Telco Customer Churn Prediction — Data Preprocessing

Step 1: Load Cleaned Data

Load the dataset prepared in the previous EDA step.

In [11]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

# Load data
df = pd.read_csv('../data/WA_Fn-UseC_-Telco-Customer-Churn.csv')

# Convert TotalCharges

# Just in case
if df['TotalCharges'].dtype == 'object':
    df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
    df['TotalCharges'].fillna(df['TotalCharges'].mean(), inplace=True)

# Map churn to binary
df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})
```

Step 2: Separate Features

Separate categorical and numerical columns.

In [12]:
```
# Separate feature types
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()

# Remove customerID from categorical columns
categorical_cols.remove('customerID')

numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
numerical_cols.remove('Churn')  # target variable

print("Categorical columns:", categorical_cols)
print("Numerical columns:", numerical_cols)
```

```
Categorical columns: ['gender', 'Partner', 'Dependents', 'PhoneService', 'Multipl
eLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
'PaymentMethod']
Numerical columns: ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']
```

Step 3: Encode Categorical Features

In [13]:
```
# Label encode binary categorical columns
binary_cols = [col for col in categorical_cols if df[col].nunique() == 2]
le = LabelEncoder()
for col in binary_cols:
    df[col] = le.fit_transform(df[col])

# One-hot encode multi-category columns
multi_cat_cols = [col for col in categorical_cols if df[col].nunique() > 2]
df = pd.get_dummies(df, columns=multi_cat_cols, drop_first=True)

df.head()
```

Out[13]:

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

5 rows × 32 columns

Step 4: Scale Numerical Features

In [14]:
```
scaler = StandardScaler()
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
df.head()
```

Out[14]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	P
0	7590-VHVEG	0	-0.439916	1	0	-1.277445	0	
1	5575-GNVDE	1	-0.439916	0	0	0.066327	1	
2	3668-QPYBK	1	-0.439916	0	0	-1.236724	1	
3	7795-CFOCW	1	-0.439916	0	0	0.514251	0	
4	9237-HQITU	0	-0.439916	0	0	-1.236724	1	

5 rows × 32 columns

Step 5: Split into Training and Test Sets

In [15]:
```
# Split features and target
X = df.drop(['customerID', 'Churn'], axis=1)
y = df['Churn']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

print(f"Training set size: {X_train.shape}")
print(f"Test set size: {X_test.shape}")
```

```
Training set size: (5634, 30)
Test set size: (1409, 30)
```

In [16]:
```
# Optional but recommended: save preprocessed dataset
df.to_csv('../data/preprocessed_telco.csv', index=False)
print("Preprocessed dataset saved as preprocessed_telco.csv in data folder.")
```

```
Preprocessed dataset saved as preprocessed_telco.csv in data folder.
```