```
import pandas as pd
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset, assuming it's a CSV despite the .xls extension
try:
    df = pd.read_csv('/content/WA_Fn-UseC_-Telco-Customer-Churn.xls')
    print("File loaded successfully as CSV.")
except Exception as e:
    print(f"Error loading file as CSV: {e}")
    print("Attempting to load as Excel with openpyxl engine (for modern .xlsx)
    try:
        df = pd.read_excel('/content/WA_Fn-UseC_-Telco-Customer-Churn.xls', en
        print("File loaded successfully as Excel.")
    except Exception as e_excel:
        print(f"Error loading file as Excel: {e_excel}")
        print("Could not load the file. Please ensure it's a valid CSV or Exce
        exit() # Exit if file cannot be loaded

# Display the first few rows and info
print(df.head())
print(df.info())

# --- Data Preprocessing for XGBoost ---

# Drop 'customerID' column as it's not useful for prediction
df = df.drop('customerID', axis=1)

# Convert 'TotalCharges' to numeric. It might have spaces or non-numeric value
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
# Handle missing 'TotalCharges' values (if any) by filling with the mean or me
df['TotalCharges'] = df['TotalCharges'].fillna(df['TotalCharges'].mean())

# Convert 'Churn' target variable to numerical (Yes=1, No=0)
df['Churn'] = df['Churn'].apply(lambda x: 1 if x == 'Yes' else 0)

# Identify categorical and numerical features
categorical_features = df.select_dtypes(include='object').columns
numerical_features = df.select_dtypes(include=['int64', 'float64']).columns.dr

# Apply One-Hot Encoding to categorical features
df_encoded = pd.get_dummies(df, columns=categorical_features, drop_first=True)

# Separate features (X) and target (y)
X = df_encoded.drop('Churn', axis=1)
```

```
X = df_encoded.drop('Churn', axis=1)
y = df_encoded['Churn']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

# --- XGBoost Model Training ---

# Initialize XGBoost Classifier
xgb_model = xgb.XGBClassifier(
    objective='binary:logistic', # For binary classification
    eval_metric='logloss',       # Evaluation metric
    n_estimators=100,            # Number of boosting rounds
    learning_rate=0.1,           # Step size shrinkage
    max_depth=3,                 # Maximum depth of a tree
    random_state=42
)

# Train the model
xgb_model.fit(X_train, y_train)

# --- Model Evaluation ---

# Make predictions on the test set
y_pred = xgb_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"\nAccuracy: {accuracy:.4f}")
print("\nClassification Report:")
print(report)
```

```
File loaded successfully as CSV.
   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
```

```
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]
<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
```