telco

June 18, 2024

[47]:
```
# Abdurrahman Bulut
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
```

[2]:
```
df = pd.read_csv("Telco-Customer-Churn.csv")
```

[4]:
```
df.head()
```

[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling
0	7590-VHVEG	Female	0	Yes	No	1	No	…	No					
1	5575-GNVDE	Male	0	No	No	34	Yes	…	Yes					
2	3668-QPYBK	Male	0	No	No	2	Yes	…	No					
3	7795-CFOCW	Male	0	No	No	45	No	…	Yes					
4	9237-HQITU	Female	0	No	No	2	Yes	…	No					

	MultipleLines	InternetService	OnlineSecurity	…	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling
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					

```
 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]
```

[5]:
```
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
```

Adım 1

[6]:
```
# Numerik değişkenler
numerik_degiskenler = df.select_dtypes(include=['int64', 'float64']).columns.
 ↪tolist()

# Kategorik değişkenler
kategorik_degiskenler = df.select_dtypes(include=['object']).columns.tolist()

print("Numerik Değişkenler:", numerik_degiskenler)
print("Kategorik Değişkenler:", kategorik_degiskenler)
```

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

Adım 2

[7]:
```
# TotalCharges kolonunun tipini kontrol edelim
df['TotalCharges'].dtype
```

[7]:
```
dtype('O')
```

[8]:
```
# TotalCharges kolonunun tipini float yapalım
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

[9]:
```
# Null değerleri kontrol edelim
df.isnull().sum()

# Null değerleri içeren satırları silelim
df = df.dropna()

# Tekrar veri tiplerini kontrol edelim
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7032 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7032 non-null   object
 1   gender            7032 non-null   object
 2   SeniorCitizen     7032 non-null   int64
 3   Partner           7032 non-null   object
 4   Dependents        7032 non-null   object
 5   tenure            7032 non-null   int64
 6   PhoneService      7032 non-null   object
```

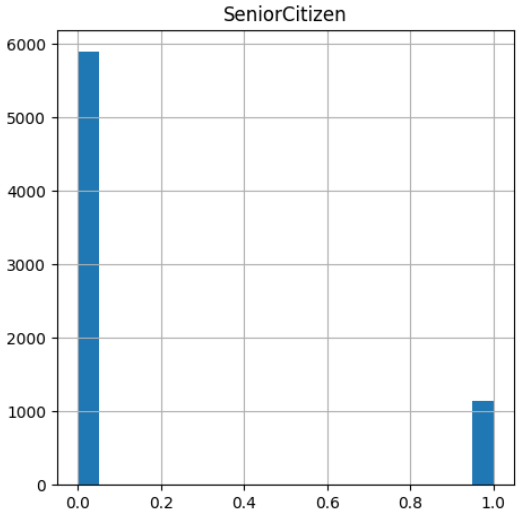
```
 7   MultipleLines     7032 non-null   object
 8   InternetService   7032 non-null   object
 9   OnlineSecurity    7032 non-null   object
 10  OnlineBackup      7032 non-null   object
 11  DeviceProtection  7032 non-null   object
 12  TechSupport       7032 non-null   object
 13  StreamingTV       7032 non-null   object
 14  StreamingMovies   7032 non-null   object
 15  Contract          7032 non-null   object
 16  PaperlessBilling  7032 non-null   object
 17  PaymentMethod     7032 non-null   object
 18  MonthlyCharges    7032 non-null   float64
 19  TotalCharges      7032 non-null   float64
 20  Churn             7032 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.2+ MB
```

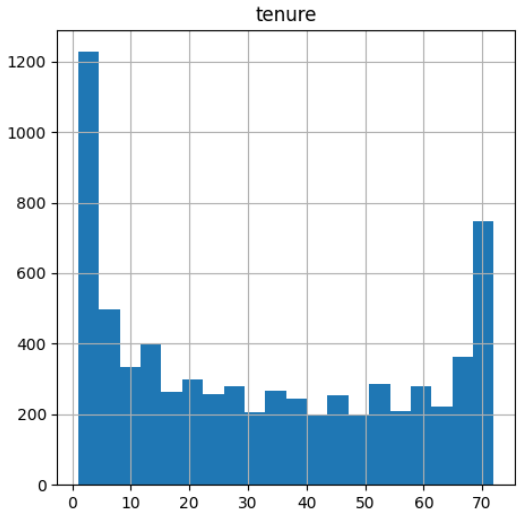
Adım 3

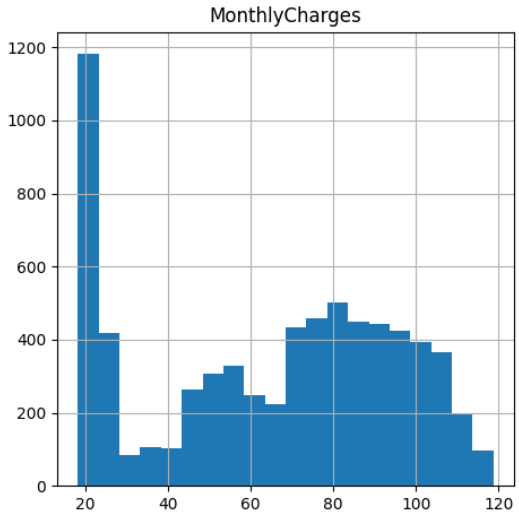
[10]:
```
# Numerik değişkenlerin dağılımı
print(df[numerik_degiskenler].describe())

# Histogram çizelim
df[numerik_degiskenler].hist(figsize=(12, 12), bins=20)
plt.show()
```

```
       SeniorCitizen       tenure  MonthlyCharges
count    7032.000000  7032.000000     7032.000000
mean        0.162400    32.421786       64.798208
std         0.368844    24.545260       30.085974
min         0.000000     1.000000       18.250000
25%         0.000000     9.000000       35.587500
50%         0.000000    29.000000       70.350000
75%         0.000000    55.000000       89.862500
max         1.000000    72.000000      118.750000
```



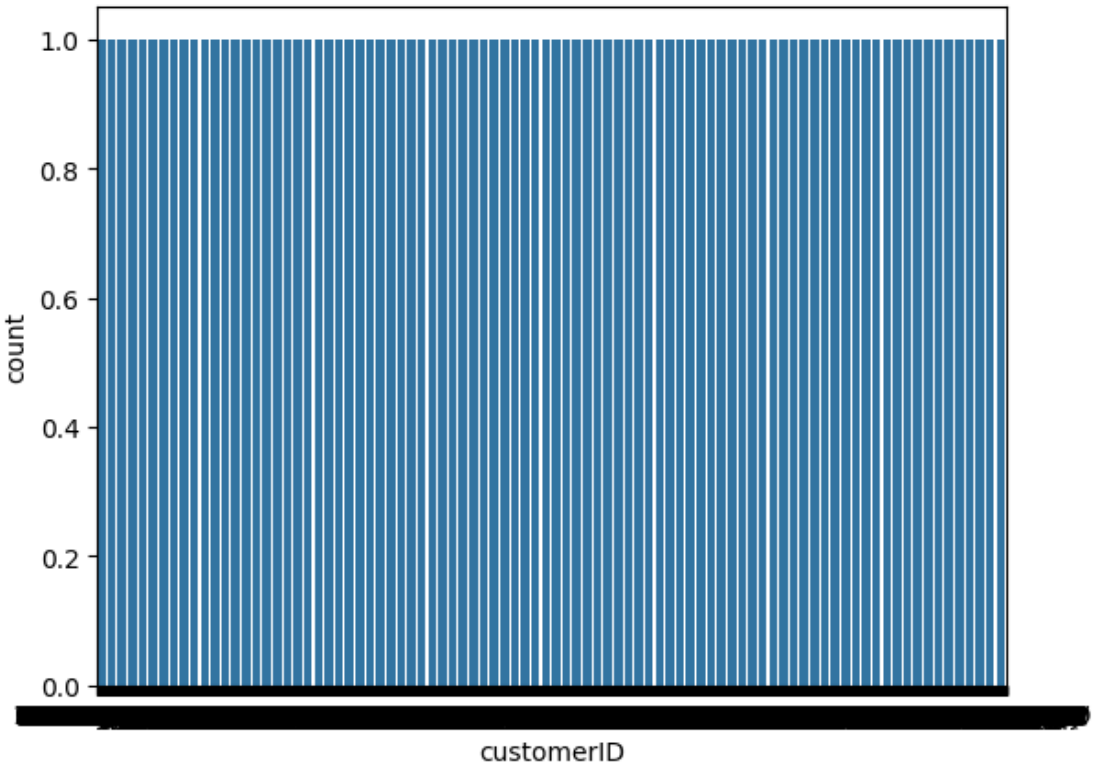




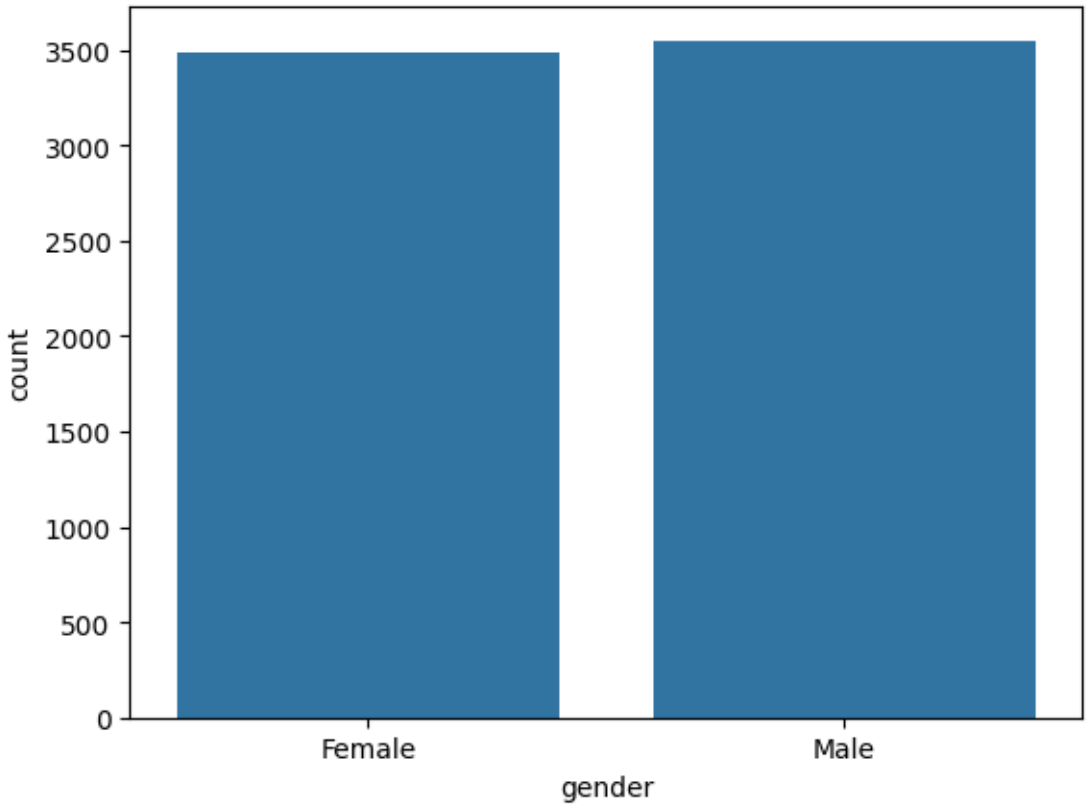
[11]:
```
# Kategorik değişkenlerin dağılımı
for col in kategorik_degiskenler:
    print(df[col].value_counts())
    sns.countplot(x=col, data=df)
    plt.show()
```

```
customerID
7590-VHVEG    1
0265-PSUAE    1
2956-GGUCQ    1
6008-NAIXK    1
5956-YHHRX    1
             ..
```

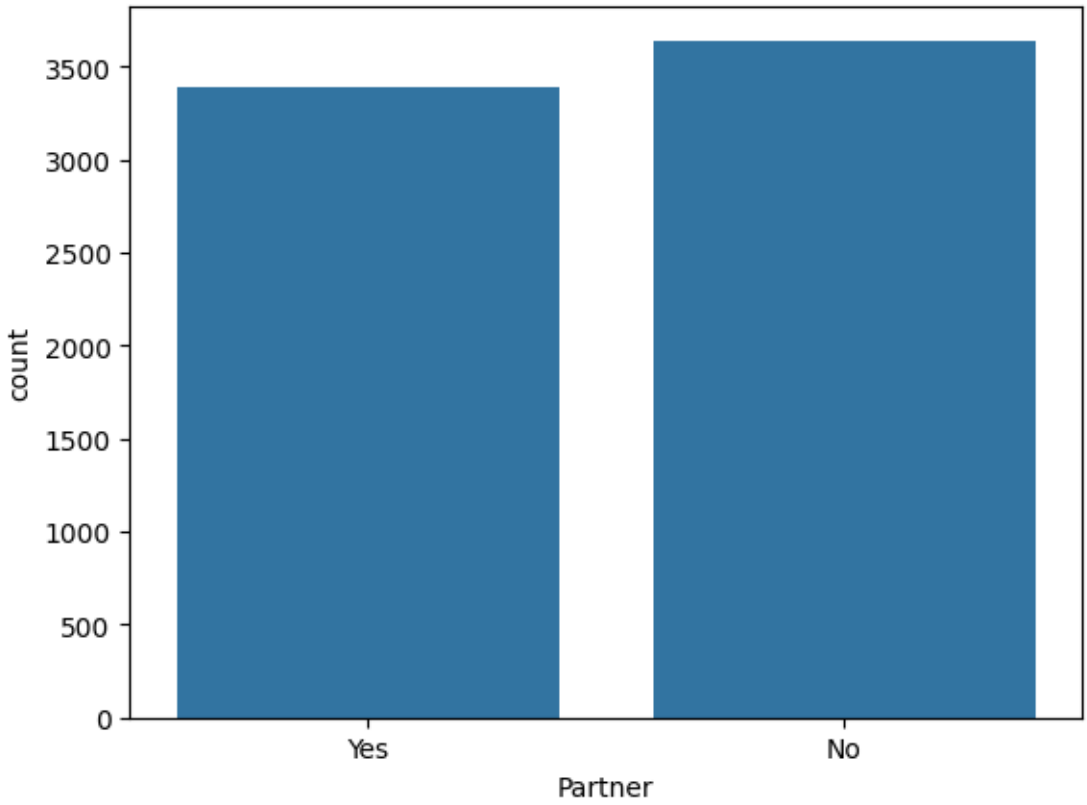
```
7874-ECPQJ    1
9796-MVYXX    1
2637-FKFSY    1
1552-AAGRX    1
3186-AJIEK    1
Name: count, Length: 7032, dtype: int64
```



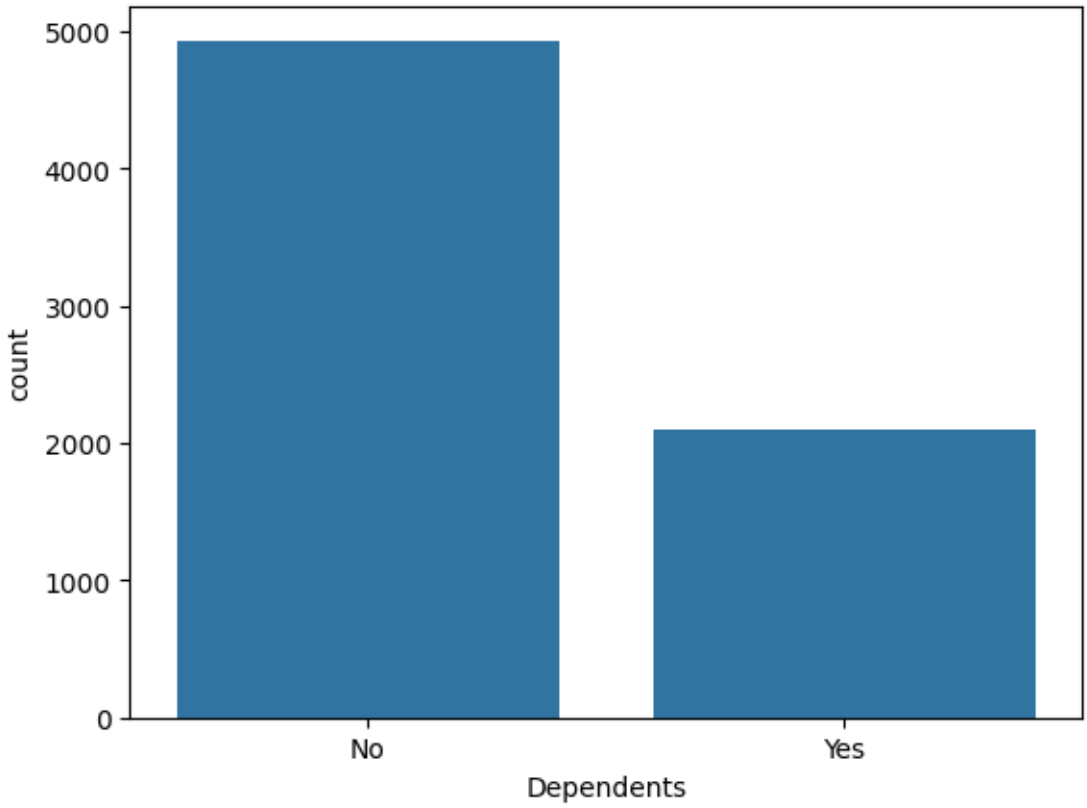

```
gender
Male      3549
Female    3483
Name: count, dtype: int64
```



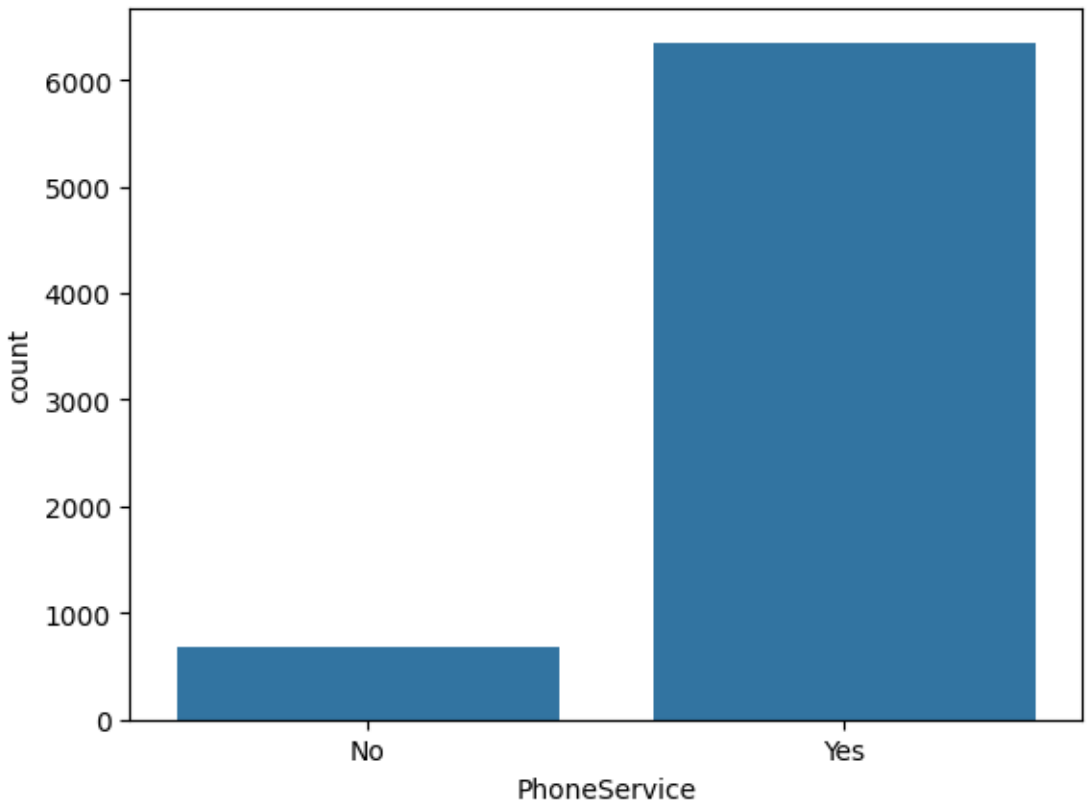

```
Partner
No     3639
Yes    3393
Name: count, dtype: int64
```



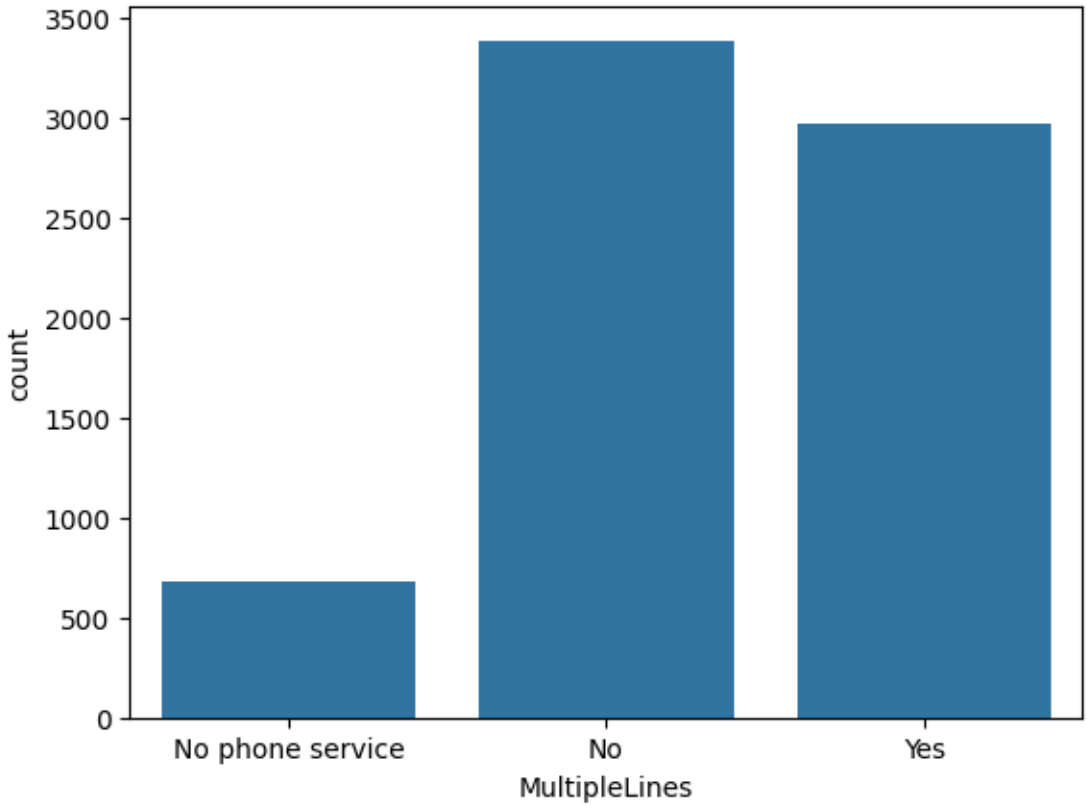

```
Dependents
No     4933
Yes    2099
Name: count, dtype: int64
```



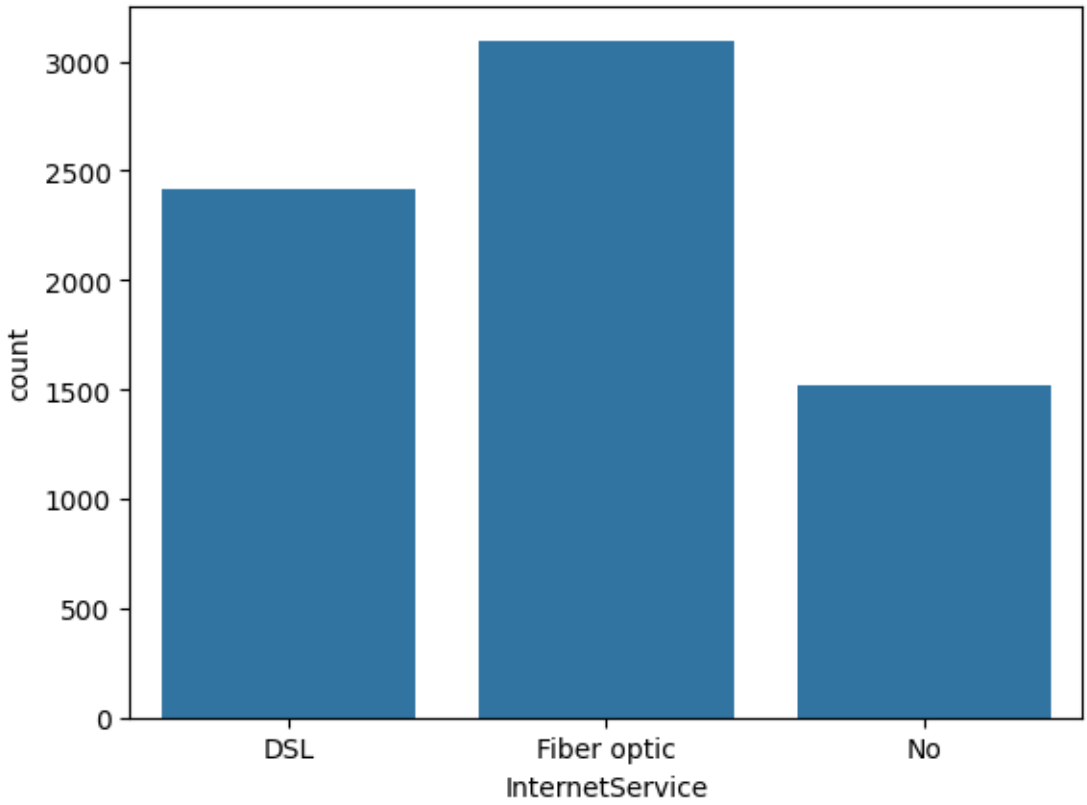

```
PhoneService
Yes    6352
No      680
Name: count, dtype: int64
```



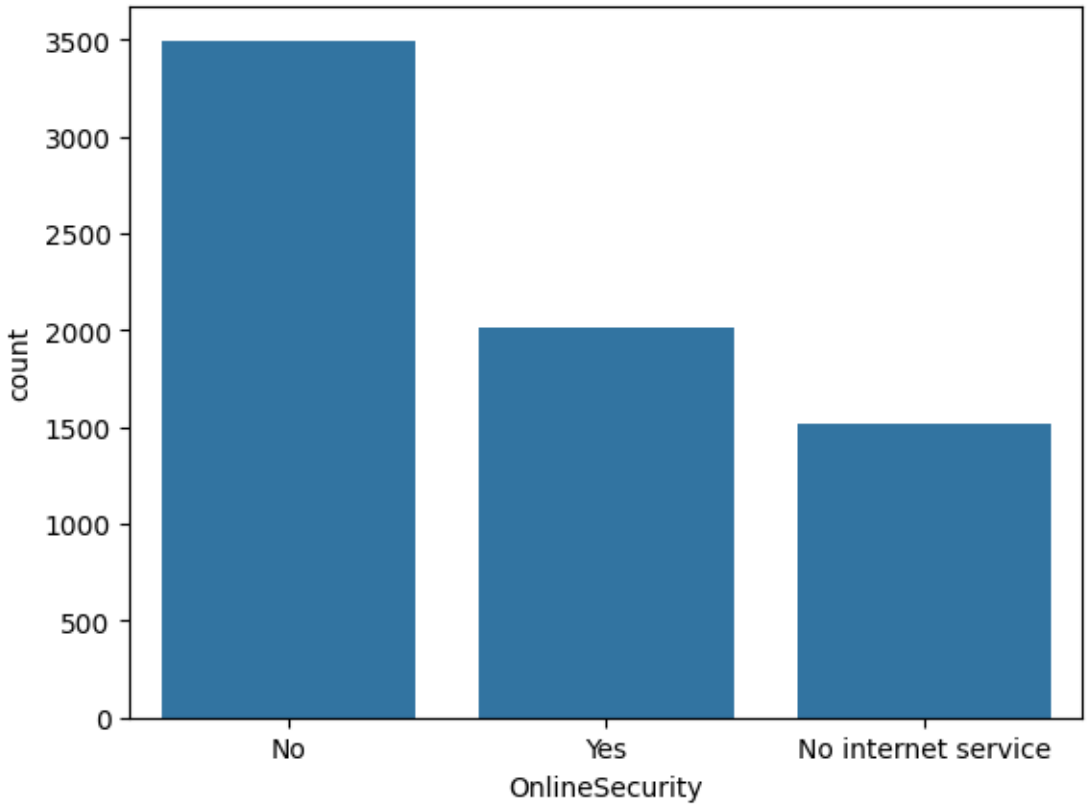

```
MultipleLines
No                  3385
Yes                 2967
No phone service     680
Name: count, dtype: int64
```



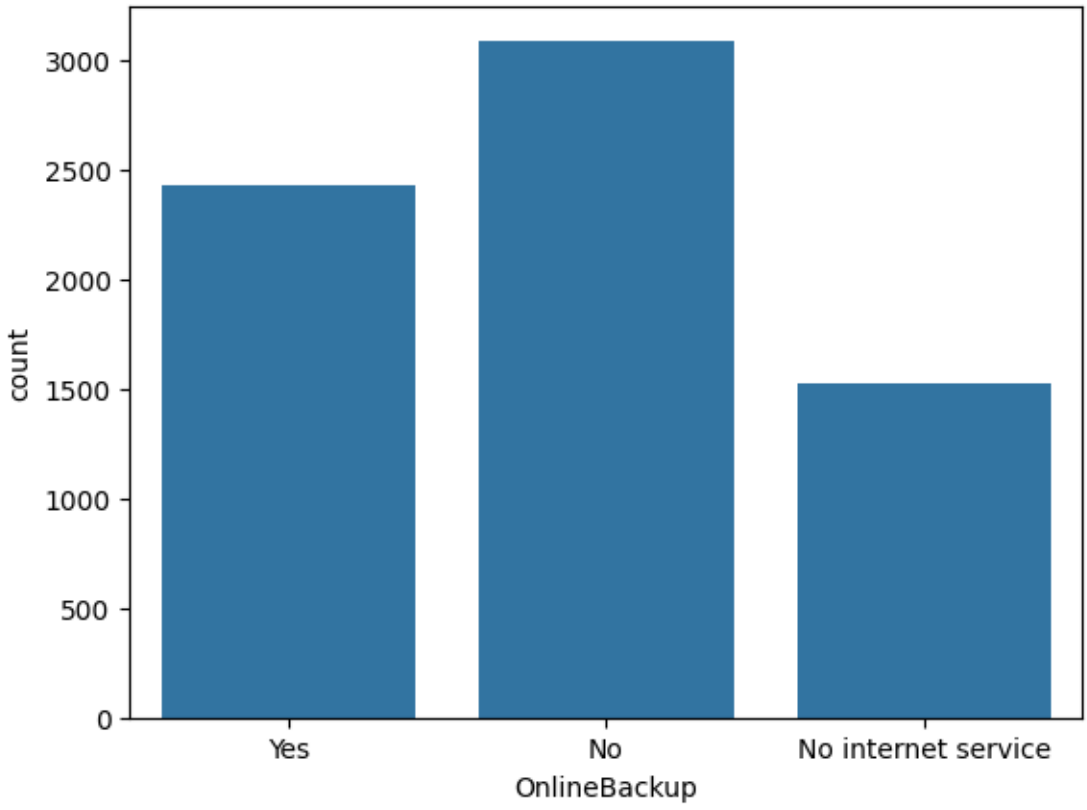

```
InternetService
Fiber optic    3096
DSL            2416
No             1520
Name: count, dtype: int64
```



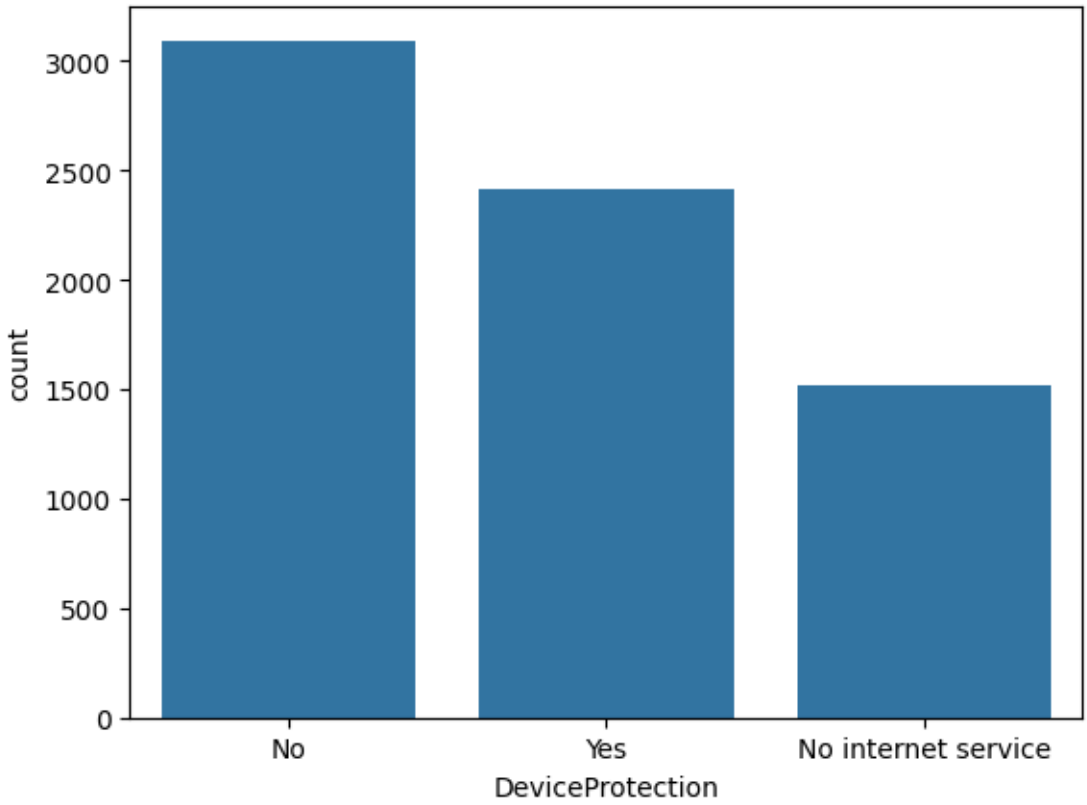

```
OnlineSecurity
No                     3497
Yes                    2015
No internet service    1520
Name: count, dtype: int64
```



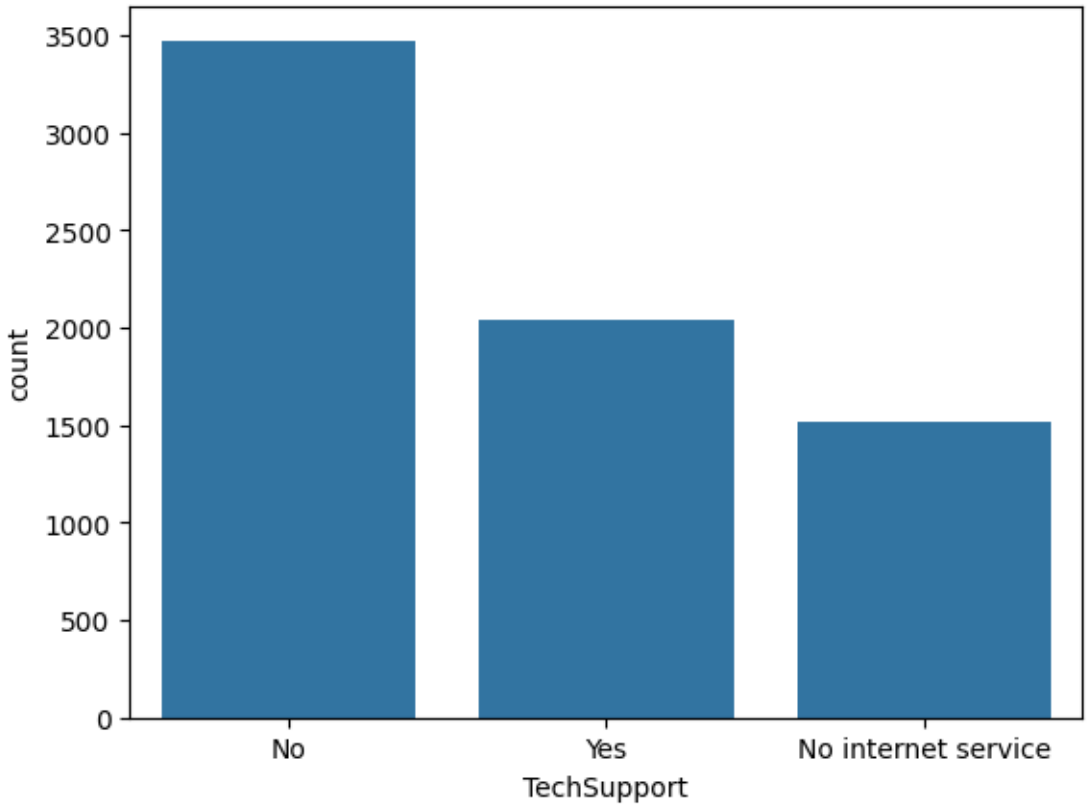

```
OnlineBackup
No                     3087
Yes                    2425
No internet service    1520
Name: count, dtype: int64
```



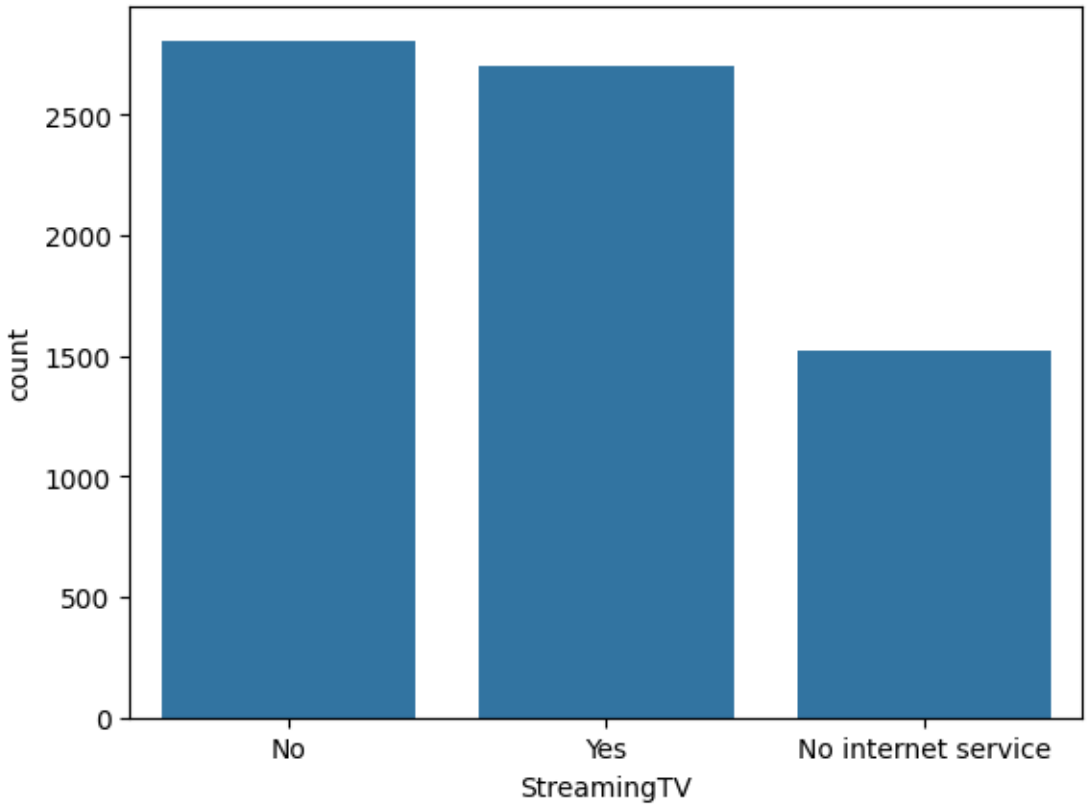

```
DeviceProtection
No                     3094
Yes                    2418
No internet service    1520
Name: count, dtype: int64
```



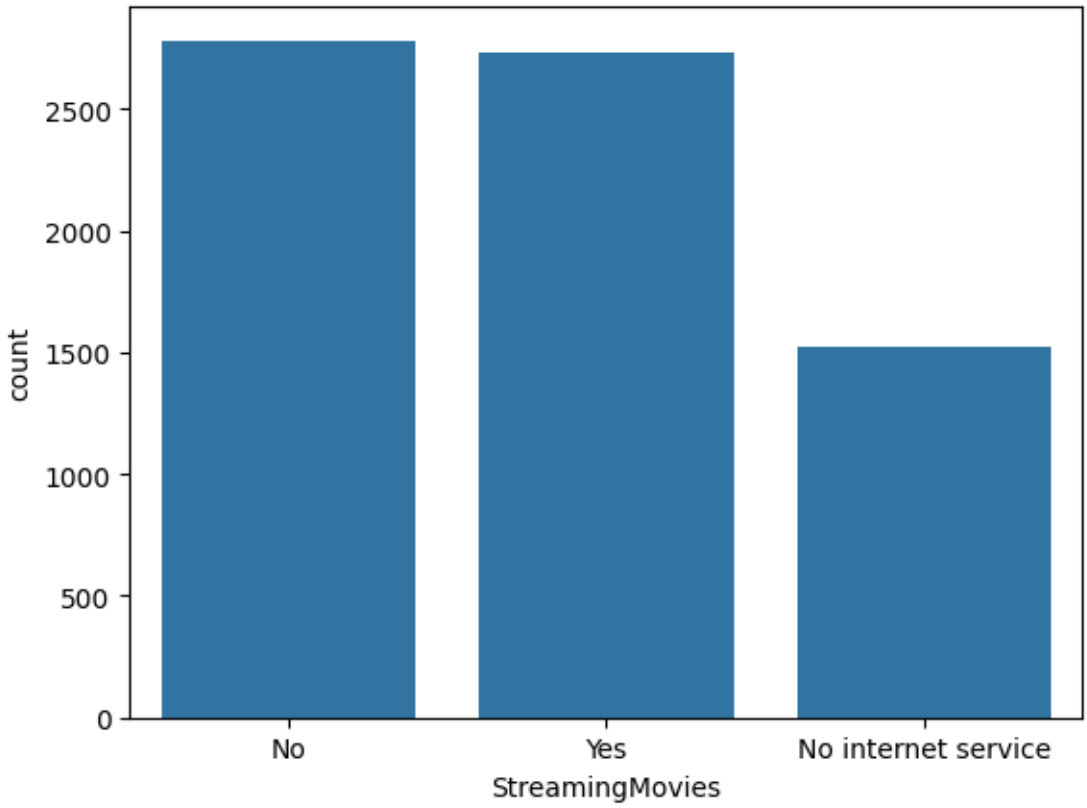

```
TechSupport
No                     3472
Yes                    2040
No internet service    1520
Name: count, dtype: int64
```



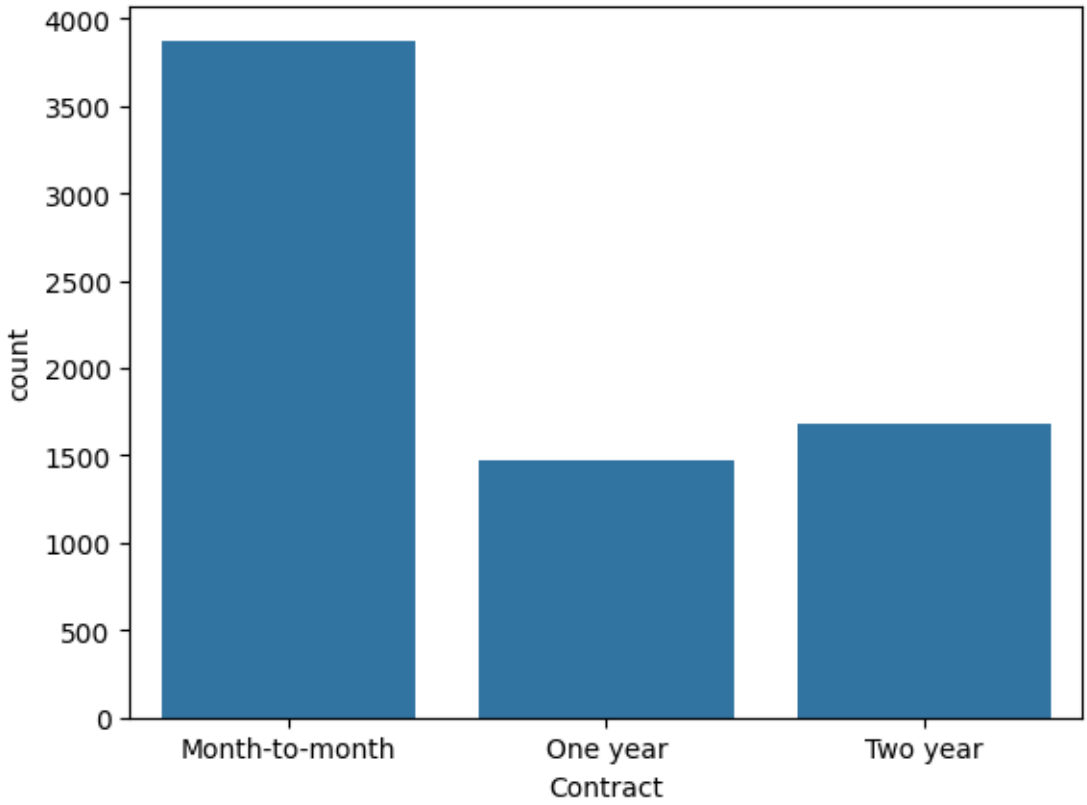

```
StreamingTV
No                     2809
Yes                    2703
No internet service    1520
Name: count, dtype: int64
```



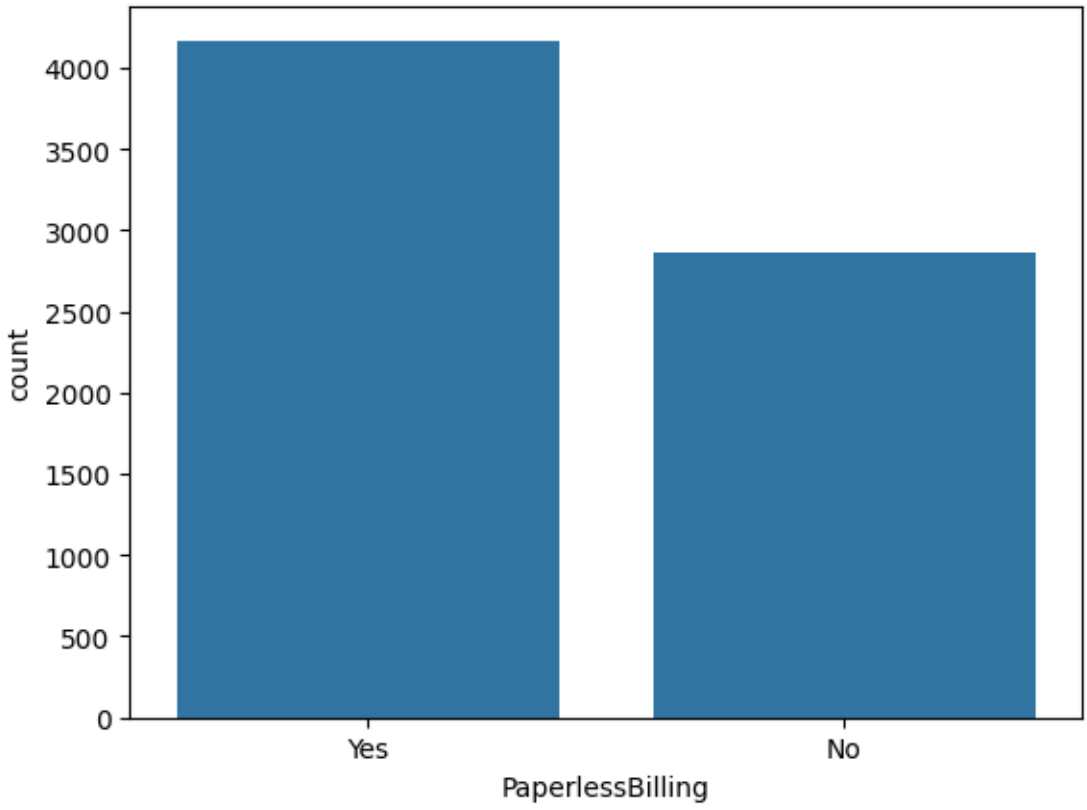

```
StreamingMovies
No                     2781
Yes                    2731
No internet service    1520
Name: count, dtype: int64
```



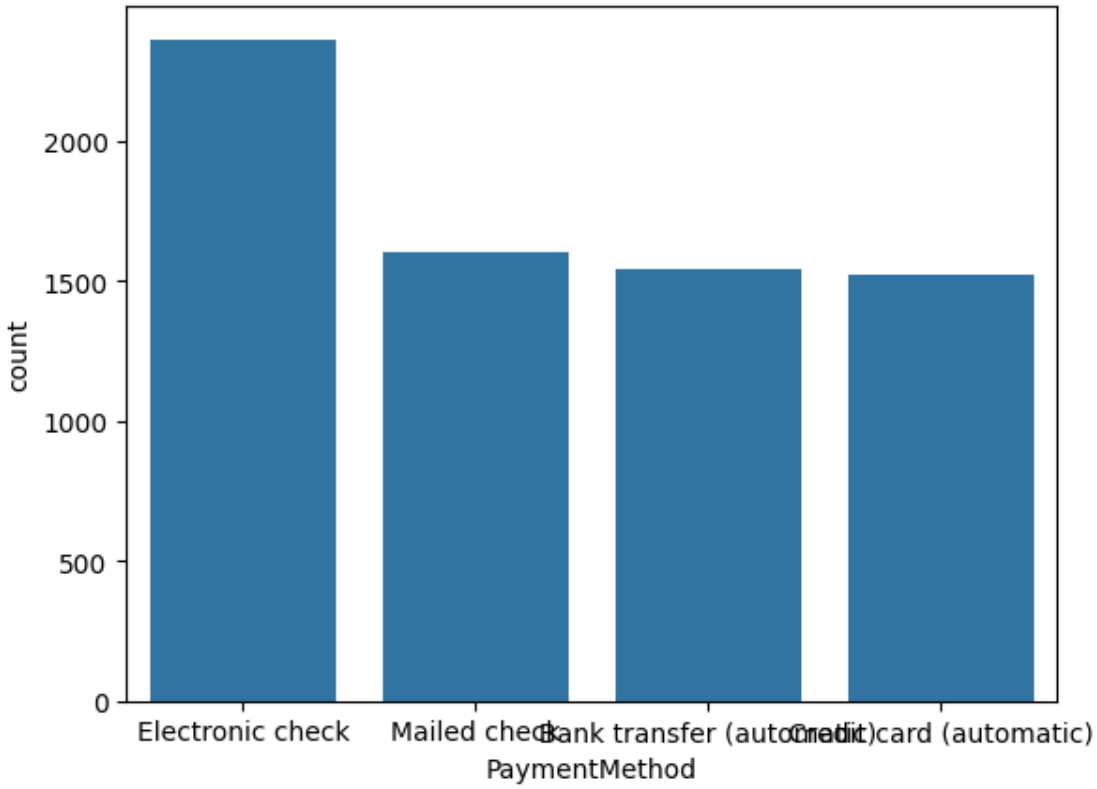

```
Contract
Month-to-month    3875
Two year          1685
One year          1472
Name: count, dtype: int64
```



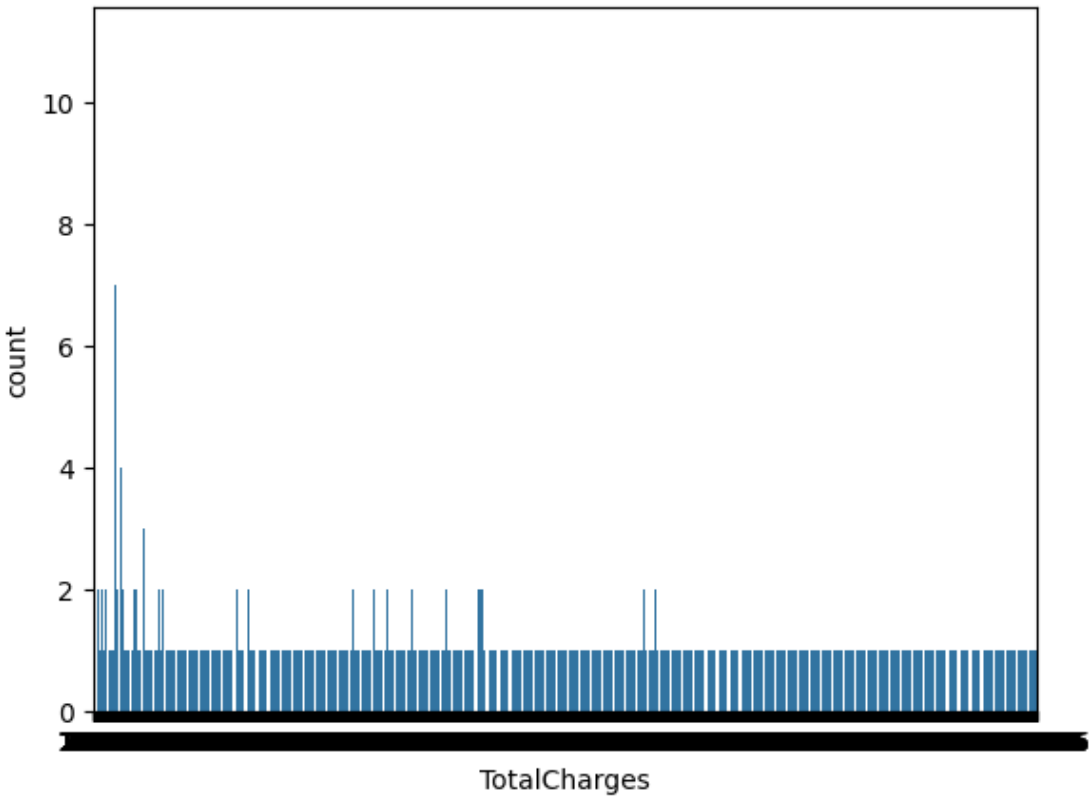

```
PaperlessBilling
Yes    4168
No     2864
Name: count, dtype: int64
```



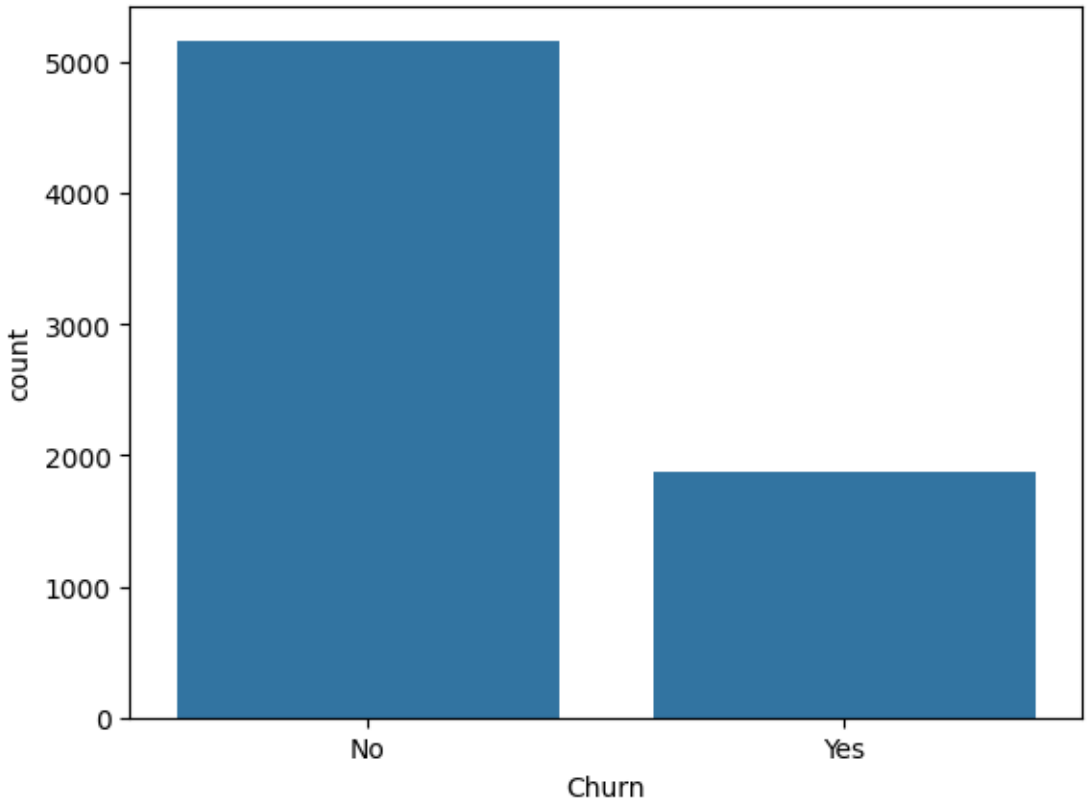

```
PaymentMethod
Electronic check             2365
Mailed check                 1604
Bank transfer (automatic)    1542
Credit card (automatic)      1521
Name: count, dtype: int64
```



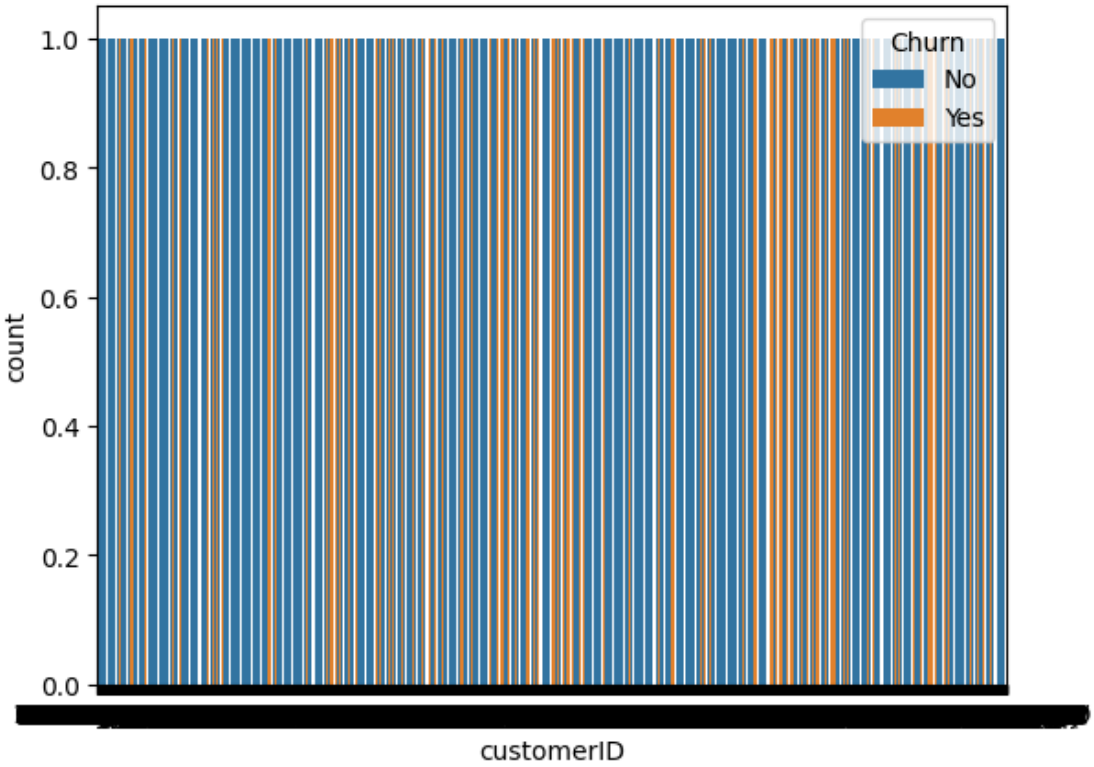

```
TotalCharges
20.20      11
19.75       9
20.05       8
19.90       8
19.65       8
           ..
6849.40     1
692.35      1
130.15      1
3211.90     1
6844.50     1
Name: count, Length: 6530, dtype: int64
```

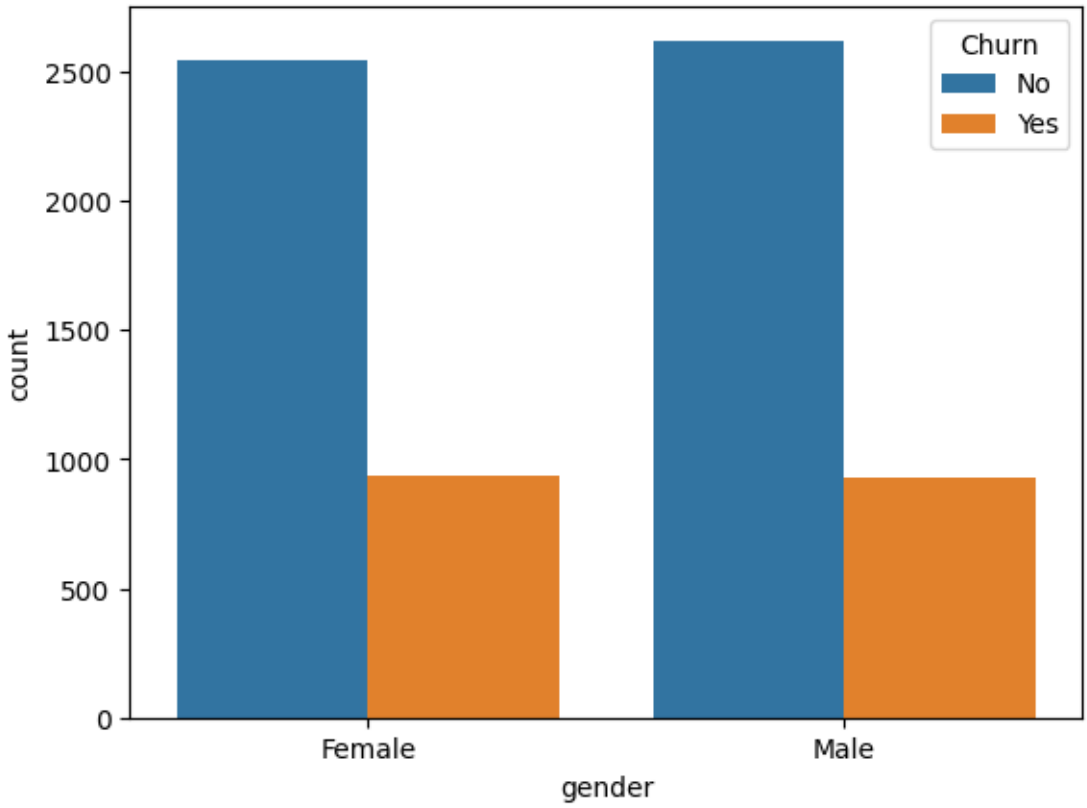


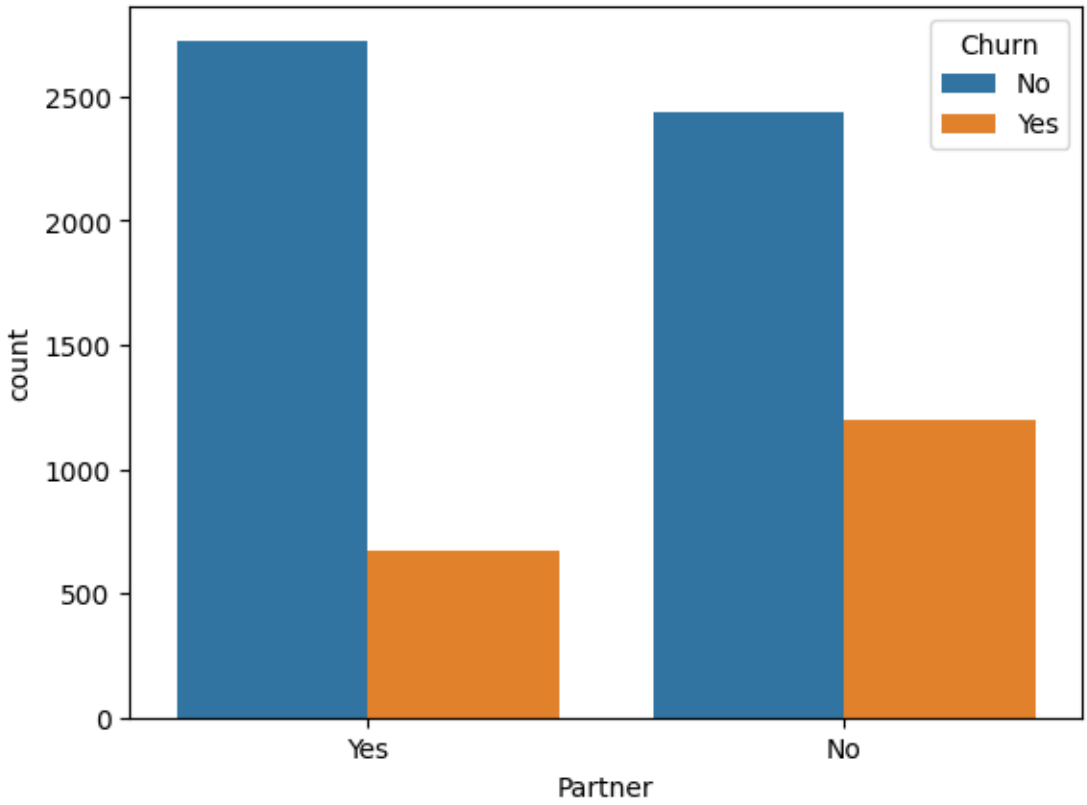

```
Churn
No     5163
Yes    1869
Name: count, dtype: int64
```

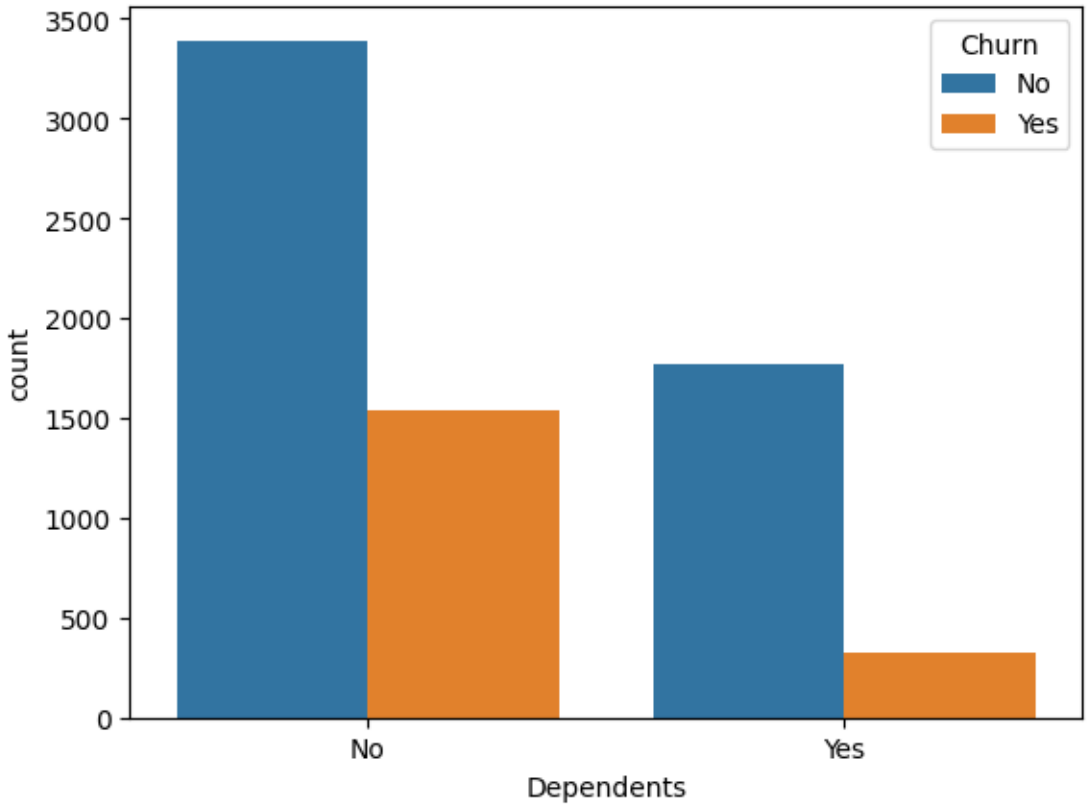


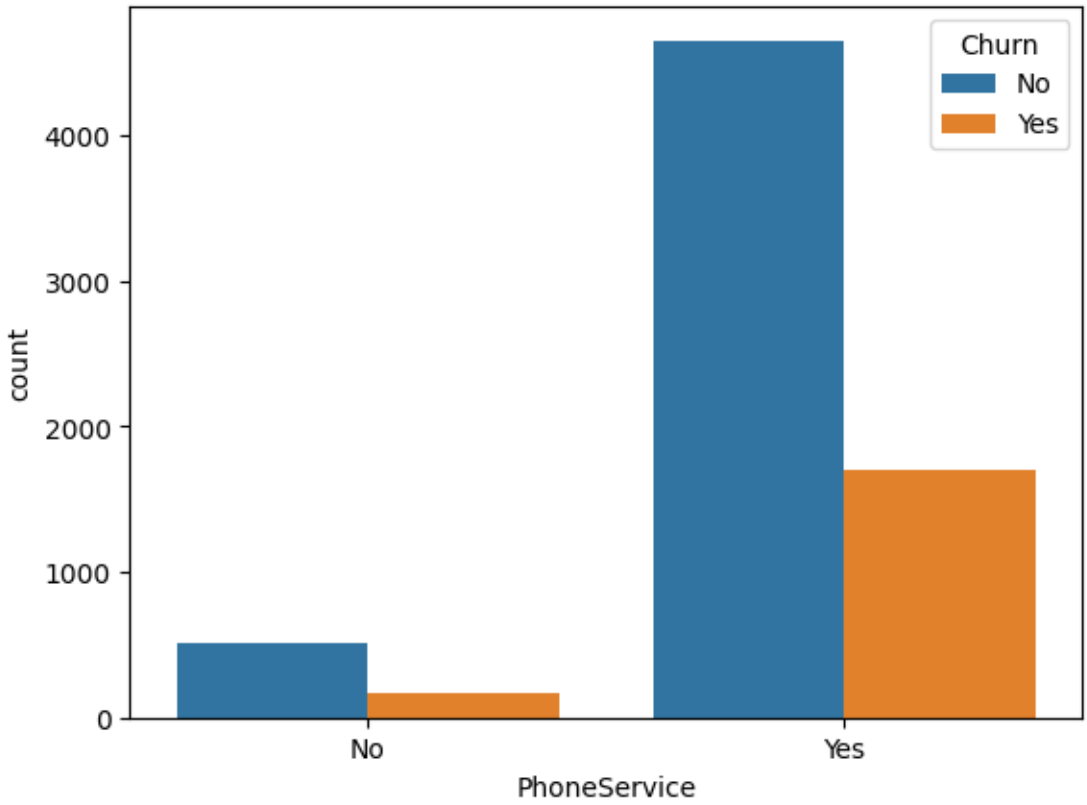
[12]:
```
# Kategorik değişkenlerin hedef değişken ile ilişkisi
for col in kategorik_degiskenler:
    if col != 'Churn':
        sns.countplot(x=col, hue='Churn', data=df)
        plt.show()
```

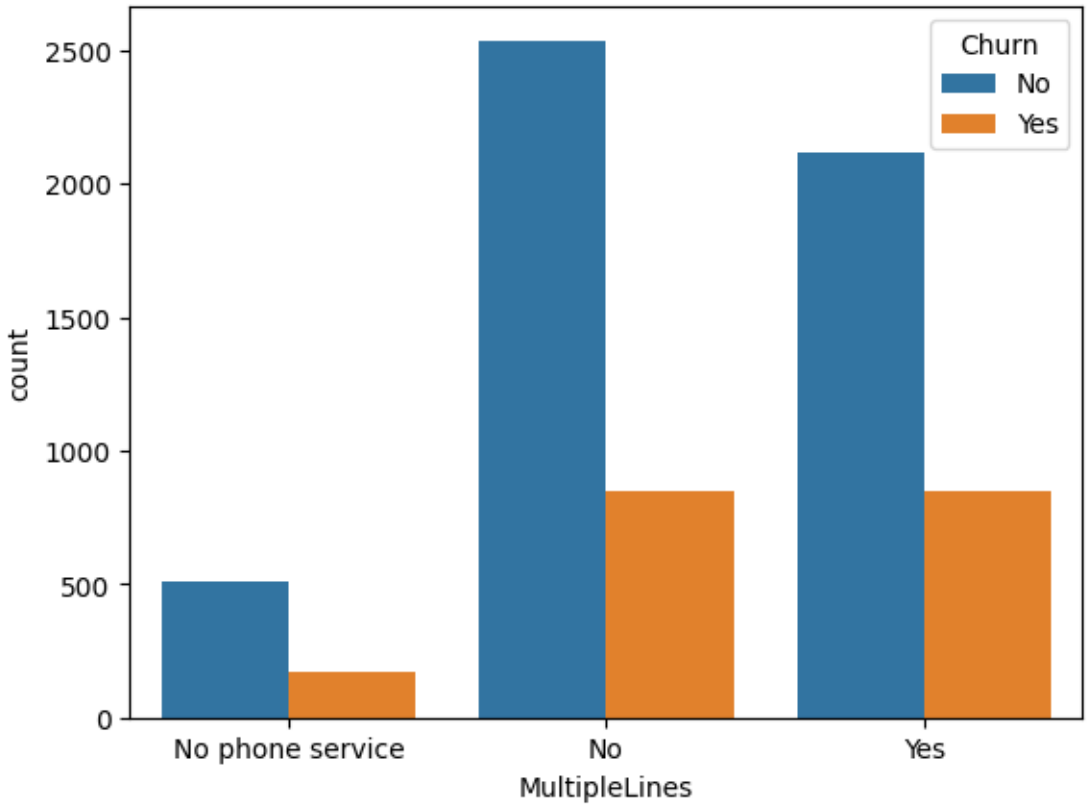

Churn
No
Yes
1.0
0.8
0.6
0.4
0.2
0.0
count
customerID

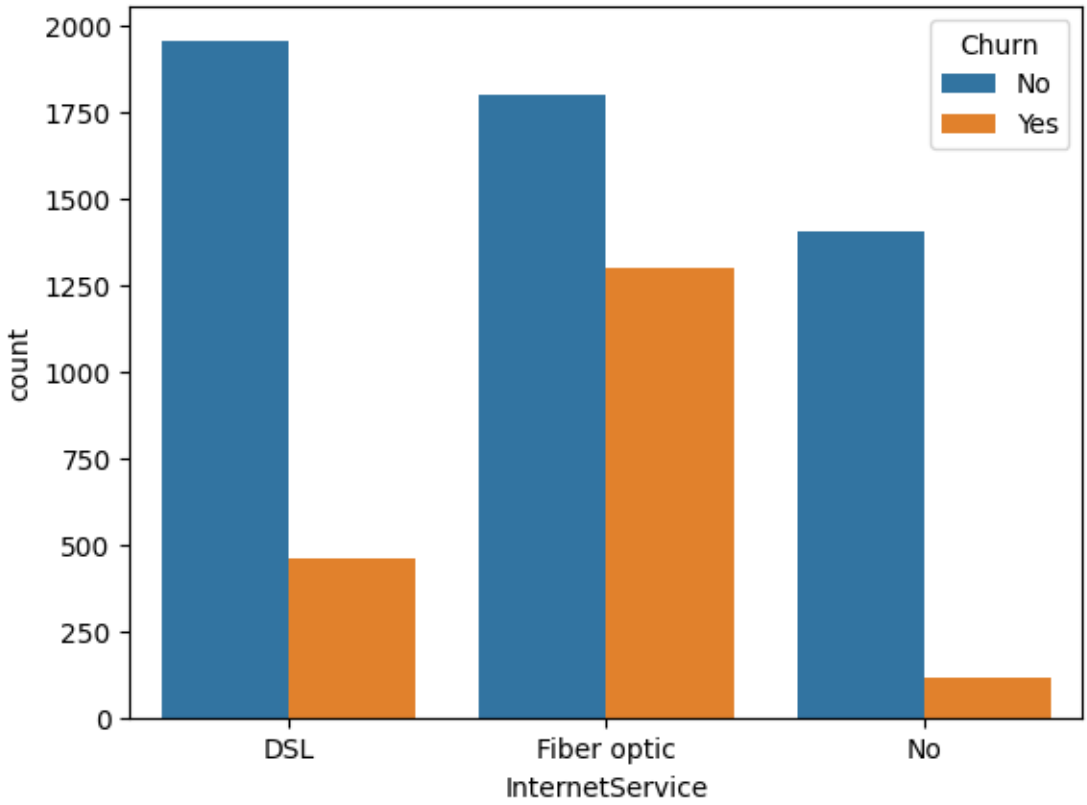

Churn
No
Yes
2500
2000
1500
1000
500
0
count
Female
Male
gender

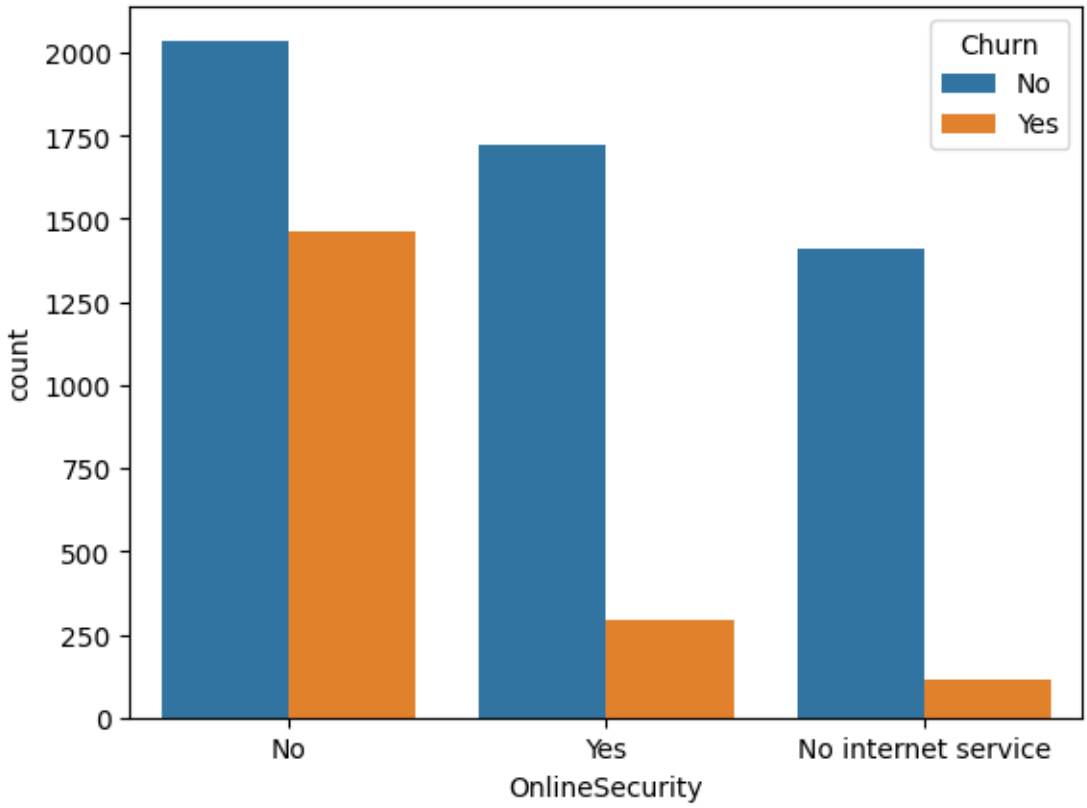

Churn
No
Yes
2500
2000
1500
1000
500
0
count
Yes
No
Partner

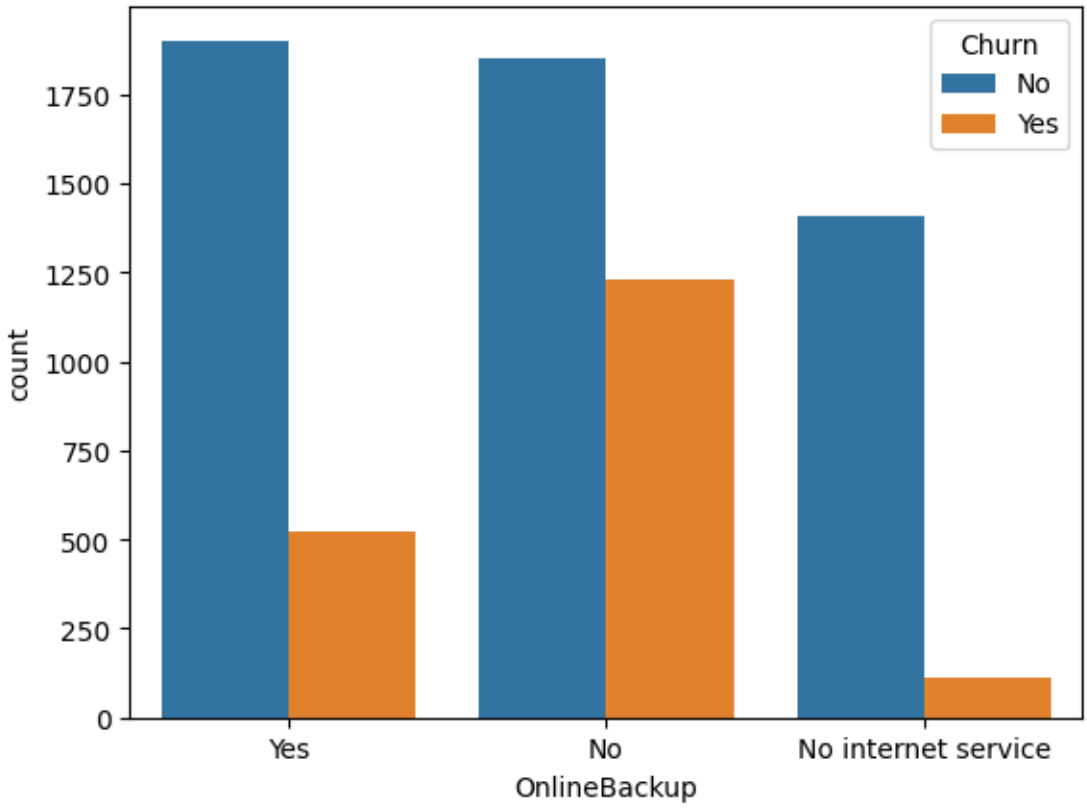

Churn
No
Yes
3500
3000
2500
2000
1500
1000
500
0
count
No
Yes
Dependents

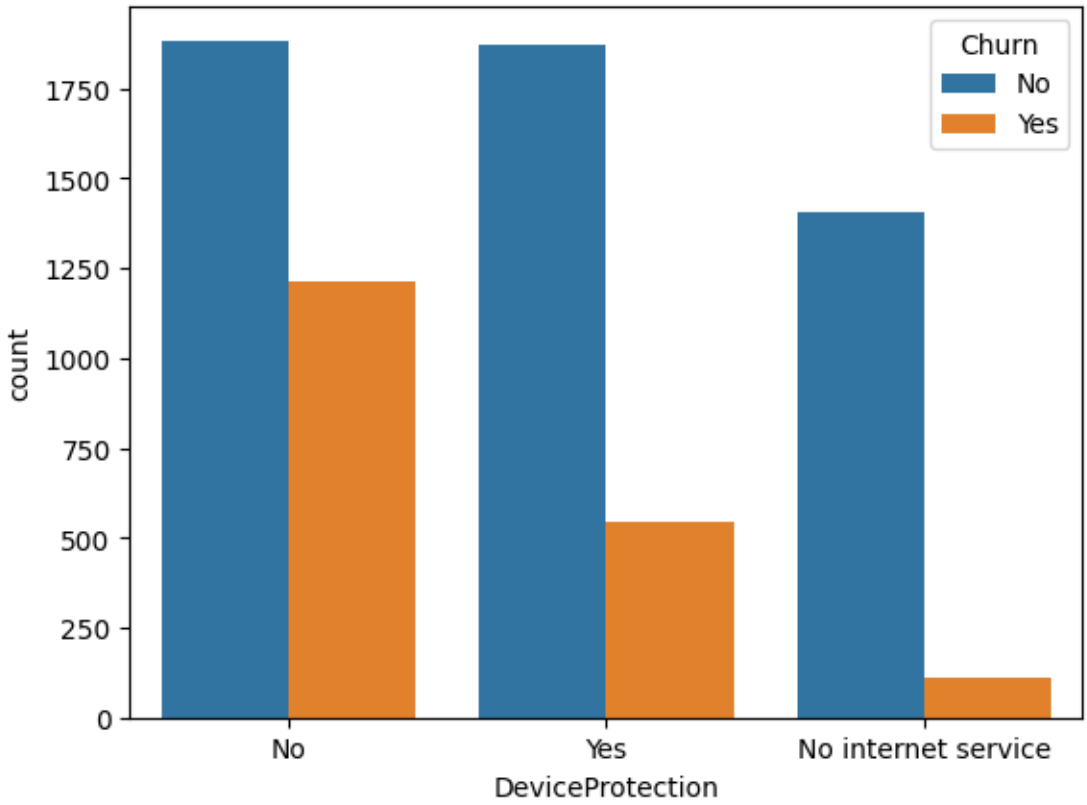

Churn
No
Yes
4000
3000
2000
1000
0
count
No
Yes
PhoneService

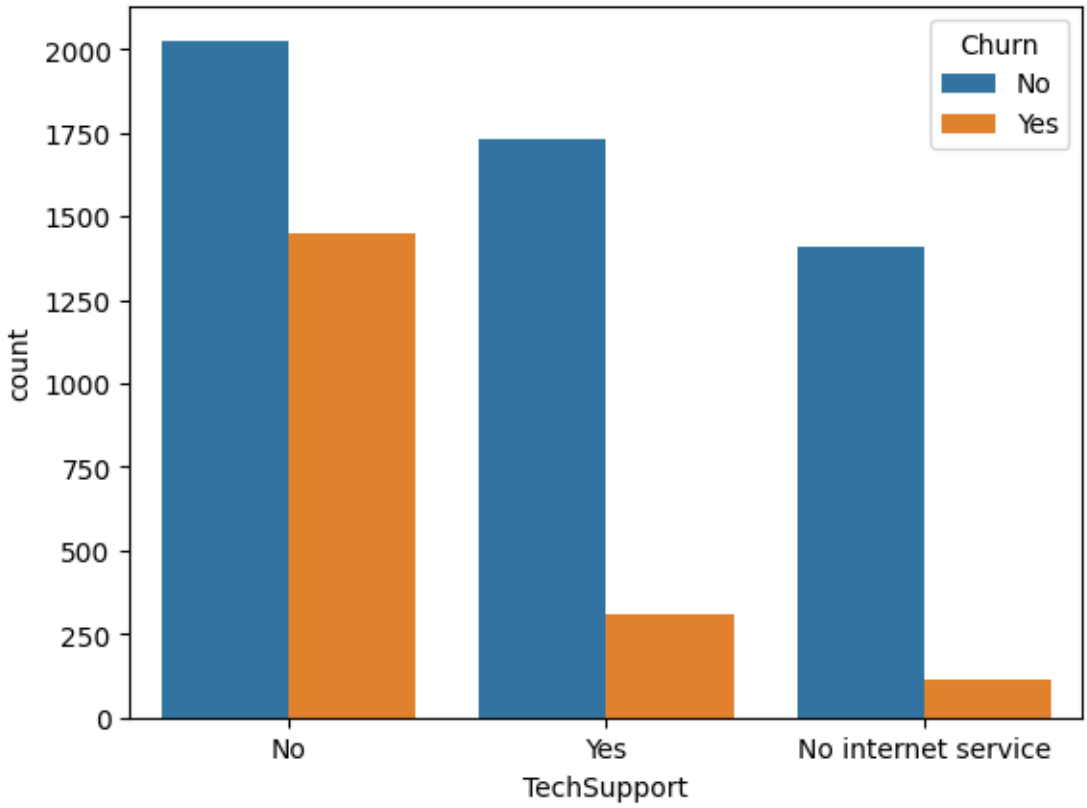

Churn
No
Yes
2500
2000
1500
1000
500
0
count
No phone service
No
Yes
MultipleLines

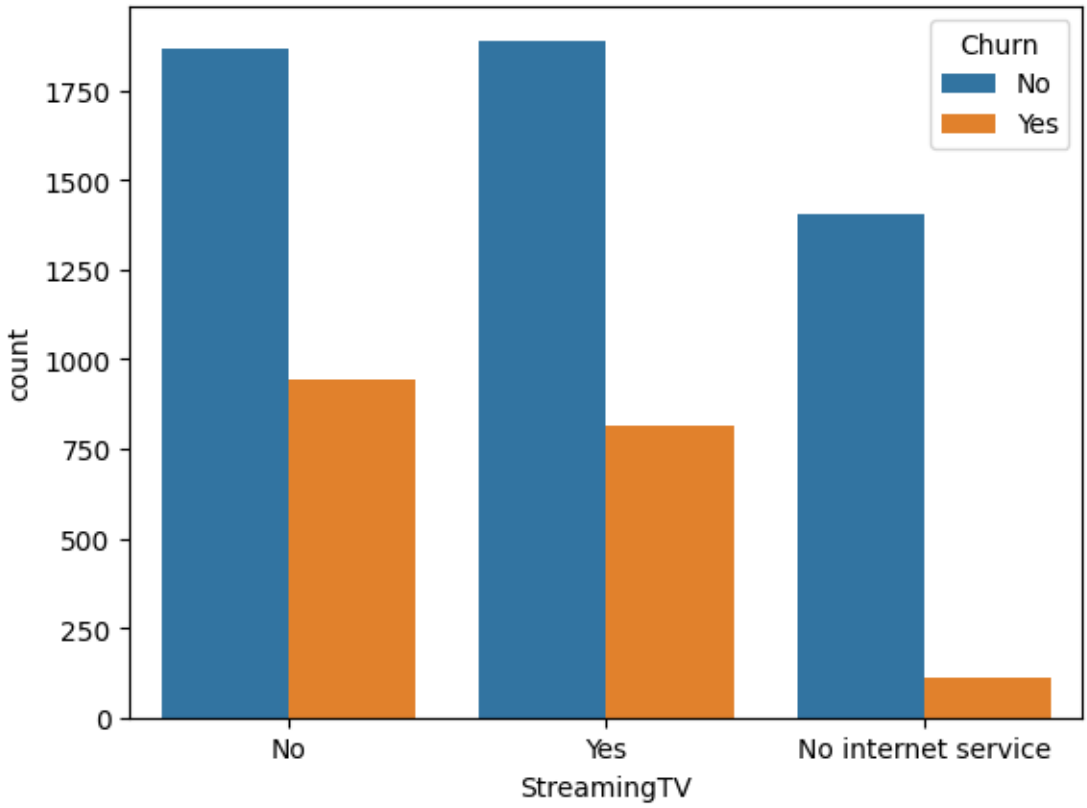

Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
DSL
Fiber optic
No
InternetService

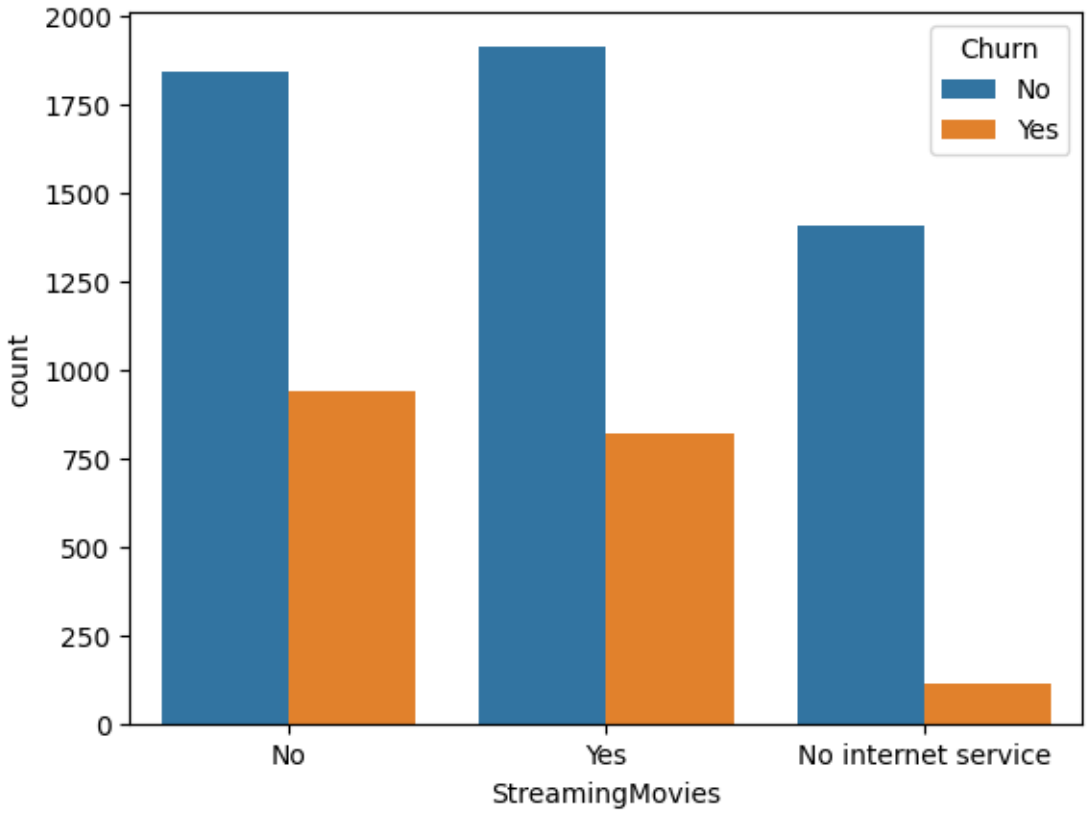

Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
OnlineSecurity

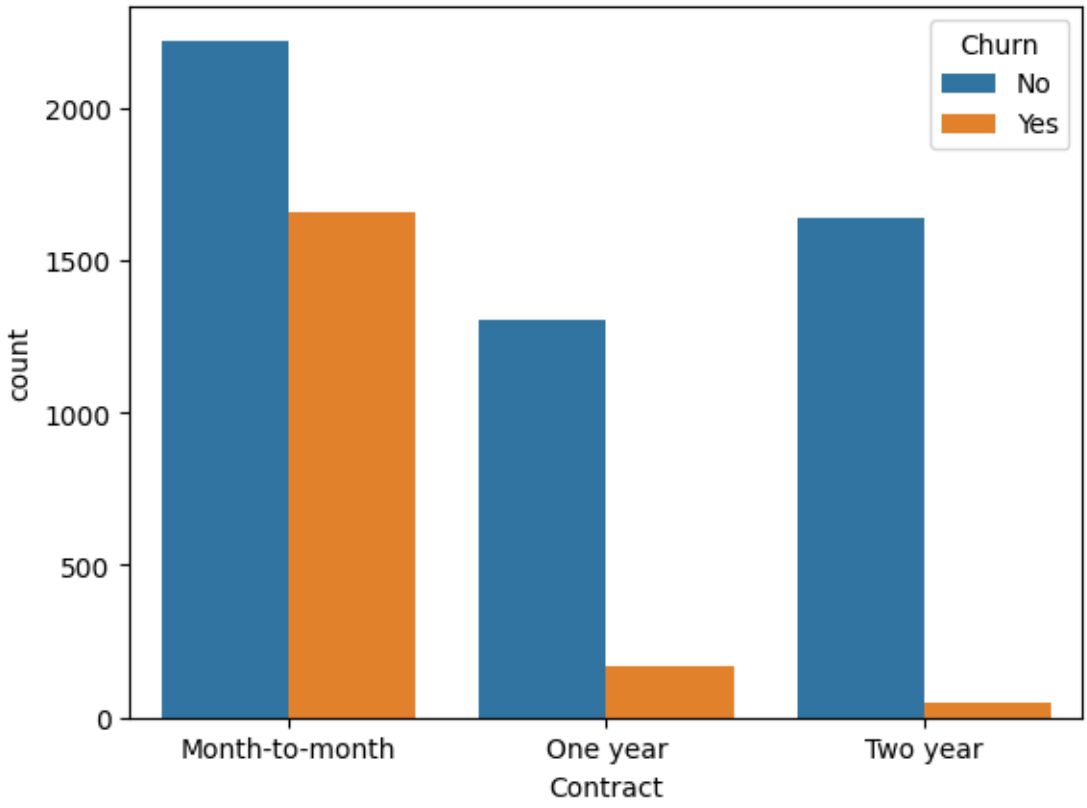

Churn
No
Yes
1750
1500
1250
1000
750
500
250
0
count
Yes
No
No internet service
OnlineBackup

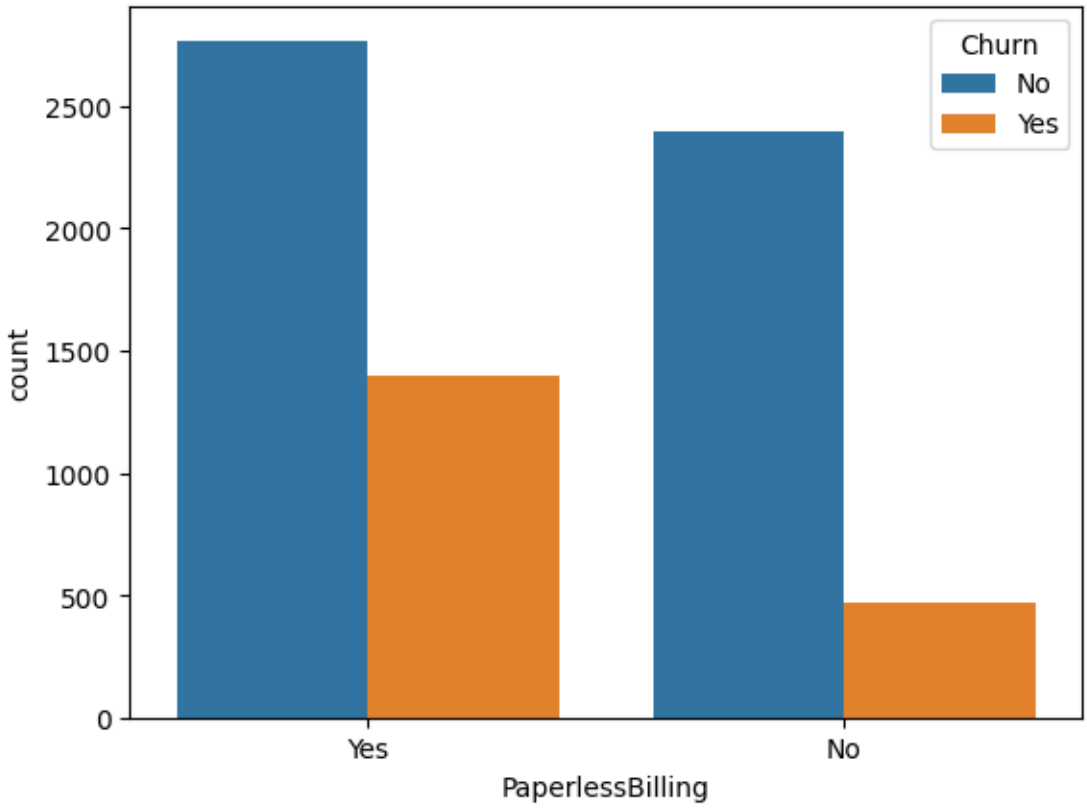

Churn
No
Yes
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
DeviceProtection

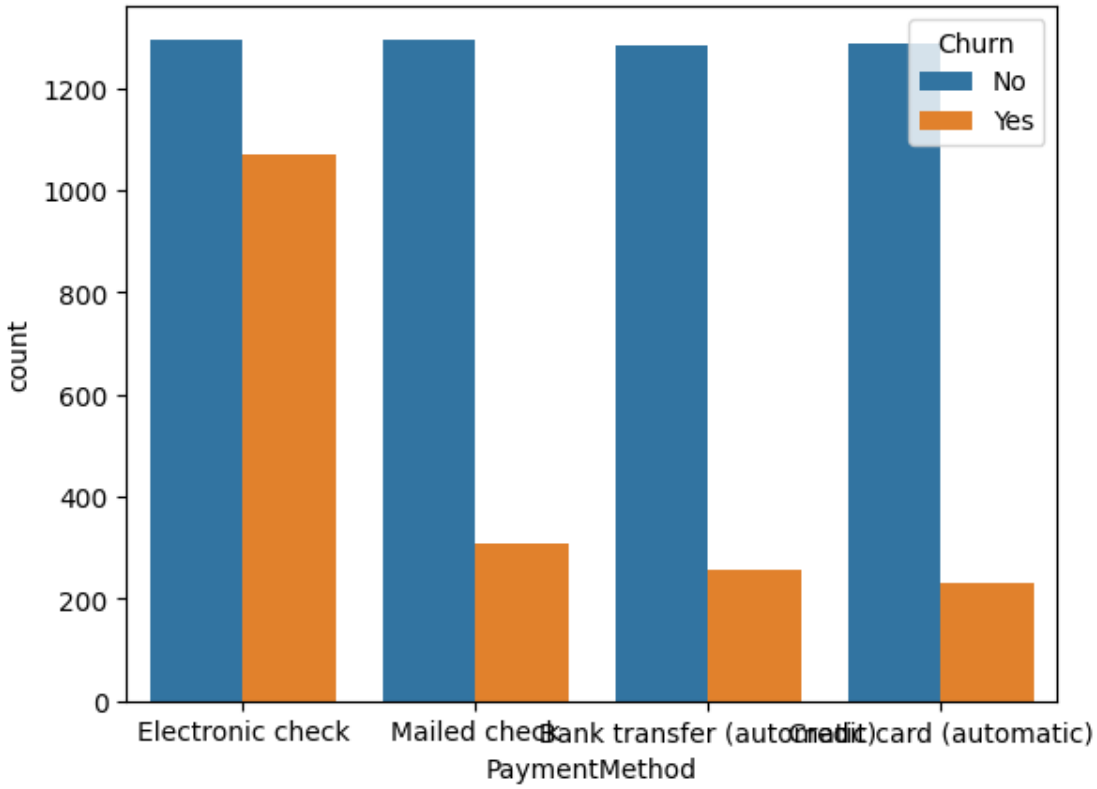

Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
TechSupport

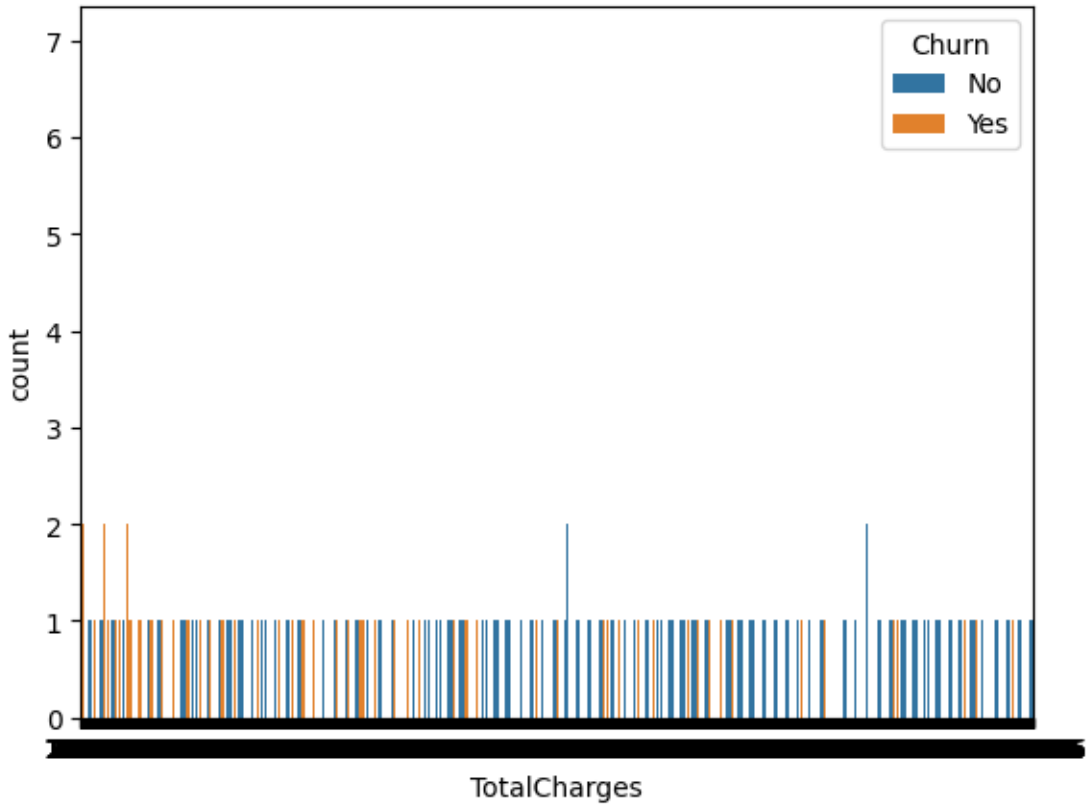

Churn
No
Yes
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
StreamingTV


Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
StreamingMovies


Churn
No
Yes
2000
1500
1000
500
0
count
Month-to-month
One year
Two year
Contract

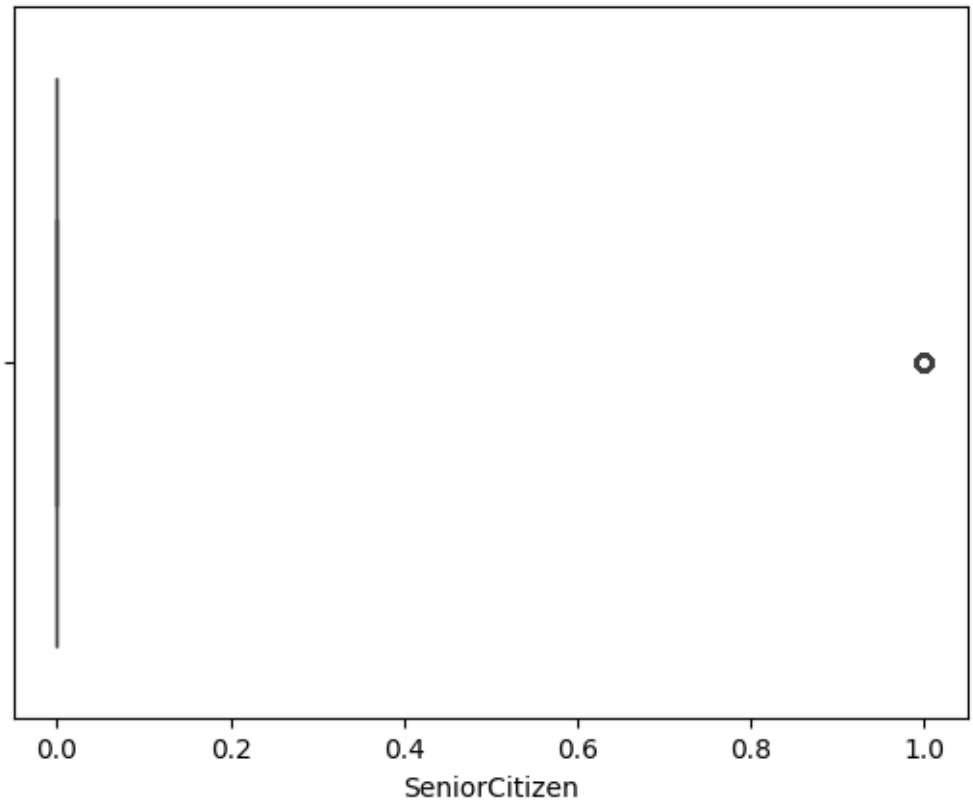

Churn
No
Yes
2500
2000
1500
1000
500
0
count
Yes
No
PaperlessBilling

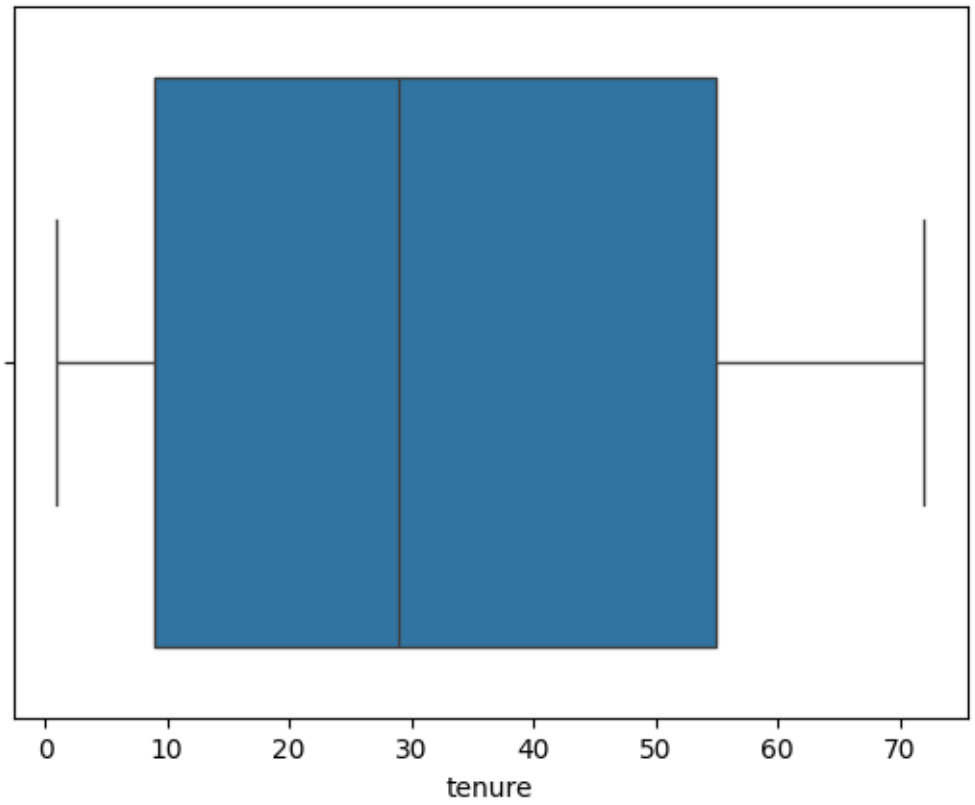

Churn
No
Yes
count
0
200
400
600
800
1000
1200
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod

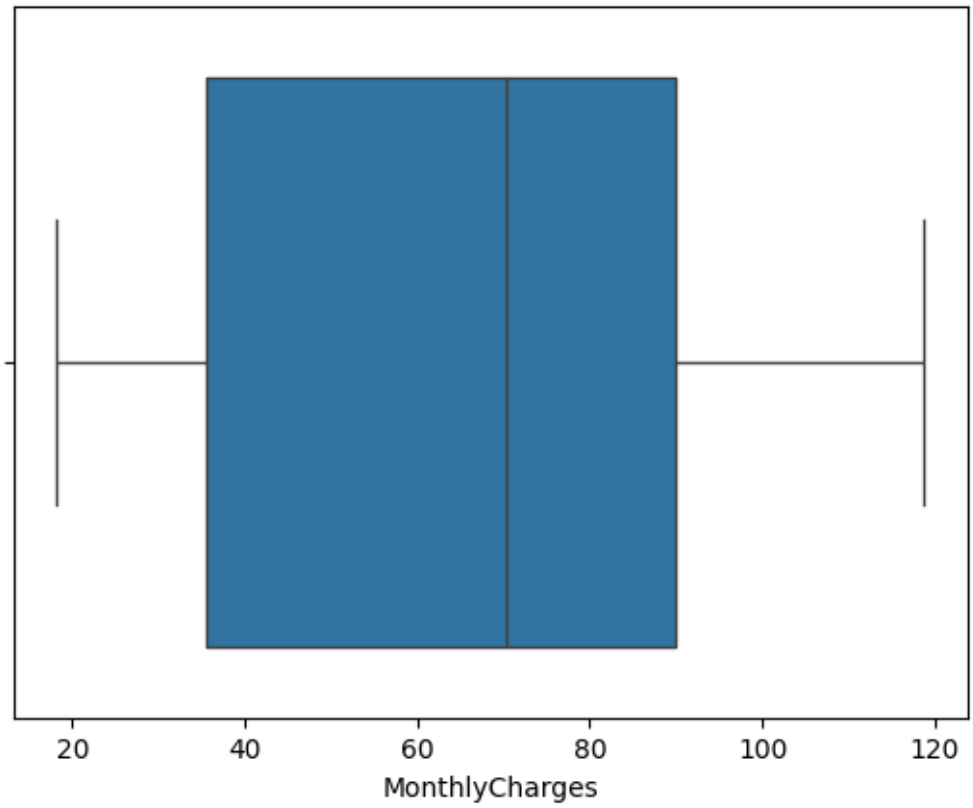


Adım 5

[14]:
```
# Aykırı gözlem kontrolü
for col in numerik_degiskenler:
    sns.boxplot(x=df[col])
    plt.show()
```


0.0
0.2
0.4
0.6
0.8
1.0
SeniorCitizen


0
10
20
30
40
50
60
70
tenure



Adım 6

[13]:
```
df.isnull().sum()
```

[13]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
```

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

1 Task 2

Adım 1

[16]:
```
# Aykırı gözlemleri IQR yöntemiyle bulma ve temizleme
for col in numerik_degiskenler:
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1
    lower_limit = q1 - 1.5 * iqr
    upper_limit = q3 + 1.5 * iqr
    df = df[(df[col] >= lower_limit) & (df[col] <= upper_limit)];
```

[17]:
```
df.head()
```

[17]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	Female	0	Yes	No	1	No	…	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	5575-GNVDE	Male	0	No	No	34	Yes	…	Yes	No	No	No	One year	No	Mailed check	56.95	1889.50	No
2	3668-QPYBK	Male	0	No	No	2	Yes	…	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	7795-CFOCW	Male	0	No	No	45	No	…	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	9237-HQITU	Female	0	No	No	2	Yes	…	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No

```
4           Electronic check          70.70        151.65    Yes

[5 rows x 21 columns]
```

Adım 2

[18]:
```
# tenure kategorik değişkene dönüştürme
df['tenure_group'] = pd.cut(df['tenure'], bins=[0, 12, 24, 36, 48, 60, 72],␣
 ↪labels=['0-12', '13-24', '25-36', '37-48', '49-60', '61-72'])
```

[19]:
```
# TotalCharges ve MonthlyCharges kategorik değişkene dönüştürme
df['TotalCharges_group'] = pd.cut(df['TotalCharges'], bins=4, labels=['Low',␣
 ↪'Medium', 'High', 'Very High'])
df['MonthlyCharges_group'] = pd.cut(df['MonthlyCharges'], bins=4,␣
 ↪labels=['Low', 'Medium', 'High', 'Very High'])
```

[20]:
```
df.head()
```

[20]:

	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	…	StreamingMovies	\
0	No phone service	DSL	No	…	No	
1	No	DSL	Yes	…	No	
2	No	DSL	Yes	…	No	
3	No phone service	DSL	Yes	…	No	
4	No	Fiber optic	No	…	No	

	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	\
0	Month-to-month	Yes	Electronic check	29.85	
1	One year	No	Mailed check	56.95	
2	Month-to-month	Yes	Mailed check	53.85	
3	One year	No	Bank transfer (automatic)	42.30	
4	Month-to-month	Yes	Electronic check	70.70	

	TotalCharges	Churn	tenure_group	TotalCharges_group	MonthlyCharges_group
0	29.85	No	0-12	Low	Low
1	1889.50	No	25-36	Low	Medium
2	108.15	Yes	0-12	Low	Medium
3	1840.75	No	37-48	Low	Low
4	151.65	Yes	0-12	Low	High

[5 rows x 24 columns]

Adım 3

[22]:
```
# Label encoding for binary categorical variables
binary_columns = ['gender', 'Partner', 'Dependents', 'PhoneService',␣
 ↪'PaperlessBilling', 'Churn']
le = LabelEncoder()
for col in binary_columns:
    df[col] = le.fit_transform(df[col])
```

[23]:
```
# One-hot encoding for multi-category categorical variables
multi_category_columns = ['MultipleLines', 'InternetService', 'OnlineSecurity',␣
 ↪'OnlineBackup', 'DeviceProtection',
                          'TechSupport', 'StreamingTV', 'StreamingMovies',␣
 ↪'Contract', 'PaymentMethod',
                          'tenure_group', 'TotalCharges_group',␣
 ↪'MonthlyCharges_group']
```

[24]:
```
df = pd.get_dummies(df, columns=multi_category_columns)
```

[25]:
```
df.head()
```

[25]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	\
0	7590-VHVEG	0	0	1	0	1	
1	5575-GNVDE	1	0	0	0	34	
2	3668-QPYBK	1	0	0	0	2	
3	7795-CFOCW	1	0	0	0	45	
4	9237-HQITU	0	0	0	0	2	

	PhoneService	PaperlessBilling	MonthlyCharges	TotalCharges	…	\
0	0	1	29.85	29.85	…	
1	1	0	56.95	1889.50	…	
2	1	1	53.85	108.15	…	
3	0	0	42.30	1840.75	…	
4	1	1	70.70	151.65	…	

	tenure_group_49-60	tenure_group_61-72	TotalCharges_group_Low	\
0	False	False	True	
1	False	False	True	
2	False	False	True	
3	False	False	True	
4	False	False	True	

	TotalCharges_group_Medium	TotalCharges_group_High	\
0	False	False	
1	False	False	
2	False	False	
3	False	False	

```
4                      False                      False

   TotalCharges_group_Very High  MonthlyCharges_group_Low  \
0                         False                      True
1                         False                     False
2                         False                     False
3                         False                      True
4                         False                     False

   MonthlyCharges_group_Medium  MonthlyCharges_group_High  \
0                        False                     False
1                         True                     False
2                         True                     False
3                        False                     False
4                        False                      True

   MonthlyCharges_group_Very High
0                           False
1                           False
2                           False
3                           False
4                           False

[5 rows x 56 columns]
```

Adım 4

[27]:
```
# Standartlaştırma işlemi
scaler = StandardScaler()
numerik_degiskenler = ['tenure', 'MonthlyCharges', 'TotalCharges',␣
 ↪'SeniorCitizen']
df[numerik_degiskenler] = scaler.fit_transform(df[numerik_degiskenler])
```

[28]:
```
df.head()
```

[28]:
```
    customerID  gender  SeniorCitizen  Partner  Dependents    tenure  \
0  7590-VHVEG       0            0.0        1           0 -1.269893
1  5575-GNVDE       1            0.0        0           0  0.071016
2  3668-QPYBK       1            0.0        0           0 -1.229260
3  7795-CFOCW       1            0.0        0           0  0.517985
4  9237-HQITU       0            0.0        0           0 -1.229260

   PhoneService  PaperlessBilling  MonthlyCharges  TotalCharges  …  \
0             0                 1       -1.056827     -0.963373  …
1             1                 0       -0.162821     -0.130580  …
2             1                 1       -0.265087     -0.928309  …
3             0                 0       -0.646112     -0.152412  …
```

```
4             1                 1        0.290780     -0.908828  …

   tenure_group_49-60  tenure_group_61-72  TotalCharges_group_Low  \
0               False               False                    True
1               False               False                    True
2               False               False                    True
3               False               False                    True
4               False               False                    True

   TotalCharges_group_Medium  TotalCharges_group_High  \
0                      False                    False
1                      False                    False
2                      False                    False
3                      False                    False
4                      False                    False

   TotalCharges_group_Very High  MonthlyCharges_group_Low  \
0                         False                      True
1                         False                     False
2                         False                     False
3                         False                      True
4                         False                     False

   MonthlyCharges_group_Medium  MonthlyCharges_group_High  \
0                        False                     False
1                         True                     False
2                         True                     False
3                        False                     False
4                        False                      True

   MonthlyCharges_group_Very High
0                           False
1                           False
2                           False
3                           False
4                           False

[5 rows x 56 columns]
```

2 Task 3

Adım 1

[30]:
```
# Veri ve hedef değişkenleri ayıralım
X = df.drop(columns=['customerID', 'Churn'])
y = df['Churn']
```

[31]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[42]:
```
models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Support Vector Machine": SVC(),
    "Naive Bayes": GaussianNB(),
    "Gradient Boosting": GradientBoostingClassifier(),
    "AdaBoost": AdaBoostClassifier(),
    "XGBoost": xgb.XGBClassifier()
 }
```

[43]:
```
accuracy_scores = {}
```

[44]:
```
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy_scores[model_name] = accuracy_score(y_test, y_pred)
```

```
c:\Users\abdur\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[45]:
```
# Accuracy skorlarını inceleyelim
for model_name, score in accuracy_scores.items():
    print(f"{model_name}: {score:.4f}")
```

```
Logistic Regression: 0.8098
Decision Tree: 0.7462
Random Forest: 0.8048
Support Vector Machine: 0.8098
Naive Bayes: 0.7173
Gradient Boosting: 0.8107
AdaBoost: 0.8081
XGBoost: 0.7946
```

[46]:
```
# En iyi 4 modeli seçelim
top_4_models = sorted(accuracy_scores, key=accuracy_scores.get, reverse=True)[:
 ↪4]
print("En iyi 4 model:", top_4_models)
```

```
En iyi 4 model: ['Gradient Boosting', 'Logistic Regression', 'Support Vector
Machine', 'AdaBoost']
```

Adım 2

[49]:
```
# Hiperparametre aralıklarını belirleyelim
param_grid = {
     "Gradient Boosting": {
        'n_estimators': [100, 200],
        'learning_rate': [0.01, 0.1, 0.2],
        'max_depth': [3, 4, 5]
    },
    "Logistic Regression": {
        'C': [0.1, 1, 10, 100],
        'solver': ['liblinear']
    },
    "Support Vector Machine": {
        'C': [0.1, 1, 10, 100],
        'gamma': [1, 0.1, 0.01, 0.001]
    },
         "AdaBoost": {
        'n_estimators': [50, 100, 200],
        'learning_rate': [0.01, 0.1, 0.5]
    },
}
```

[50]:
```
best_params = {}
best_scores = {}
```

[51]:
```
# Hiperparametre optimizasyonunu gerçekleştirelim
for model_name in top_4_models:
    grid_search = GridSearchCV(models[model_name], param_grid[model_name],␣
 ↪cv=5, scoring='accuracy', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    best_params[model_name] = grid_search.best_params_
    best_scores[model_name] = grid_search.best_score_
```

[52]:
```
# En iyi hiperparametreler ve doğruluk skorları
for model_name in best_params:
    print(f"{model_name} - Best Params: {best_params[model_name]} - Best Score:␣
 ↪{best_scores[model_name]:.4f}")
```

```
Gradient Boosting - Best Params: {'learning_rate': 0.1, 'max_depth': 3,
```

```
'n_estimators': 100} - Best Score: 0.8158
Logistic Regression - Best Params: {'C': 100, 'solver': 'liblinear'} - Best
Score: 0.8213
Support Vector Machine - Best Params: {'C': 100, 'gamma': 0.01} - Best Score:
0.8166
AdaBoost - Best Params: {'learning_rate': 0.5, 'n_estimators': 100} - Best
Score: 0.8203
```

[53]:
```python
# Seçilen modelleri en iyi hiperparametreler ile tekrar eğitelim
final_models = {}
for model_name in best_params:
    model = models[model_name].set_params(**best_params[model_name])
    model.fit(X_train, y_train)
    final_models[model_name] = model
```

[54]:
```python
# Final modellerin doğruluk skorlarını test seti üzerinde inceleyelim
final_accuracy_scores = {}
for model_name, model in final_models.items():
    y_pred = model.predict(X_test)
    final_accuracy_scores[model_name] = accuracy_score(y_test, y_pred)
```

[55]:
```python
# Final doğruluk skorları
for model_name, score in final_accuracy_scores.items():
    print(f"{model_name} Final Accuracy: {score:.4f}")
```

```
Gradient Boosting Final Accuracy: 0.8098
Logistic Regression Final Accuracy: 0.8098
Support Vector Machine Final Accuracy: 0.8124
AdaBoost Final Accuracy: 0.8107
```