```
# NAME :- Akula Hema Venkata Sriram
# ROLL NO. :- 04
# REGISTRATION NO. :- 12210461
# SECTION :- K22BW
# COURSE CODE :- INT-254

#-----------------------------------------------
# PROJECT TITLE :- CUSTOMER CHURN PREDICTION
#-----------------------------------------------

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

telco_data = pd.read_csv("telco_customer_churn.csv")

# Display basic information about the dataset
print("Shape of the dataset:", telco_data.shape)
print("\nColumns in the dataset:")
print(telco_data.columns)
print("\nSample data:")
print(telco_data.head())

# Summary statistics
print("\nSummary statistics:")
print(telco_data.describe())

# Check for missing values
print("\nMissing values:")
print(telco_data.isnull().sum())

print("\nColumn names : ")
print(telco_data.columns.values)

print("\nColumns Data Types : ")
print(telco_data.dtypes)
# Check for duplicate rows
print("\nDuplicate rows:", telco_data.duplicated().sum())

# Visualize the distribution of the target variable 'Churn'
plt.figure(figsize=(8, 6))
sns.countplot(x='Churn', data=telco_data)
plt.title('Distribution of Churn')
plt.show()

# Visualize the distribution of numerical features
numerical_features =
telco_data.select_dtypes(include=[np.number]).columns.tolist()
telco_data[numerical_features].hist(figsize=(12, 10))
```

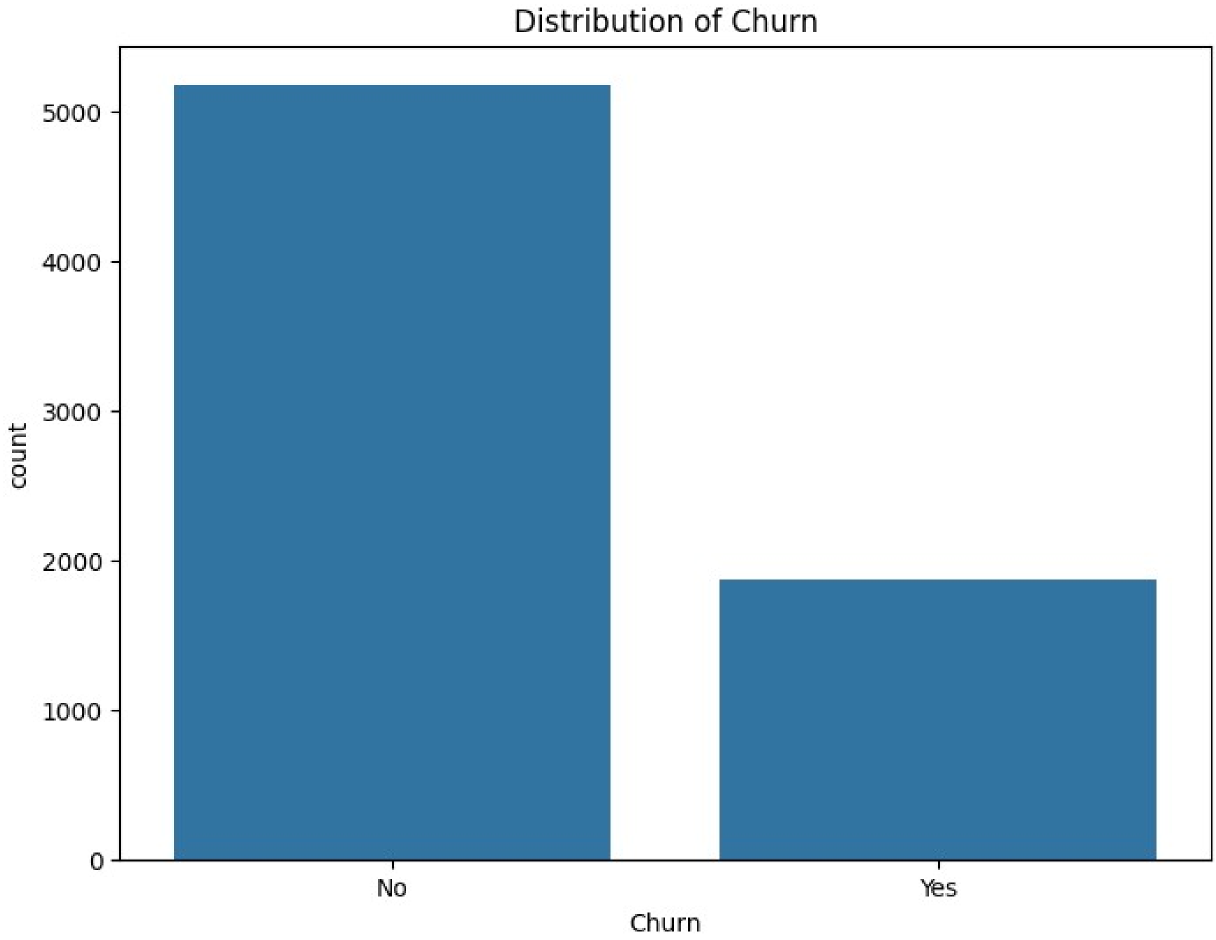
```
plt.suptitle('Distribution of Numerical Features')
plt.show()

# Visualize the correlation matrix
plt.figure(figsize=(10, 8))
correlation_matrix = telco_data[numerical_features].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

```
C:\Users\Hitech\AppData\Local\Temp\ipykernel_4944\4206063596.py:2:
DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major
release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type,
and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at
https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd

Shape of the dataset: (7043, 21)

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

Sample data:
   customerID  gender  SeniorCitizen Partner Dependents  tenure
PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1
No
1  5575-GNVDE    Male              0      No         No      34
Yes
2  3668-QPYBK    Male              0      No         No       2
Yes
```

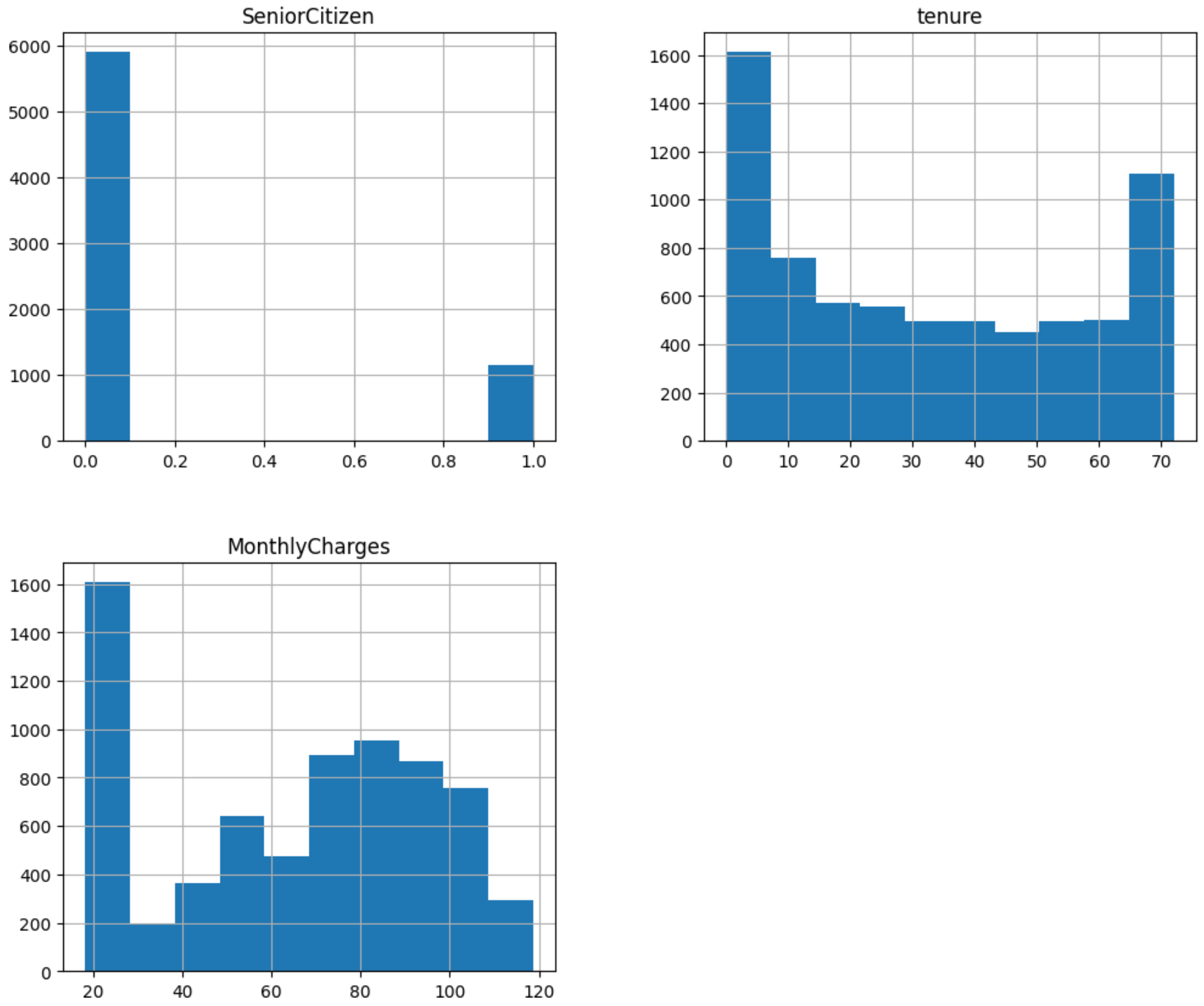
```
3  7795-CFOCW    Male                 0      No          No       45
No
4  9237-HQITU  Female                 0      No          No        2
Yes

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

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

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

[5 rows x 21 columns]

Summary statistics:
       SeniorCitizen       tenure  MonthlyCharges
count    7043.000000  7043.000000     7043.000000
mean        0.162147    32.371149       64.761692
std         0.368612    24.559481       30.090047
min         0.000000     0.000000       18.250000
25%         0.000000     9.000000       35.500000
50%         0.000000    29.000000       70.350000
75%         0.000000    55.000000       89.850000
max         1.000000    72.000000      118.750000
```

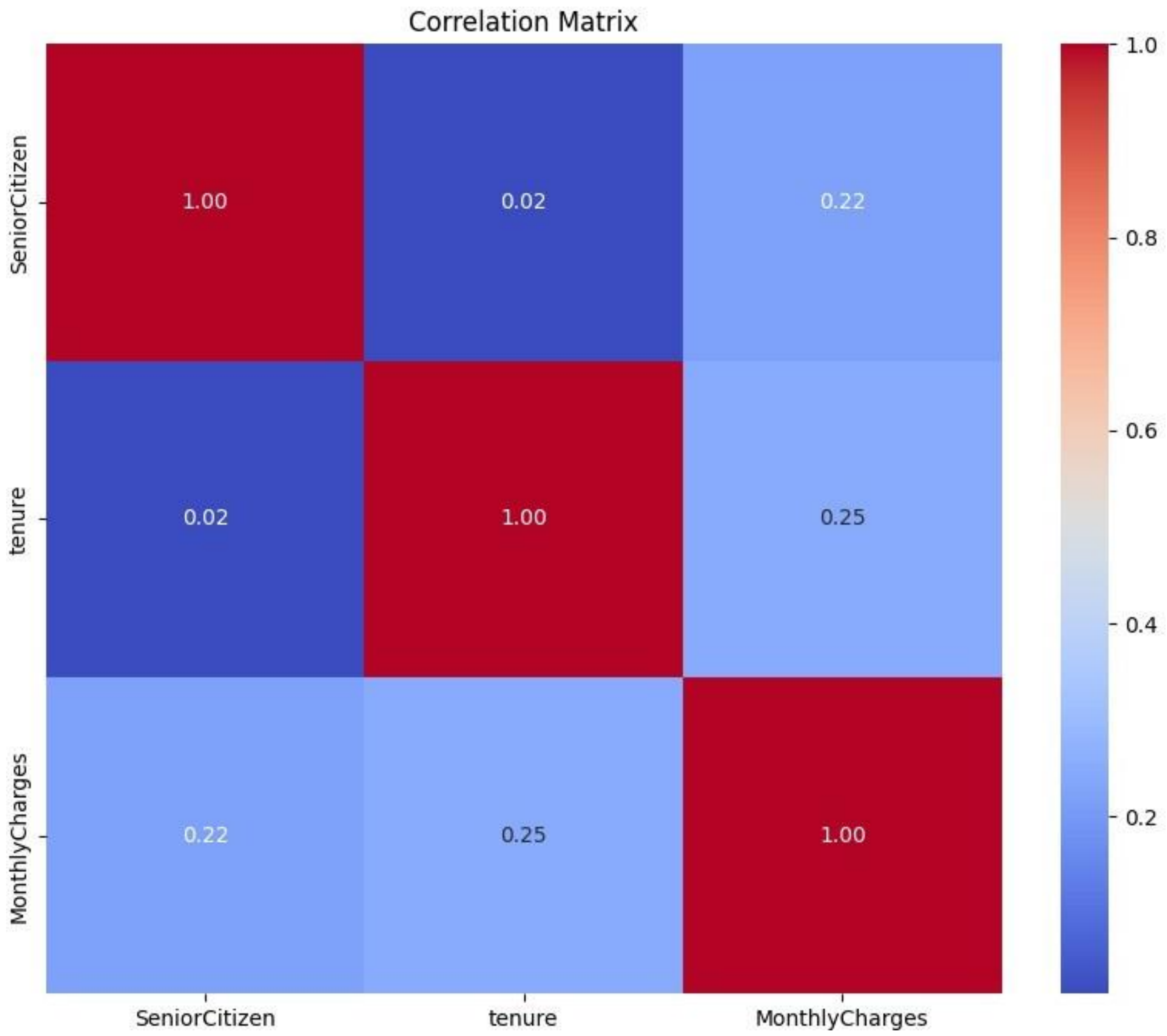
```
Missing values:
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64

Column names :
['customerID' 'gender' 'SeniorCitizen' 'Partner' 'Dependents' 'tenure'
 'PhoneService' 'MultipleLines' 'InternetService' 'OnlineSecurity'
 'OnlineBackup' 'DeviceProtection' 'TechSupport' 'StreamingTV'
 'StreamingMovies' 'Contract' 'PaperlessBilling' 'PaymentMethod'
 'MonthlyCharges' 'TotalCharges' 'Churn']

Columns Data Types :
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
```

```
PaymentMethod          object
MonthlyCharges        float64
TotalCharges           object
Churn                  object
dtype: object

Duplicate rows: 0
```



Distribution of Numerical Features





```
print(100*telco_data['Churn'].value_counts()/len(telco_data['Churn']))
print(telco_data['Churn'].value_counts())
telco_base_data=telco_data.copy()

Churn
No     73.463013
Yes    26.536987
Name: count, dtype: float64
Churn
No     5174
Yes    1869
Name: count, dtype: int64

telco_data.TotalCharges=pd.to_numeric(telco_data.TotalCharges,
errors='coerce')
telco_data.isnull().sum()
```

```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

```
telco_data.loc[telco_data['TotalCharges'].isnull()==True]
telco_data.dropna(how='any',inplace=True)
telco_data.shape
```

```
(7032, 21)
```

```
print(telco_data['tenure'].max())
```

```
72
```

```
labels=["{0} - {1}".format(i,i+11)for i in range(1,72,12)]
telco_data['tenure_group']=pd.cut(telco_data.tenure,range(1,80,12),
right=False, labels=labels)
```

```
telco_data['tenure_group'].value_counts()
```

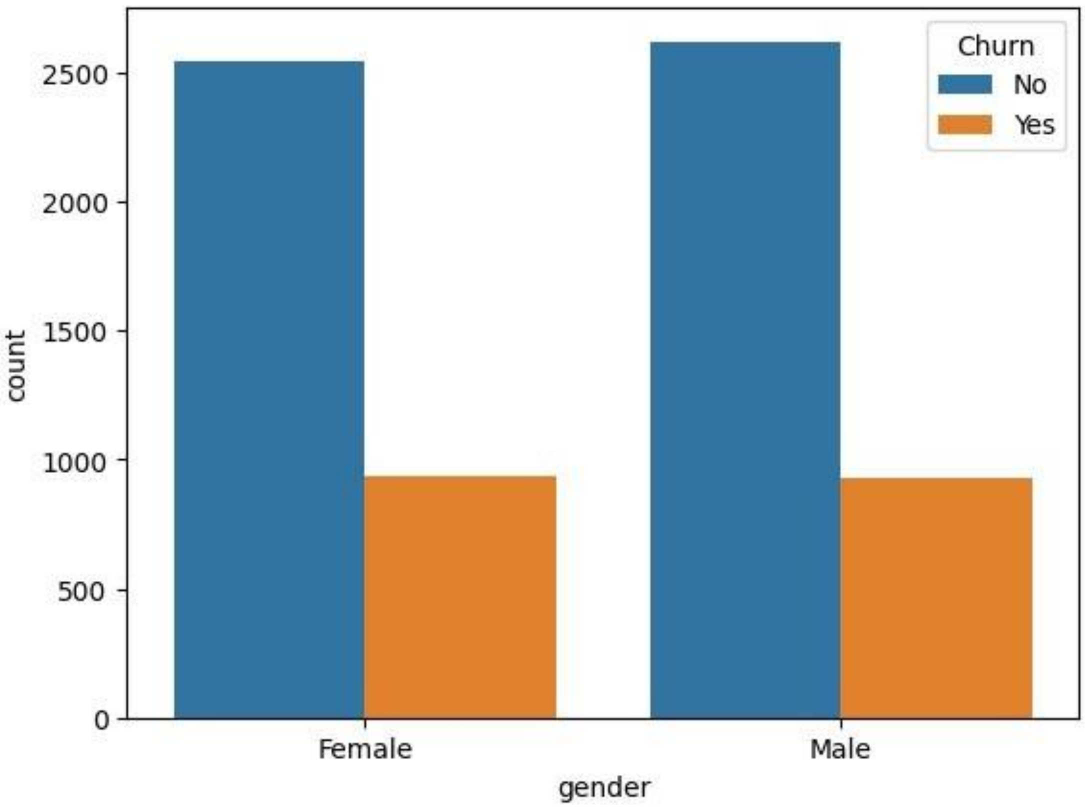
```
tenure_group
1 - 12     2175
61 - 72    1407
13 - 24    1024
25 - 36     832
49 - 60     832
37 - 48     762
Name: count, dtype: int64
```

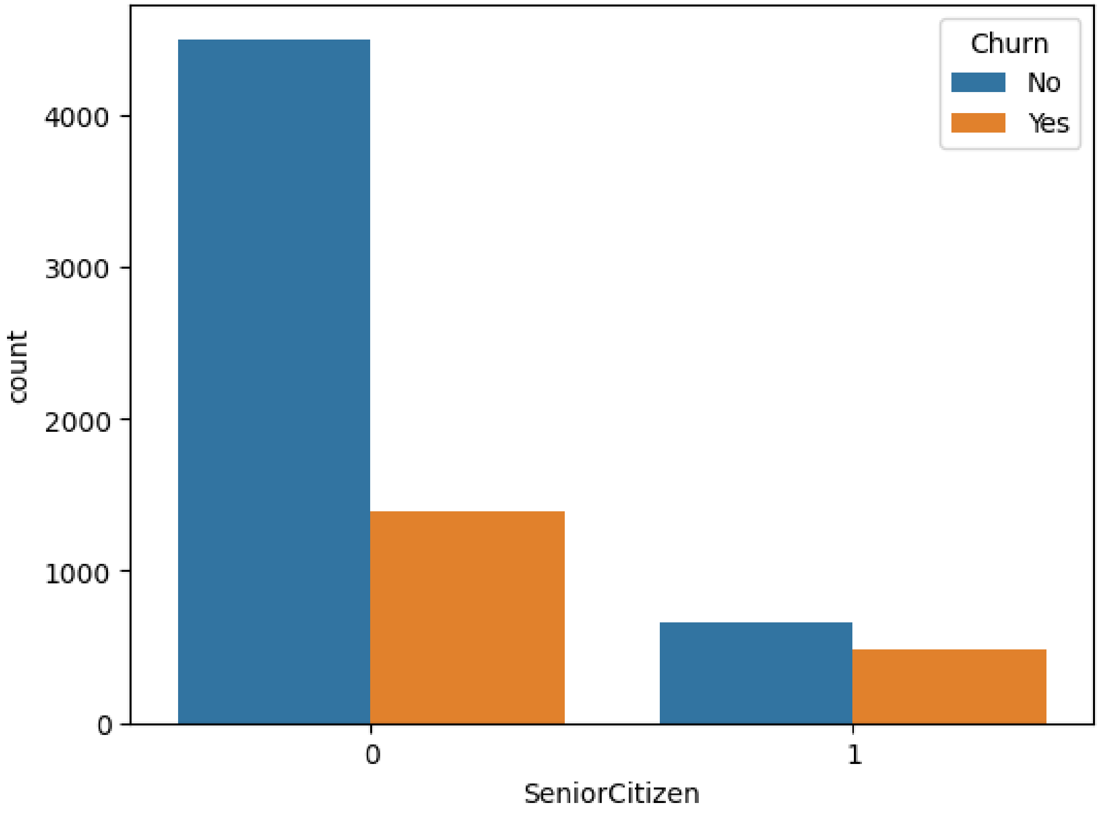
```
telco_data.drop(columns=['customerID'],axis=1,inplace=True)
telco_data.head()
```

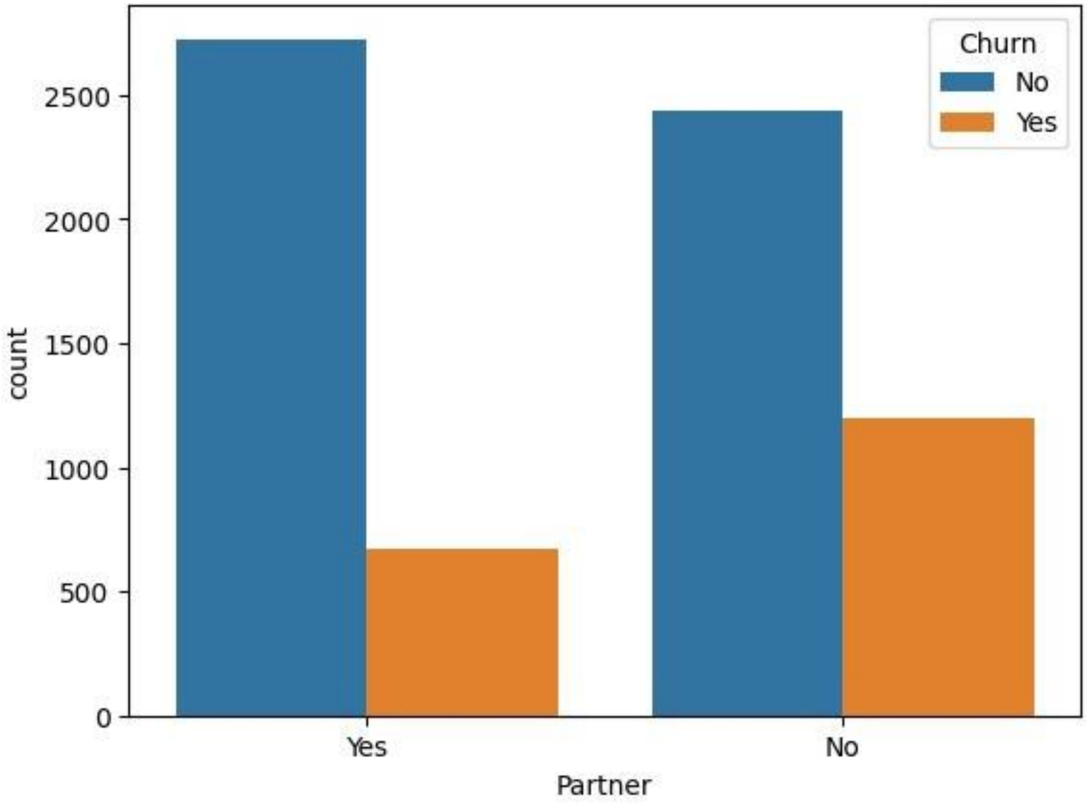
```
   gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  Female              0     Yes         No       1           No
1    Male              0      No         No      34          Yes
2    Male              0      No         No       2          Yes
3    Male              0      No 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


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

              PaymentMethod MonthlyCharges  TotalCharges  Churn
tenure_group
0          Electronic check          29.85         29.85     No
1 - 12
1              Mailed check          56.95       1889.50     No
25 - 36
2              Mailed check          53.85        108.15    Yes
1 - 12
3  Bank transfer (automatic)          42.30       1840.75     No
37 - 48
4          Electronic check          70.70        151.65    Yes
1 - 12

[5 rows x 21 columns]

telco_data.head()
```

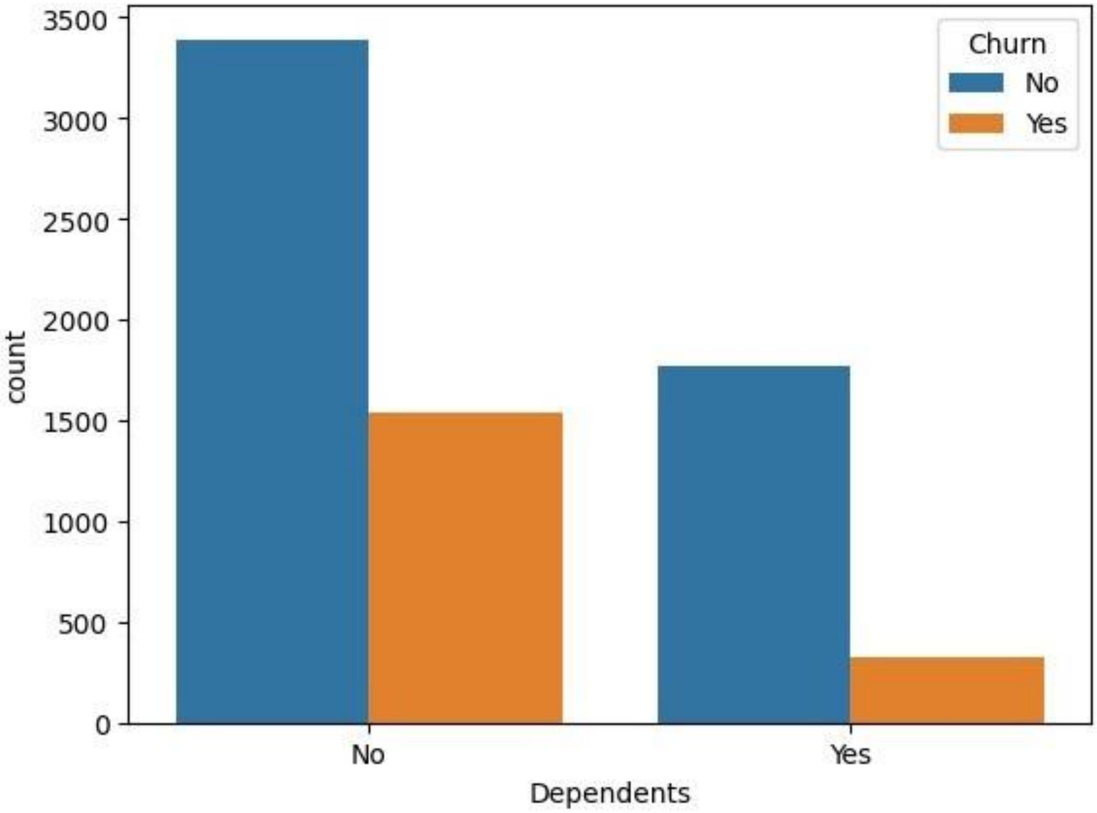
```
   gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  Female              0     Yes         No       1           No   
1    Male              0      No         No      34          Yes   
2    Male              0      No         No       2          Yes   
3    Male              0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


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

               PaymentMethod MonthlyCharges  TotalCharges  Churn
tenure_group  
0           Electronic check          29.85         29.85     No
1 - 12  
1               Mailed check          56.95       1889.50     No
25 - 36  
2               Mailed check          53.85        108.15    Yes
1 - 12  
3  Bank transfer (automatic)          42.30       1840.75     No
37 - 48  
4           Electronic check          70.70        151.65    Yes
1 - 12  

[5 rows x 21 columns]
```

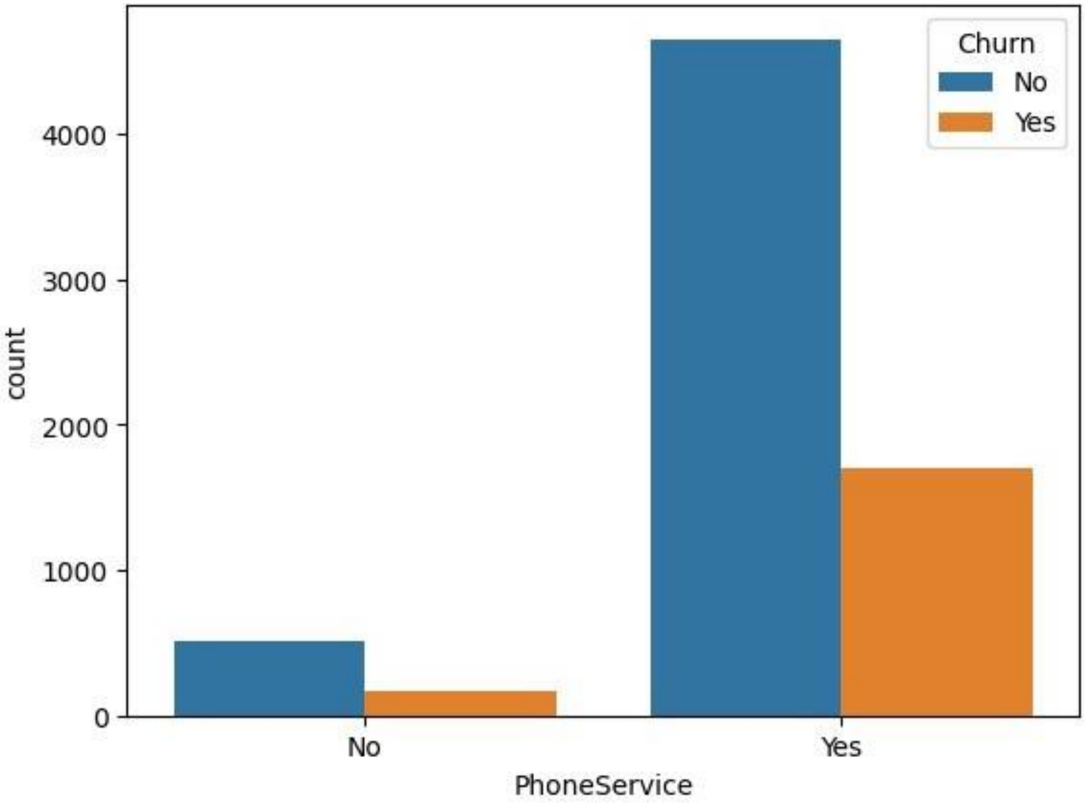
```
for i, predictor in
enumerate(telco_data.drop(columns=['Churn','TotalCharges','MonthlyChar
ges','tenure'])):
    plt.figure(i)
    sns.countplot(data=telco_data,x=predictor,hue='Churn')
```

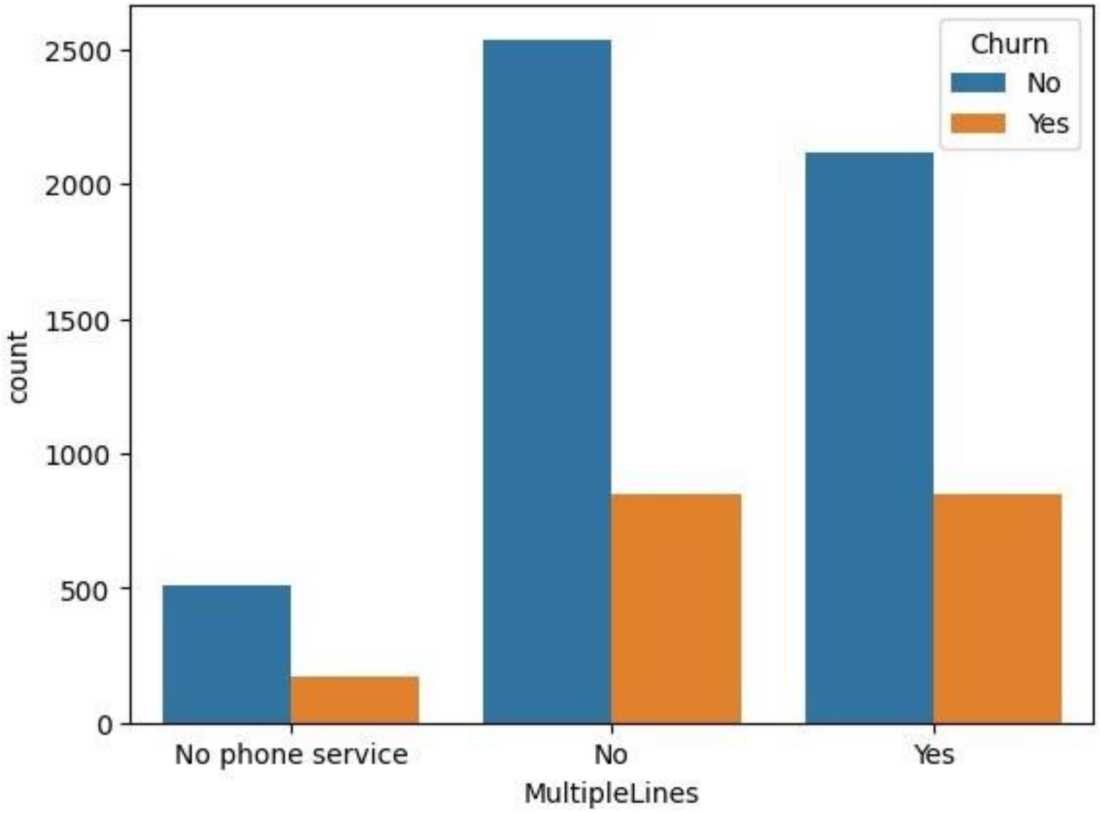


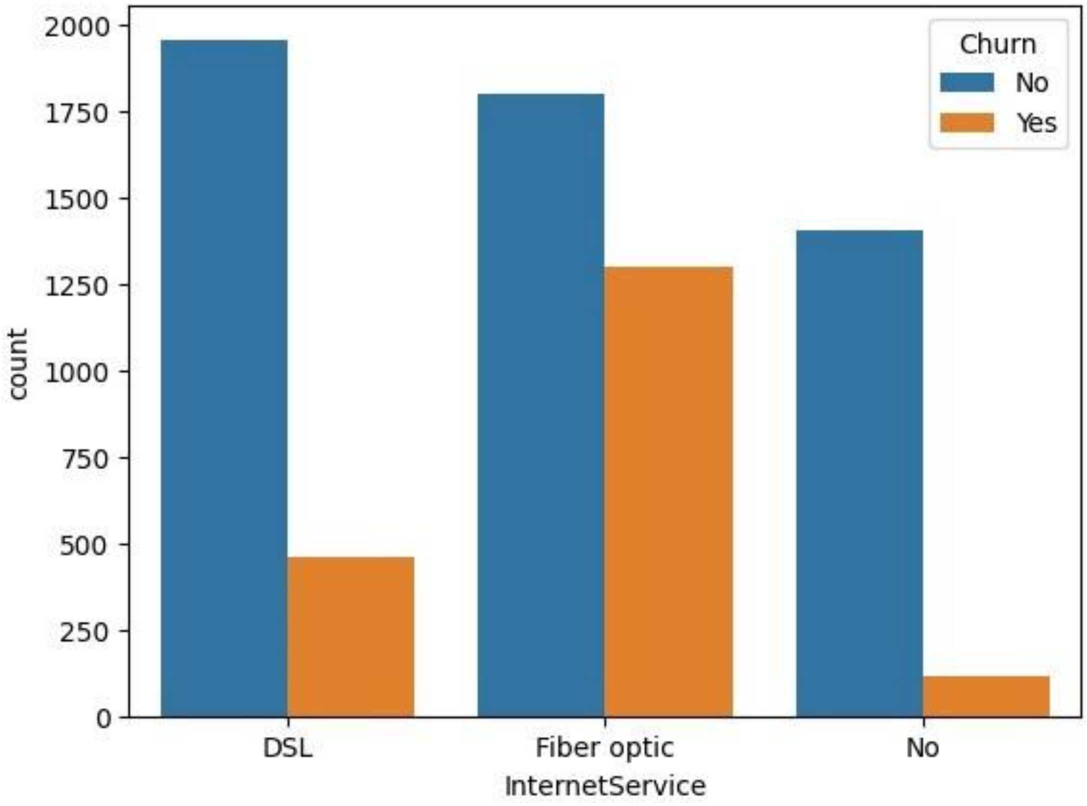

count
4000
3000
2000
1000
0
0
1
SeniorCitizen
Churn
No
Yes

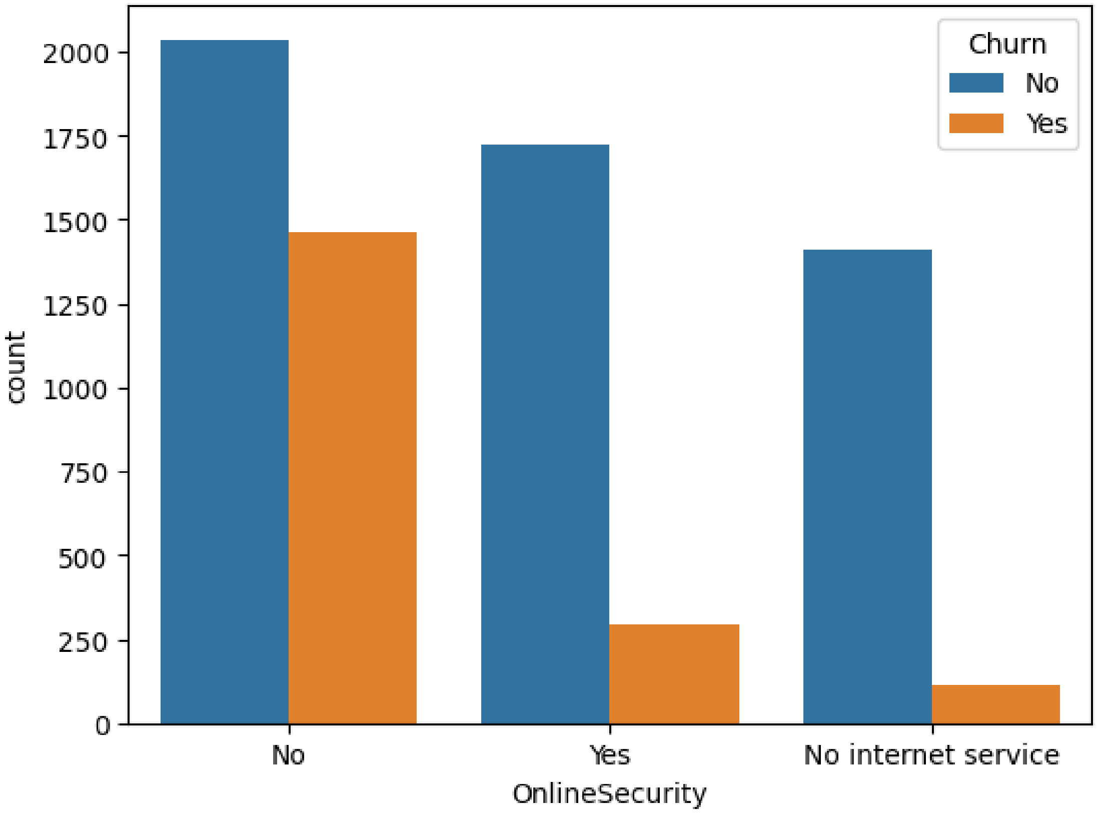

count
2500
2000
1500
1000
500
0
Yes
No
Partner
Churn
No
Yes


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

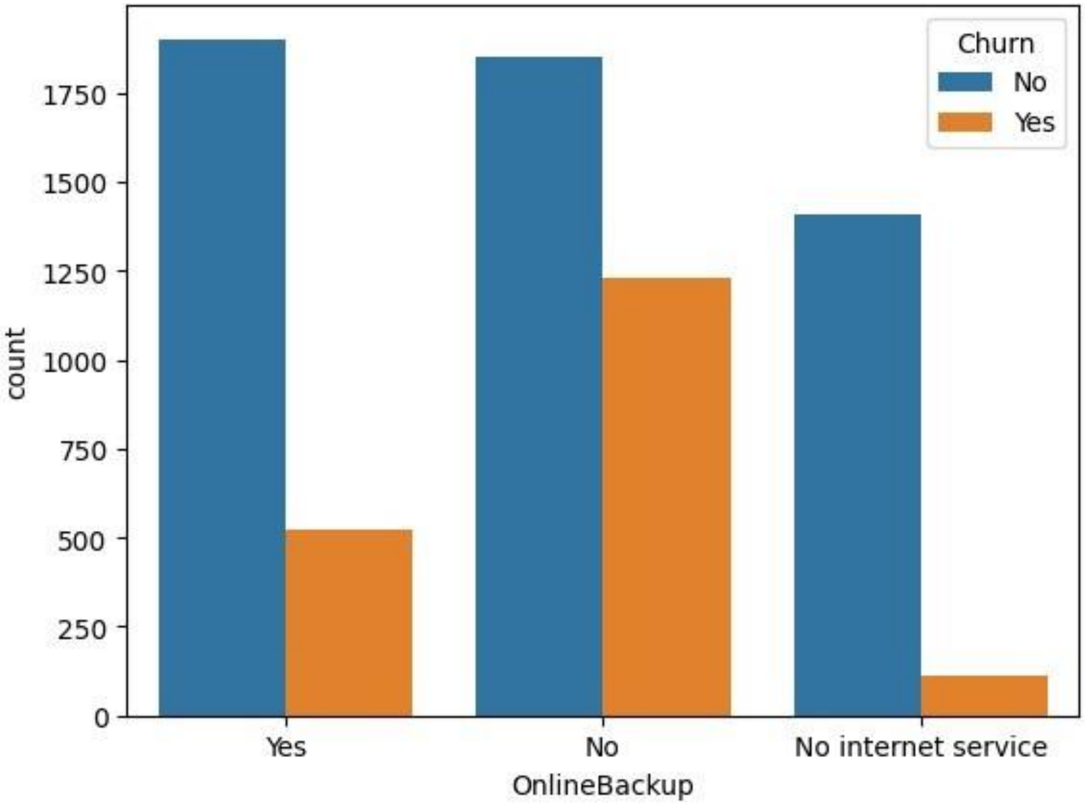

Churn
No
Yes
count
4000
3000
2000
1000
0
No
Yes
PhoneService


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

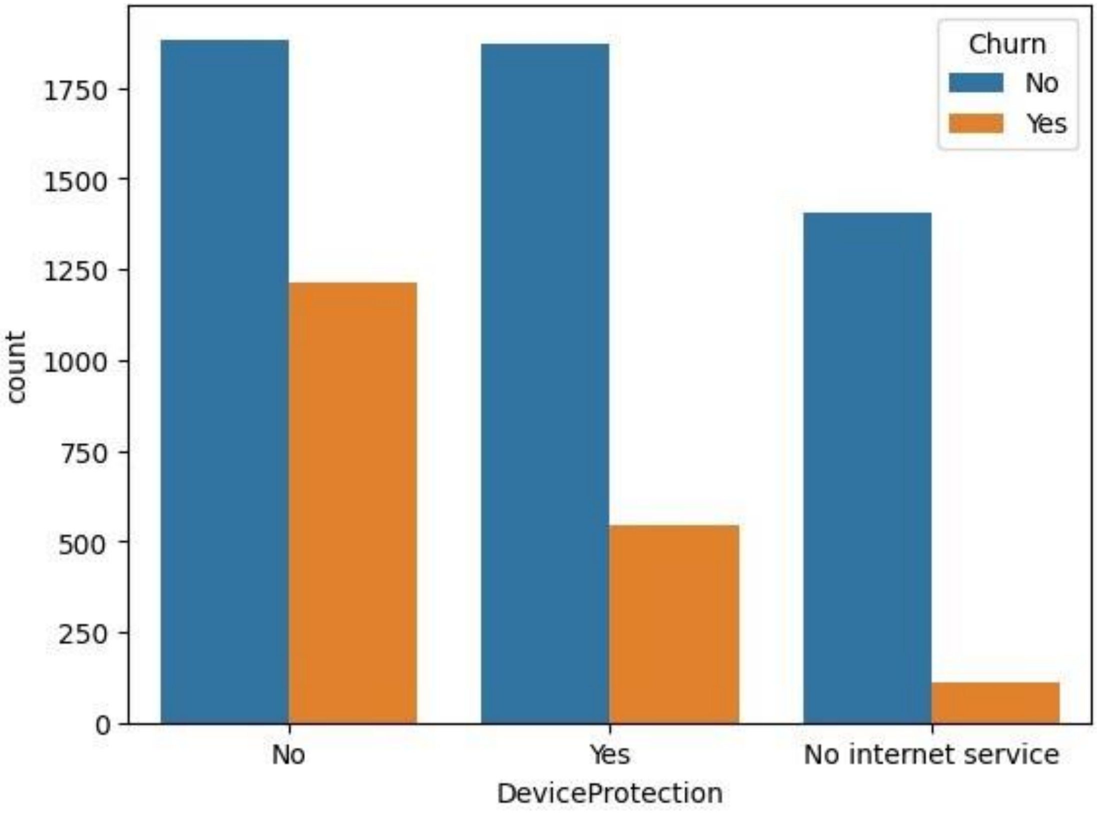

Churn
🟦 No
🟧 Yes
count
0
250
500
750
1000
1250
1500
1750
2000
DSL
Fiber optic
No
InternetService

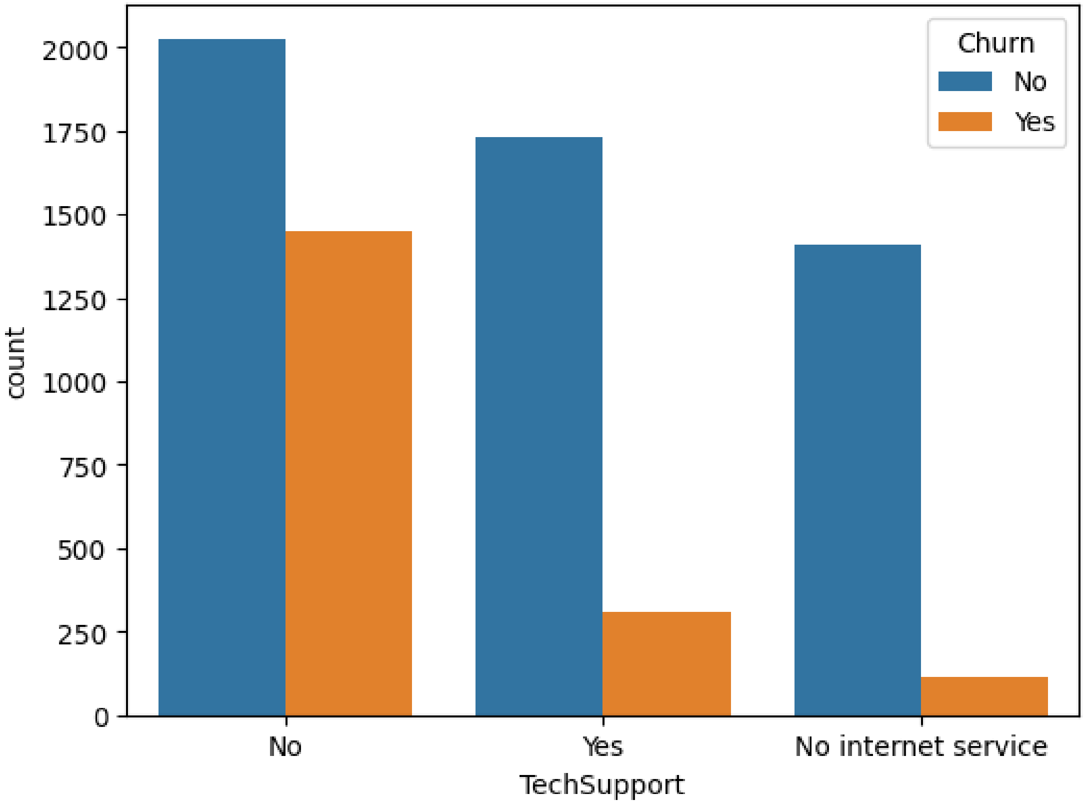


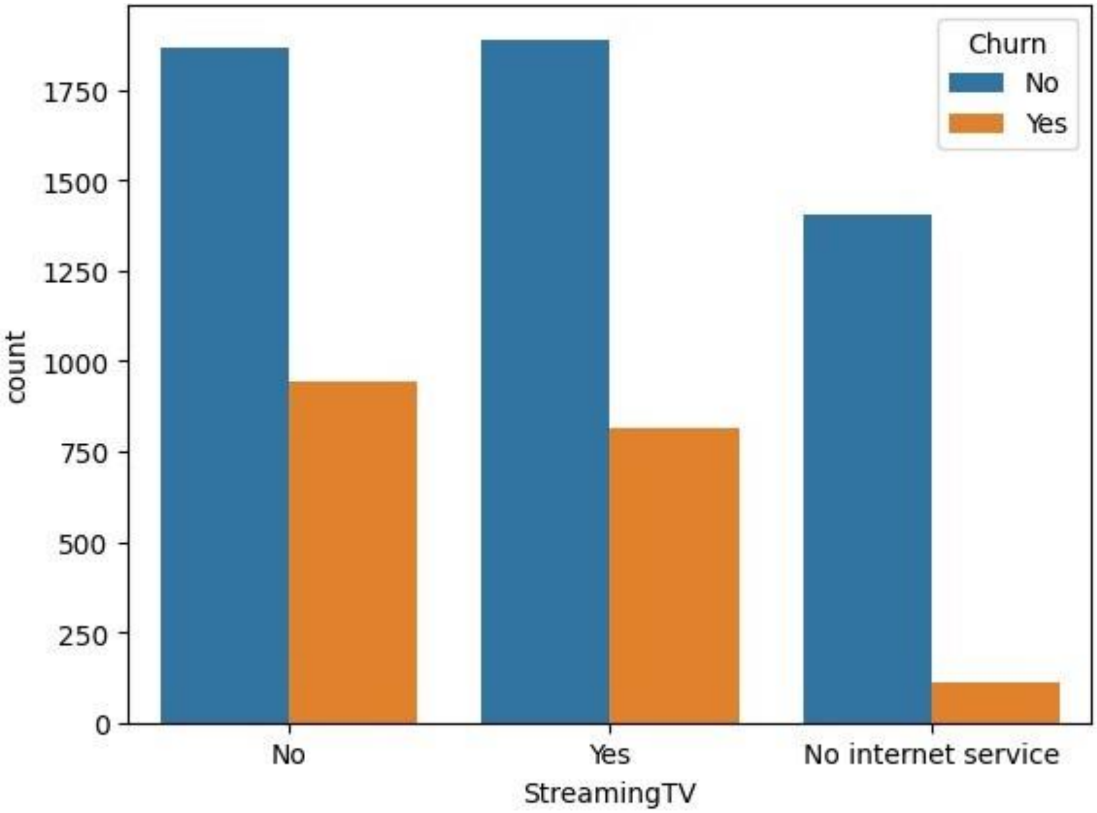
Churn
🟦 No
🟧 Yes
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
OnlineSecurity

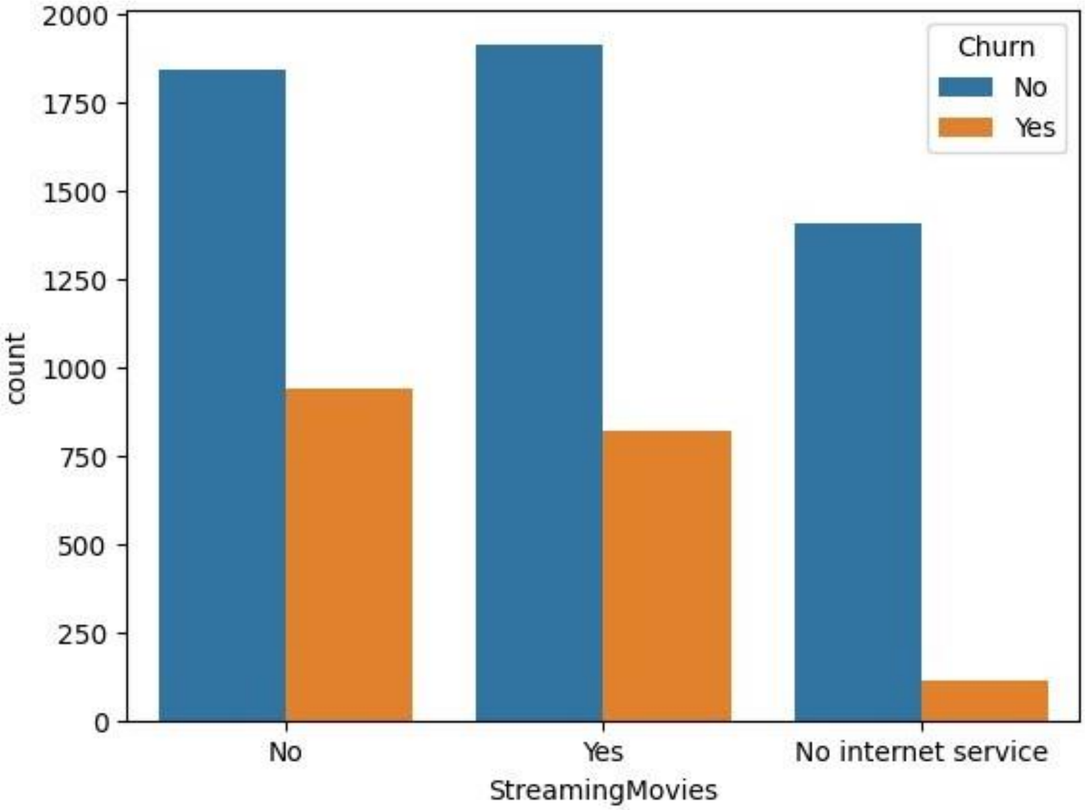


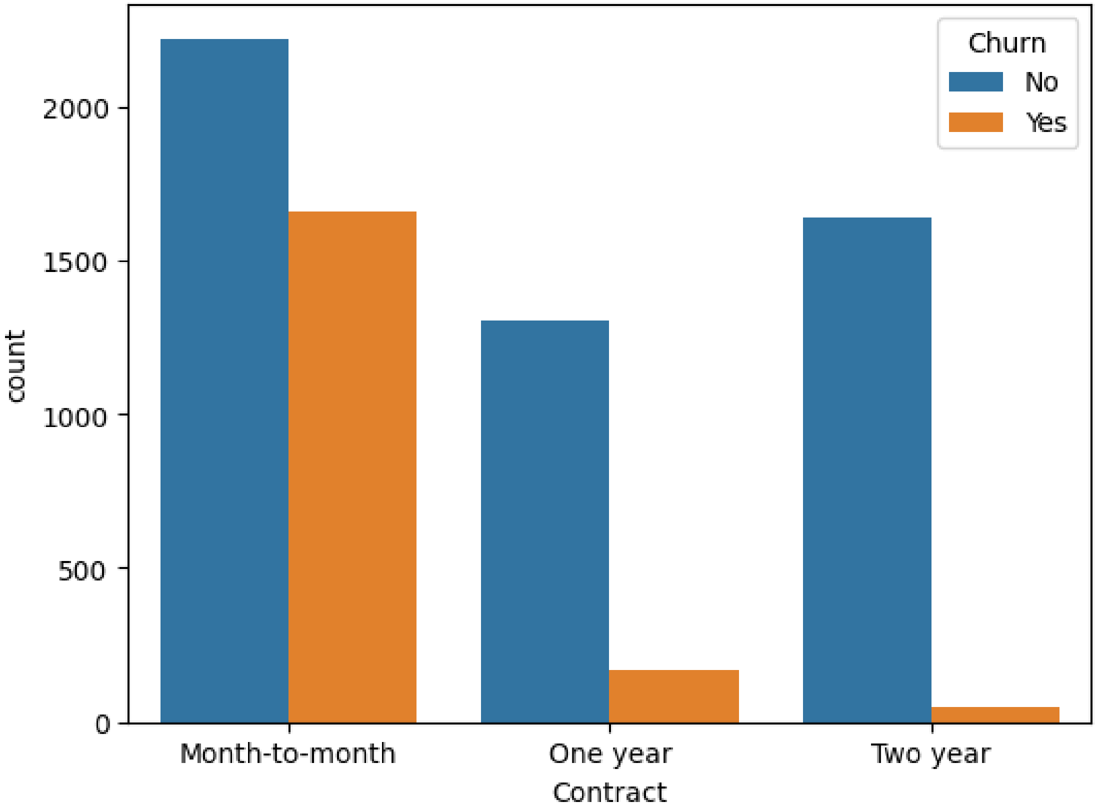
Churn
🟦 No
🟧 Yes
1750
1500
1250
1000
750
500
250
0
count
Yes
No
No internet service
OnlineBackup

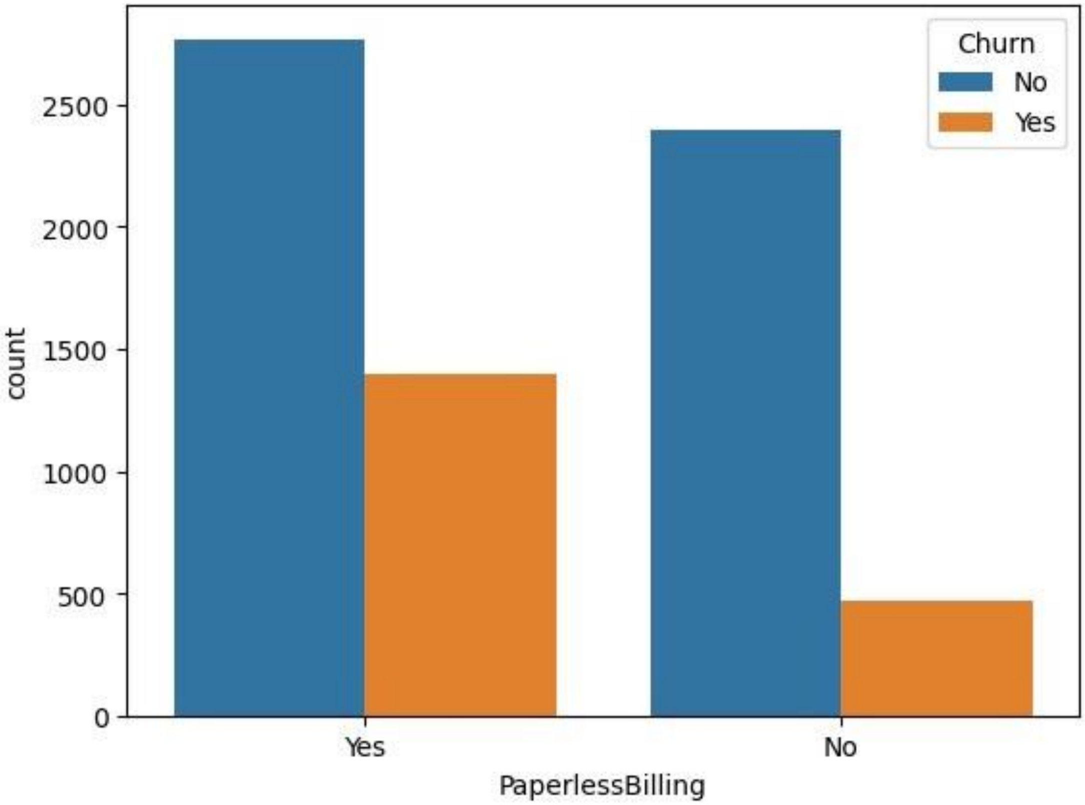

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
No
Yes
No internet service
DeviceProtection

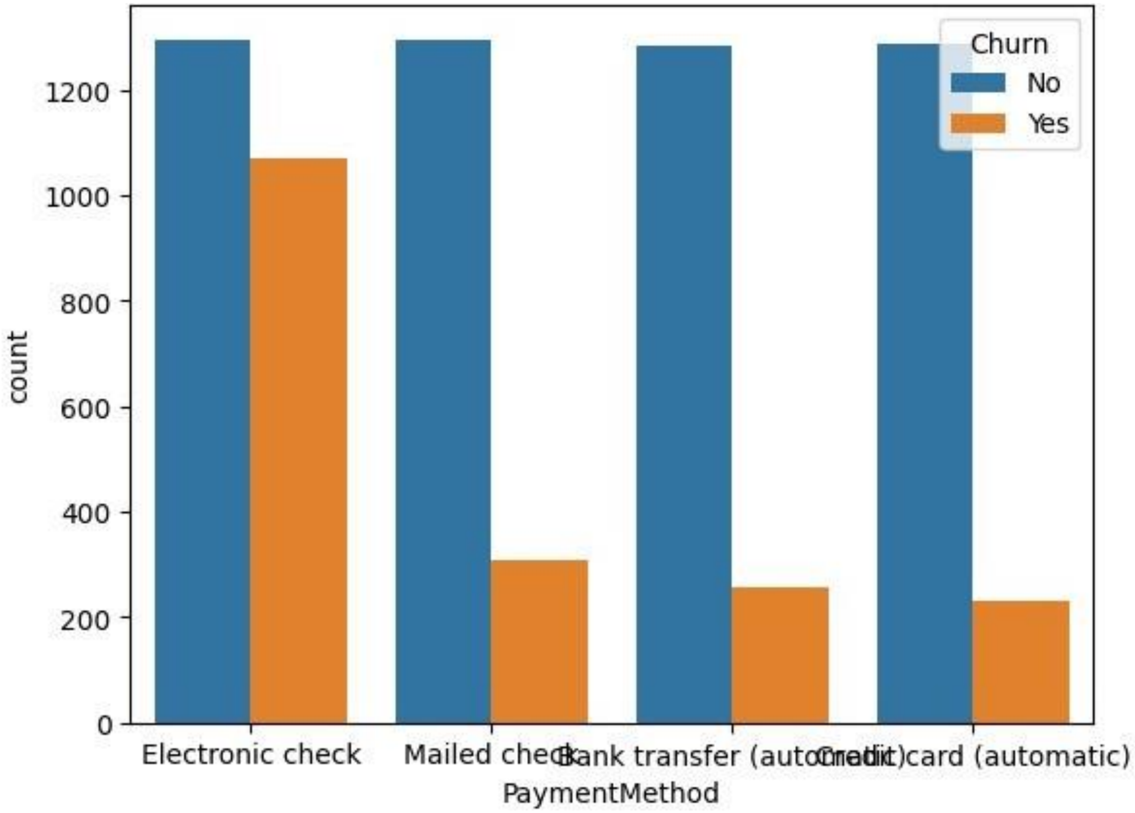

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
No
Yes
No internet service
TechSupport

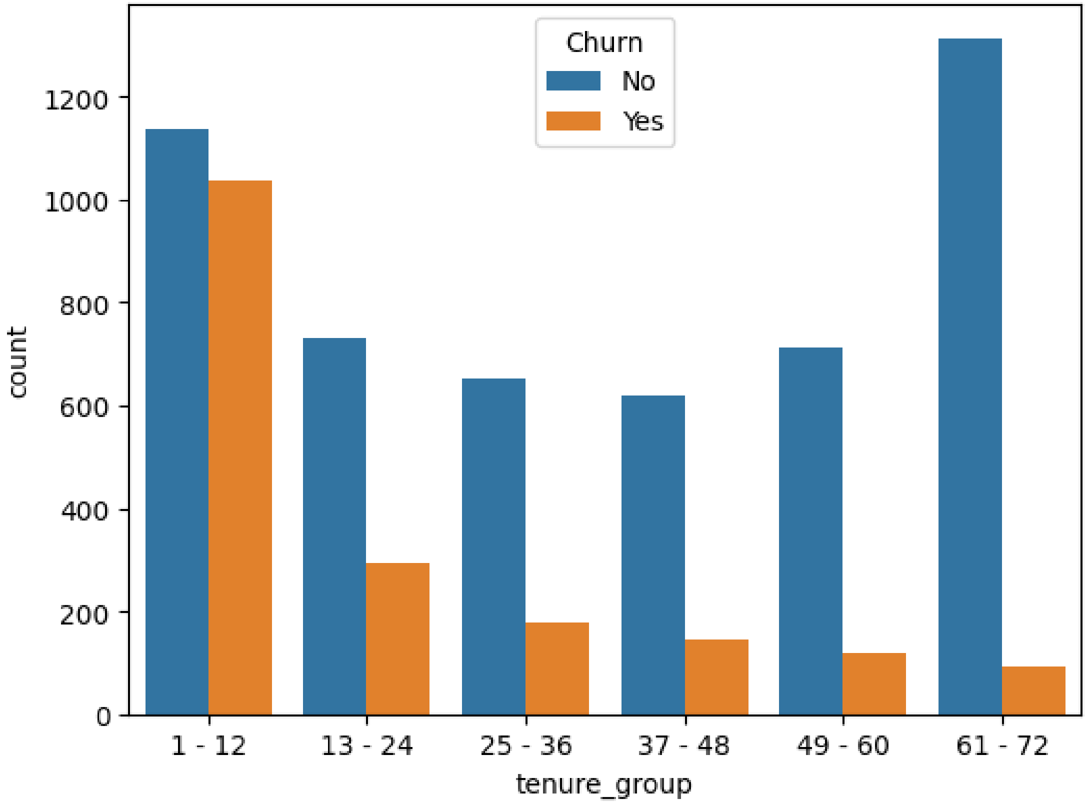

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
No
Yes
No internet service
StreamingTV


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


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


Churn
No
Yes
count
2500
2000
1500
1000
500
0
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


Churn
No
Yes
count
0
200
400
600
800
1000
1200
1 - 12
13 - 24
25 - 36
37 - 48
49 - 60
61 - 72
tenure_group

```
telco_data['Churn']=np.where(telco_data.Churn=='Yes',1,0)

telco_data.head()

   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  ...


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

              PaymentMethod MonthlyCharges  TotalCharges  Churn
tenure_group
0          Electronic check          29.85         29.85      0
1 - 12
1              Mailed check          56.95       1889.50      0
25 - 36
2              Mailed check          53.85        108.15      1
1 - 12
3  Bank transfer (automatic)          42.30       1840.75      0
37 - 48
4          Electronic check          70.70        151.65      1
1 - 12

[5 rows x 21 columns]
```

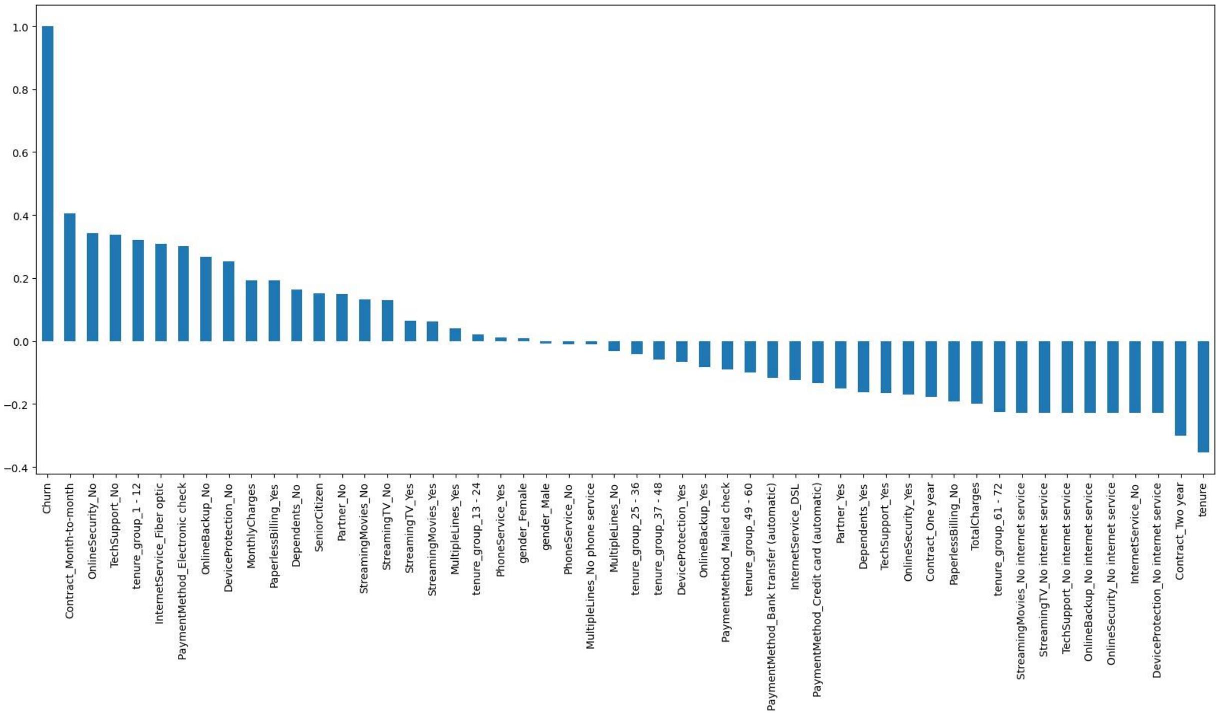
```
telco_data_dummies=pd.get_dummies(telco_data,dtype=int)
telco_data_dummies.head()
```

```
   SeniorCitizen  tenure  MonthlyCharges  TotalCharges  Churn  
gender_Female  \
0              0       1           29.85         29.85      0   
1
1              0      34           56.95       1889.50      0   
0
2              0       2           53.85        108.15      1   
0
3              0      45           42.30       1840.75      0   
0
4              0       2           70.70        151.65      1   
1

   gender_Male  Partner_No  Partner_Yes  Dependents_No  ...  \
0            0           0            1              1  ...   
1            1           1            0              1  ...   
2            1           1            0              1  ...   
3            1           1            0              1  ...   
4            0           1            0              1  ...   

   PaymentMethod_Bank transfer (automatic)  \
0                                        0   
1                                        0   
2                                        0   
3                                        1   
4                                        0   

   PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic 
check  \
0                                      0                           
1
1                                      0                           
0
2                                      0                           
0
3                                      0                           
0
4                                      0                           
1

   PaymentMethod_Mailed check  tenure_group_1 - 12  tenure_group_13 - 
24  \
0                           0                    1                    
0
1                           1                    0                    
0
2                           1                    1                    
```

```
0
3                              0                         0
0
4                              0                         1
0

   tenure_group_25 - 36  tenure_group_37 - 48  tenure_group_49 - 60  \
0                     0                     0                     0
1                     1                     0                     0
2                     0                     0                     0
3                     0                     1                     0
4                     0                     0                     0

   tenure_group_61 - 72
0                     0
1                     0
2                     0
3                     0
4                     0

[5 rows x 52 columns]
```

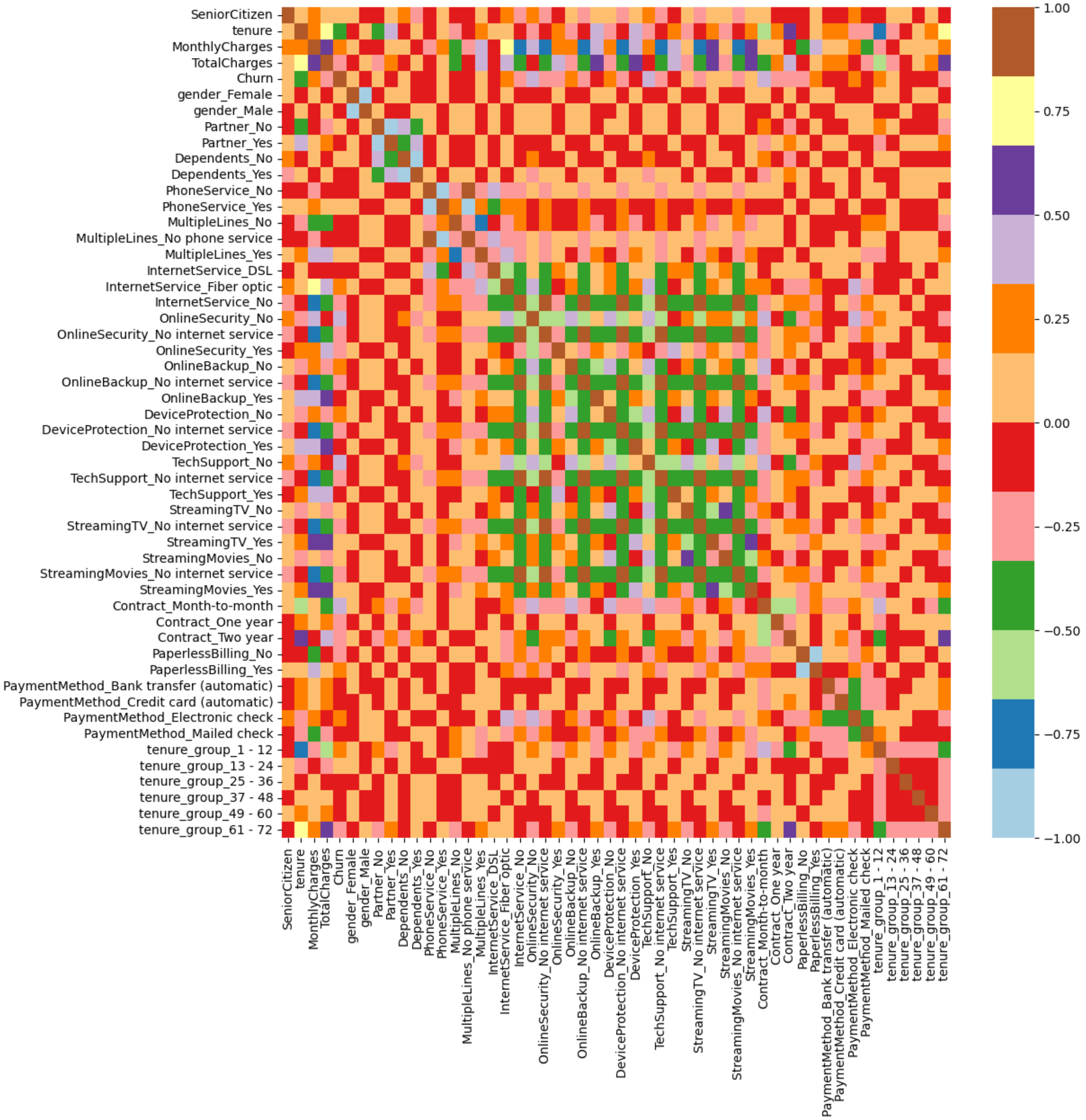
```
plt.figure(figsize=(20,8))
telco_data_dummies.corr()
['Churn'].sort_values(ascending=False).plot(kind='bar')
```

```
<Axes: >
```



```
plt.figure(figsize=(12,12))
sns.heatmap(telco_data_dummies.corr(),cmap="Paired")
```

```
<Axes: >
```



```
df_target_churn0=telco_data.loc[telco_data["Churn"]==0]
df_target_churn1=telco_data.loc[telco_data["Churn"]==1]
```

#model

```
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split

df=telco_data_dummies.copy()
df.head()
```

```
   SeniorCitizen  tenure  MonthlyCharges  TotalCharges  Churn  
gender_Female  \
0              0       1           29.85         29.85      0  
1   
1              0      34           56.95       1889.50      0  
0   
2              0       2           53.85        108.15      1  
0   
3              0      45           42.30       1840.75      0  
0   
4              0       2           70.70        151.65      1  
1   

   gender_Male  Partner_No  Partner_Yes  Dependents_No  ...  \
0            0           0            1              1  ...   
1            1           1            0              1  ...   
2            1           1            0              1  ...   
3            1           1            0              1  ...   
4            0           1            0              1  ...   

   PaymentMethod_Bank transfer (automatic)  \
0                                        0   
1                                        0   
2                                        0   
3                                        1   
4                                        0   

   PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic 
check  \
0                                      0  
1   
1                                      0  
0   
2                                      0  
0   
3                                      0  
0   
4                                      0  
1   

   PaymentMethod_Mailed check  tenure_group_1 - 12  tenure_group_13 - 
24  \
0                           0                    1  
```

```
0
1                              1                      0
0
2                              1                      1
0
3                              0                      0
0
4                              0                      1
0

   tenure_group_25 - 36  tenure_group_37 - 48  tenure_group_49 - 60  \
0                     0                     0                     0
1                     1                     0                     0
2                     0                     0                     0
3                     0                     1                     0
4                     0                     0                     0

   tenure_group_61 - 72
0                     0
1                     0
2                     0
3                     0
4                     0

[5 rows x 52 columns]
```

```
X=df.drop("Churn",axis=1)
Y=df['Churn']
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,rando
m_state=42)
```

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier(criterion='gini',random_state=100,max_depth=
6,min_samples_leaf=8)
dt.fit(X_train,Y_train)
```

```
DecisionTreeClassifier(max_depth=6, min_samples_leaf=8,
random_state=100)
```

```
Y_predict=dt.predict(X_test)
```

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
print(classification_report(Y_test,Y_predict))
print(confusion_matrix(Y_test,Y_predict))
```

```
              precision    recall  f1-score   support

           0       0.84      0.85      0.85      1033
           1       0.58      0.56      0.57       374
```

```
    accuracy                           0.78      1407
   macro avg       0.71      0.71      0.71      1407
weighted avg       0.77      0.78      0.78      1407

[[883 150]
 [164 210]]
```

```python
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100)
rf.fit(X_train,Y_train)
Y_rf_pred=rf.predict(X_test)
print(classification_report(Y_test,Y_rf_pred))
print(confusion_matrix(Y_test,Y_rf_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.89      0.85      1033
           1       0.59      0.45      0.51       374

    accuracy                           0.77      1407
   macro avg       0.71      0.67      0.68      1407
weighted avg       0.76      0.77      0.76      1407

[[918 115]
 [206 168]]
```

```python
from imblearn.combine import SMOTEENN
sm=SMOTEENN()
X_resampled,Y_resampled=sm.fit_resample(X,Y)
Xr_train,Xr_test,Yr_train,Yr_test=train_test_split(X_resampled,Y_resampled,test_size=0.2,random_state=42)
print(Y_resampled.value_counts())
print(Y.value_counts())
```

```
Churn
1    3196
0    2652
Name: count, dtype: int64
Churn
0    5163
1    1869
Name: count, dtype: int64
```

```python
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier(criterion='gini',random_state=100,max_depth=6,min_samples_leaf=8)
dt.fit(Xr_train,Yr_train)
Yr_predict=dt.predict(Xr_test)
print(classification_report(Yr_test,Yr_predict))
print(confusion_matrix(Yr_test,Yr_predict))
```

```
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       556
           1       0.94      0.93      0.94       614

    accuracy                           0.93      1170
   macro avg       0.93      0.93      0.93      1170
weighted avg       0.93      0.93      0.93      1170

[[520  36]
 [ 43 571]]
```

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100)
rf.fit(Xr_train,Yr_train)
Yr_rf_pred=rf.predict(Xr_test)
print(classification_report(Yr_test,Yr_rf_pred))
print(confusion_matrix(Yr_test,Yr_rf_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       556
           1       0.96      0.97      0.97       614

    accuracy                           0.96      1170
   macro avg       0.96      0.96      0.96      1170
weighted avg       0.96      0.96      0.96      1170

[[532  24]
 [ 18 596]]
```

```
from sklearn.neural_network import MLPClassifier

# Model 6: Neural Network Classifier
print("\nModel 6: Neural Network Classifier")
model_nn = MLPClassifier(hidden_layer_sizes=(100,), max_iter=1000)
model_nn.fit(Xr_train, Yr_train)
yr_pred_nn = model_nn.predict(Xr_test)
print("Classification Report:")
print(classification_report(Yr_test, yr_pred_nn))
print("Confusion Matrix:")
print(confusion_matrix(Yr_test, yr_pred_nn))
```

```

Model 6: Neural Network Classifier
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.94       556
           1       0.95      0.94      0.94       614
```

```
    accuracy                           0.94      1170
   macro avg       0.94      0.94      0.94      1170
weighted avg       0.94      0.94      0.94      1170

Confusion Matrix:
[[523  33]
 [ 37 577]]
```

```python
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Define pipeline
pipe_dt = Pipeline([
    ('clf', DecisionTreeClassifier())
])

# Define parameter grid
param_grid_dt = {
    'clf__criterion': ['gini', 'entropy'],
    'clf__max_depth': [None, 10, 20, 30, 40, 50],
    'clf__min_samples_split': [2, 5, 10],
    'clf__min_samples_leaf': [1, 2, 4],
    'clf__max_features': ['sqrt', 'log2', None]
}

# Perform GridSearchCV
grid_dt = GridSearchCV(pipe_dt, param_grid_dt, cv=5)
grid_dt.fit(Xr_train, Yr_train)

# Print best parameters
print("Best Parameters (GridSearchCV):", grid_dt.best_params_)

# Predict on the testing set using the best model
best_classifier_dt = grid_dt.best_estimator_
yrr_pred_dt = best_classifier_dt.predict(Xr_test)

# Evaluate the model
print("\nClassification Report:")
print(classification_report(Yr_test, yrr_pred_dt))

print("\nConfusion Matrix:")
print(confusion_matrix(Yr_test, yrr_pred_dt))
```

```
Best Parameters (GridSearchCV): {'clf  criterion': 'gini',
'clf  max_depth': 10, 'clf  max_features': None,
'clf__min_samples_leaf': 2, 'clf__min_samples_split': 10}

Classification Report:
              precision    recall  f1-score   support
```

```
           0       0.93      0.94      0.93       556
           1       0.94      0.93      0.94       614

    accuracy                           0.94      1170
   macro avg       0.94      0.94      0.94      1170
weighted avg       0.94      0.94      0.94      1170


Confusion Matrix:
[[522  34]
 [ 41 573]]
```

```
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder



# Initialize and train Random Forest classifier
rf_clf = RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
grid_search = GridSearchCV(rf_clf, param_grid, cv=5,
scoring='accuracy')
grid_search.fit(Xr_train, Yr_train)

# Get the best estimator
best_rf_clf = grid_search.best_estimator_

# Evaluate on test data
Yrgd_pred = best_rf_clf.predict(Xr_test)
print(classification_report(Yr_test, Yrgd_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       556
           1       0.96      0.97      0.97       614

    accuracy                           0.96      1170
   macro avg       0.96      0.96      0.96      1170
weighted avg       0.96      0.96      0.96      1170
```

```python
from xgboost import XGBClassifier

# Model 7: XGBoost Classifier
print("\nModel 7: XGBoost Classifier")
model_xgb = XGBClassifier()
model_xgb.fit(Xr_train, Yr_train)
yr_pred_xgb = model_xgb.predict(Xr_test)
print("Classification Report:")
print(classification_report(Yr_test, yr_pred_xgb))
print("Confusion Matrix:")
print(confusion_matrix(Yr_test, yr_pred_xgb))
```

```
Model 7: XGBoost Classifier
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.96       556
           1       0.97      0.96      0.97       614

    accuracy                           0.97      1170
   macro avg       0.97      0.97      0.97      1170
weighted avg       0.97      0.97      0.97      1170

Confusion Matrix:
[[538  18]
 [ 22 592]]
```