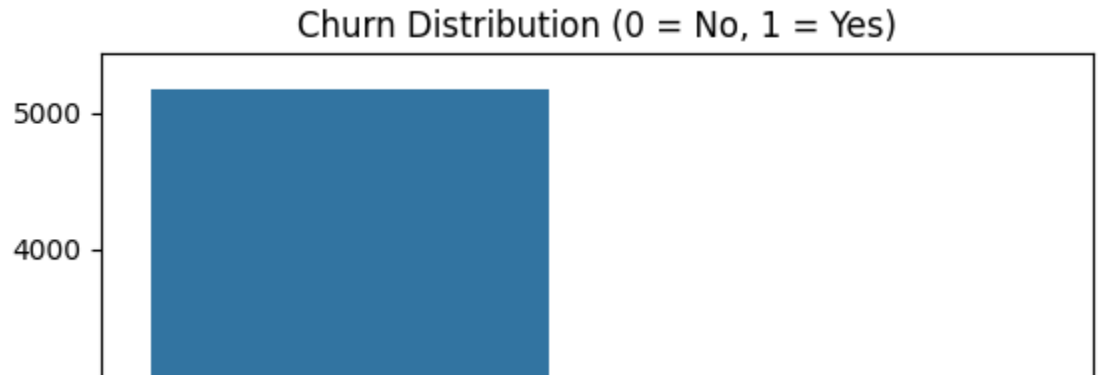
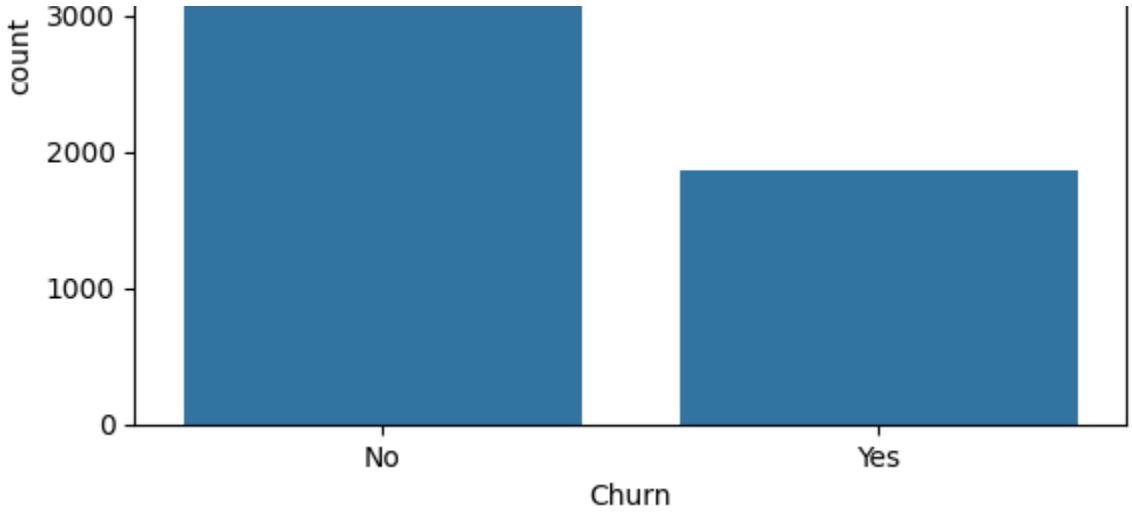
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
print(df.head())

# Check target distribution
sns.countplot(x='Churn', data=df)
plt.title('Churn Distribution (0 = No, 1 = Yes)')
plt.show()
```

```
   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]
```






```
df.dropna(inplace=True)
```

```
import sklearn
from sklearn.preprocessing import LabelEncoder

# Replace ... with the actual categorical column names from your dataframe 'df'
cat_cols = ['gender', 'Partner', 'Contract', 'Dependents', 'PhoneService', 'MultipleLines
            'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'Streami
            'StreamingMovies', 'PaperlessBilling', 'PaymentMethod', 'TotalCharges', 'Chur

# Now apply Label Encoding to these specific columns
for col in cat_cols:
    if col in df.columns:  # Check if column exists in DataFrame
        df[col] = LabelEncoder().fit_transform(df[col])
    else:
        print(f"Column '{col}' not found in DataFrame. Skipping...")
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['tenure', 'MonthlyCharges']] = scaler.fit_transform(df[['tenure', 'MonthlyCharges']])
```

```
from sklearn.model_selection import train_test_split

X = df.drop(['customerID', 'Churn'], axis=1)
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
import sklearn
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
```

```
# Assuming 'df' is your DataFrame

import sklearn
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

# Assuming 'df' is your DataFrame

# 1. Identify categorical and numerical features
categorical_features = ['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines
                        'DeviceProtection', 'TechSupport', 'StreamingTV', 'Streaming']


import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier

# Assuming 'df' is your DataFrame loaded from 'WA_Fn-UseC_-Telco-Customer-Churn.csv'

# Check if 'customerID' is in the columns before dropping
if 'customerID' in df.columns:
    df = df.drop(['customerID'], axis=1)
else:
    print("Column 'customerID' already dropped or not present.")

# 2. Identify categorical and numerical features
categorical_features = ['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines
                        'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
                        'Contract', 'PaperlessBilling', 'PaymentMethod', 'InternetService'
numerical_features = ['tenure', 'MonthlyCharges', 'TotalCharges']

# 3. Create preprocessing pipelines
numerical_pipeline = Pipeline([('scaler', StandardScaler())])
categorical_pipeline = Pipeline([('onehot', OneHotEncoder(handle_unknown='ignore'))])

# 4. Combine pipelines using ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_pipeline, numerical_features),
        ('cat', categorical_pipeline, categorical_features)
    ])
```

```
 Column 'customerID' already dropped or not present.
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
```

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from xgboost import XGBClassifier


# ... (previous code) ...

# 1. Include 'InternetService' in Label Encoding
cat_cols = ['gender', 'Partner', 'Contract', 'Dependents', 'PhoneService', 'MultipleLines
            'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'Streami
            'StreamingMovies', 'PaperlessBilling', 'PaymentMethod', 'TotalCharges', 'Chur
            'InternetService']  # Add InternetService here

for col in cat_cols:
    if col in df.columns:
        df[col] = LabelEncoder().fit_transform(df[col])
    else:
        print(f"Column '{col}' not found in DataFrame. Skipping...")

# ... (rest of your code, including preprocessor and model training) ...

xgb = XGBClassifier(scale_pos_weight=(len(y_train) - sum(y_train)) / sum(y_train))
# Apply preprocessor to X_train and X_test
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)

# Fit the model using the processed data
xgb.fit(X_train_processed, y_train)
print(f"XGB Accuracy: {xgb.score(X_test_processed, y_test):.2f}")
```

```
XGB Accuracy: 0.76
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from xgboost import XGBClassifier


# ... (previous code) ...

# 1. Include 'InternetService' in Label Encoding
cat_cols = ['gender', 'Partner', 'Contract', 'Dependents', 'PhoneService', 'MultipleLines
            'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'Streami
            'StreamingMovies', 'PaperlessBilling', 'PaymentMethod', 'TotalCharges', 'Chur
            'InternetService']  # Add InternetService here

# Apply Label Encoding to the entire DataFrame before splitting
for col in cat_cols:
    if col in df.columns:
```

```
    if col in df.columns:
        df[col] = LabelEncoder().fit_transform(df[col])
    else:
        print(f"Column '{col}' not found in DataFrame. Skipping...")


# ... (rest of your code, including preprocessor and model training) ...

# Apply preprocessor to X_train and X_test
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)

# Fit the model using the processed data
xgb.fit(X_train_processed, y_train)
print(f"XGB Accuracy: {xgb.score(X_test_processed, y_test):.2f}")

# ... (prediction and evaluation) ...

# Predict using preprocessed data
y_pred = xgb.predict(X_test_processed)  # Use X_test_processed for prediction
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
XGB Accuracy: 0.76
[[822 214]
 [119 254]]
              precision    recall  f1-score   support

           0       0.87      0.79      0.83      1036
           1       0.54      0.68      0.60       373

    accuracy                           0.76      1409
   macro avg       0.71      0.74      0.72      1409
weighted avg       0.79      0.76      0.77      1409
```

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

# Pie chart
df['Churn'].value_counts().plot.pie(autopct='%1.1f%%', explode=[0, 0.1],
                                    colors=['#66b3ff', '#ff9999'], ax=ax1)
ax1.set_title('Churn Proportion')

# Bar plot
sns.countplot(x='Churn', data=df, palette=['#66b3ff', '#ff9999'], ax=ax2)
ax2.set_title('Churn Count')
plt.show()
```

```
<ipython-input-21-c367215fad3f>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.
```

```
sns.countplot(x='Churn', data=df, palette=['#66b3ff', '#ff9999'], ax=ax2)
```