identified.

The final model provides actionable insights that can help businesses improve customer retention.

Overall, this project reflects practical application of machine learning in a real-world business scenario.

[ ]:
```

```