DSC 680 Project 1

December 18, 2025

[3]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
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.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
```

[4]:
```python
#load flat file dataset & select columns needed for analysis
telecom_data = pd.read_csv('/Volumes/Editing/Bellevue Univ/Masters in Data␣
↪Science/DSC 680 Applied Data Science/Project 1/
↪WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

[5]:
```python
telecom_data.head()
```

[5]:
```
   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]
```

0.1 Data Cleaning

[6]:
```python
# Count of rows/columns
print(telecom_data.shape)
```

```
(7043, 21)
```

[7]:
```python
telecom_data.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
```

[8]:
```
telecom_data['Churn'].value_counts()
```

[8]:
```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
```

[9]:
```
telecom_data['gender'].value_counts()
```

[9]:
```
gender
Male      3555
Female    3488
Name: count, dtype: int64
```

[10]:
```
telecom_data['Churn'].value_counts(normalize=True)
```

[10]:
```
Churn
No     0.73463
Yes    0.26537
Name: proportion, dtype: float64
```

[11]:
```
# Convert Total Charges to Numeric data type to determine nulls
telecom_data['TotalCharges'] = pd.to_numeric(telecom_data['TotalCharges'],␣
 ↪errors='coerce')
telecom_data['MonthlyCharges'] = pd.to_numeric(telecom_data['MonthlyCharges'],␣
 ↪errors='coerce')
```

[12]:
```
# Count of missing values
telecom_data.isnull().sum()
```

[12]:
```
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
```

11 Null values found

[13]:
```
# Impute Using tenure × MonthlyCharges
telecom_data.loc[telecom_data['TotalCharges'].isna(), 'TotalCharges'] = (
    telecom_data.loc[telecom_data['TotalCharges'].isna(), 'tenure'] *
    telecom_data.loc[telecom_data['TotalCharges'].isna(), 'MonthlyCharges'])
```

[14]:
```
# Final Check to verify missing values
telecom_data['TotalCharges'].isna().sum()
```

[14]:
```
np.int64(0)
```

[15]:
```
telecom_data.describe
```

[15]:
```
<bound method NDFrame.describe of       customerID  gender  SeniorCitizen
Partner Dependents  tenure  \
0     7590-VHVEG  Female              0     Yes         No       1
1     5575-GNVDE    Male              0      No         No      34
2     3668-QPYBK    Male              0      No         No       2
3     7795-CFOCW    Male              0      No         No      45
4     9237-HQITU  Female              0      No         No       2
…            …     …             …     …        …      …
7038  6840-RESVB    Male              0     Yes        Yes      24
7039  2234-XADUH  Female              0     Yes        Yes      72
7040  4801-JZAZL  Female              0     Yes        Yes      11
7041  8361-LTMKD    Male              1     Yes         No       4
7042  3186-AJIEK    Male              0      No         No      66

     PhoneService     MultipleLines InternetService OnlineSecurity  …  \
0              No  No phone service             DSL             No  …
1             Yes                No             DSL            Yes  …
2             Yes                No             DSL            Yes  …
3              No  No phone service             DSL            Yes  …
4             Yes                No     Fiber optic             No  …
…           …                …             …            …  …
7038          Yes               Yes             DSL            Yes  …
7039          Yes               Yes     Fiber optic             No  …
7040           No  No phone service             DSL            Yes  …
7041          Yes               Yes     Fiber optic             No  …
7042          Yes                No     Fiber optic            Yes  …

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0                  No          No          No              No  Month-to-month
```

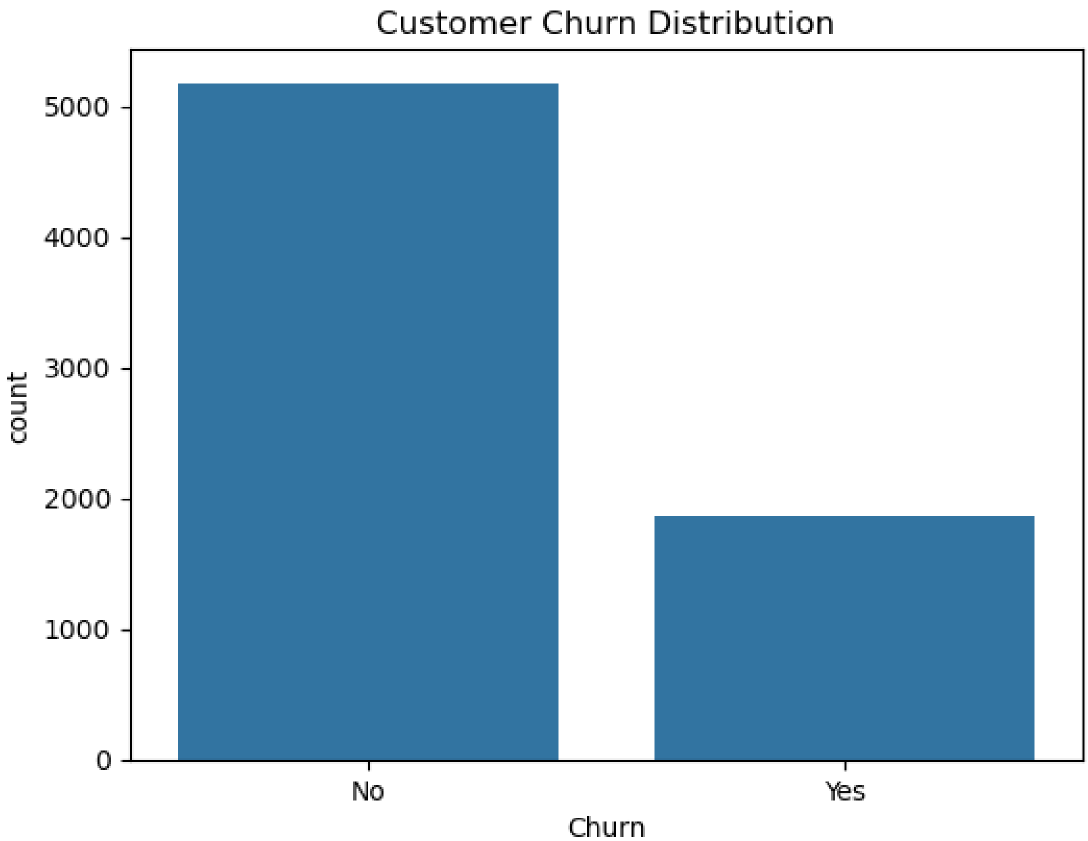
```
1                 Yes          No          No              No        One year
2                  No          No          No              No  Month-to-month
3                 Yes         Yes          No              No        One year
4                  No          No          No              No  Month-to-month
…                 …         …         …               …             …
7038              Yes         Yes         Yes             Yes        One year
7039              Yes          No         Yes             Yes        One year
7040               No          No          No              No  Month-to-month
7041               No          No          No              No  Month-to-month
7042              Yes         Yes         Yes             Yes        Two year

     PaperlessBilling              PaymentMethod MonthlyCharges  TotalCharges  \
0                 Yes           Electronic check          29.85         29.85
1                  No               Mailed check          56.95       1889.50
2                 Yes               Mailed check          53.85        108.15
3                  No  Bank transfer (automatic)          42.30       1840.75
4                 Yes           Electronic check          70.70        151.65
…                 …                        …            …           …
7038              Yes               Mailed check          84.80       1990.50
7039              Yes    Credit card (automatic)         103.20       7362.90
7040              Yes           Electronic check          29.60        346.45
7041              Yes               Mailed check          74.40        306.60
7042              Yes  Bank transfer (automatic)         105.65       6844.50

      Churn
0        No
1        No
2       Yes
3        No
4       Yes
…       …
7038     No
7039     No
7040     No
7041    Yes
7042     No

[7043 rows x 21 columns]>
```

EDA

[16]:
```python
telecom_data['TotalCharges'] = pd.to_numeric(telecom_data['TotalCharges'], errors='coerce')
```

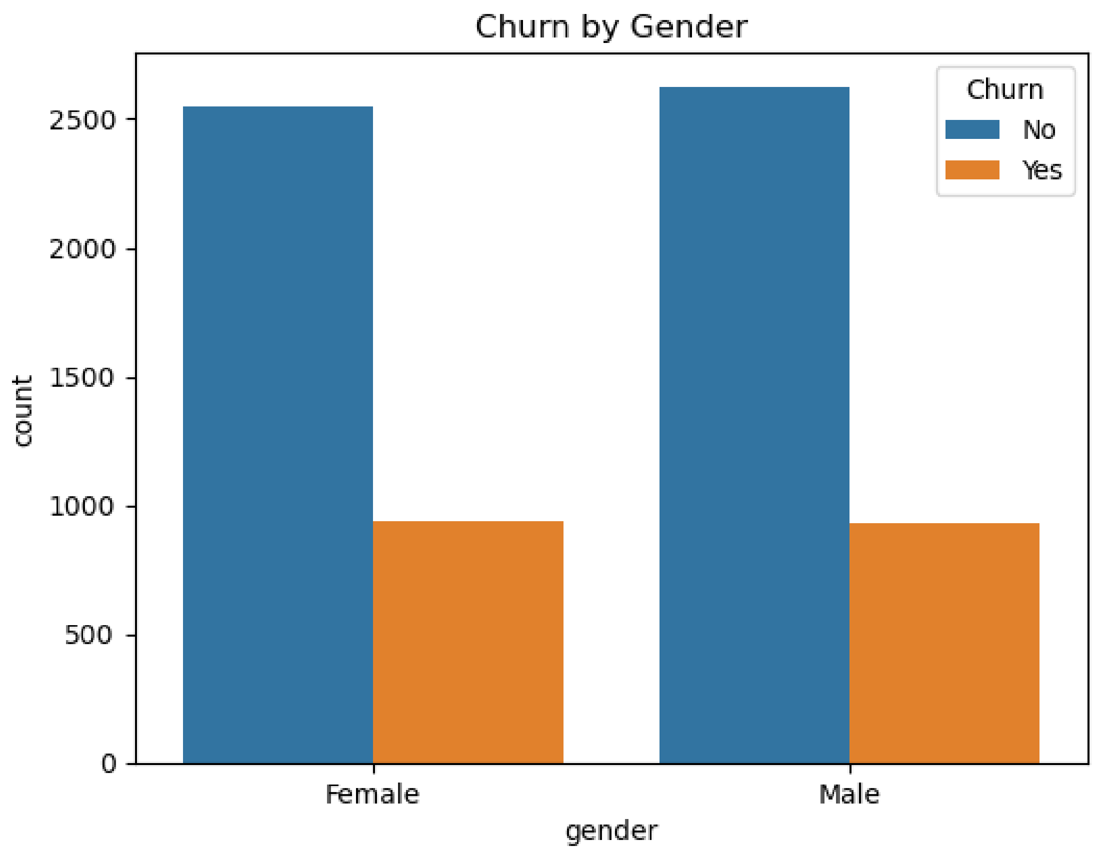
[17]:
```python
# Drop the customer ID column
telecom_data.drop('customerID', axis=1, inplace=True)
```

[18]:
```
# Visualize churn
sns.countplot(data=telecom_data, x='Churn')
plt.title("Customer Churn Distribution")
plt.show()
```



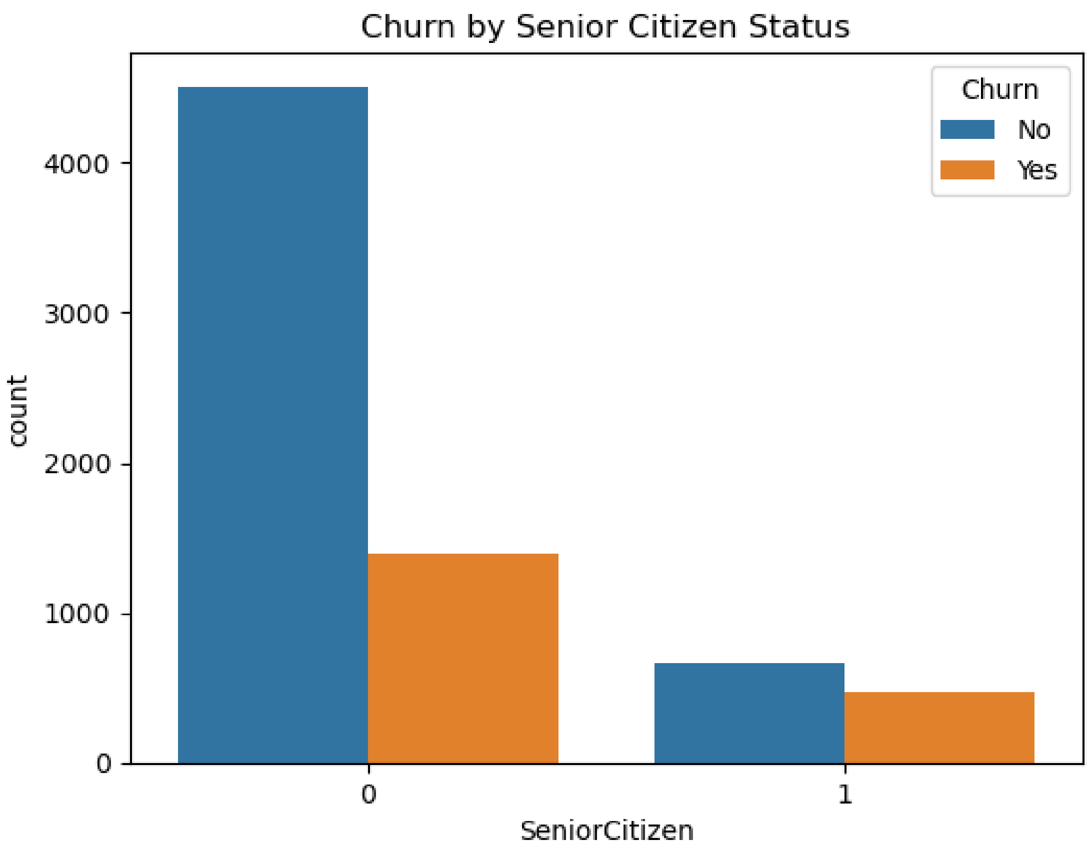
The dataset shows class imbalance, with more customers staying than leaving. This is important for modeling decisions later.

[19]:
```
# Churn by Gender
sns.countplot(data=telecom_data, x='gender', hue='Churn')
plt.title("Churn by Gender")
plt.show()
```



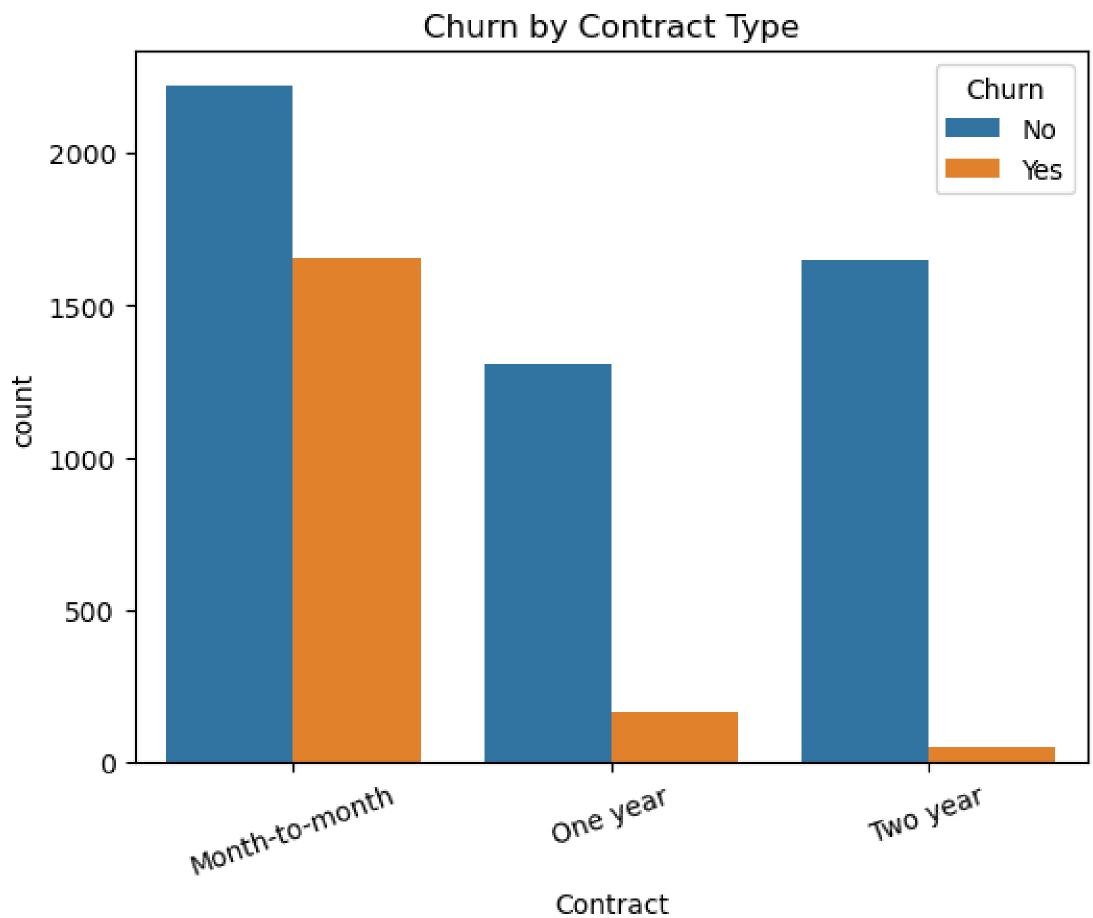
Churn rates appear similar across genders, suggesting gender alone is not a strong predictor.

[20]:
```
# Churn by Senior Citizen Status
sns.countplot(data=telecom_data, x='SeniorCitizen', hue='Churn')
plt.title("Churn by Senior Citizen Status")
plt.show()
```



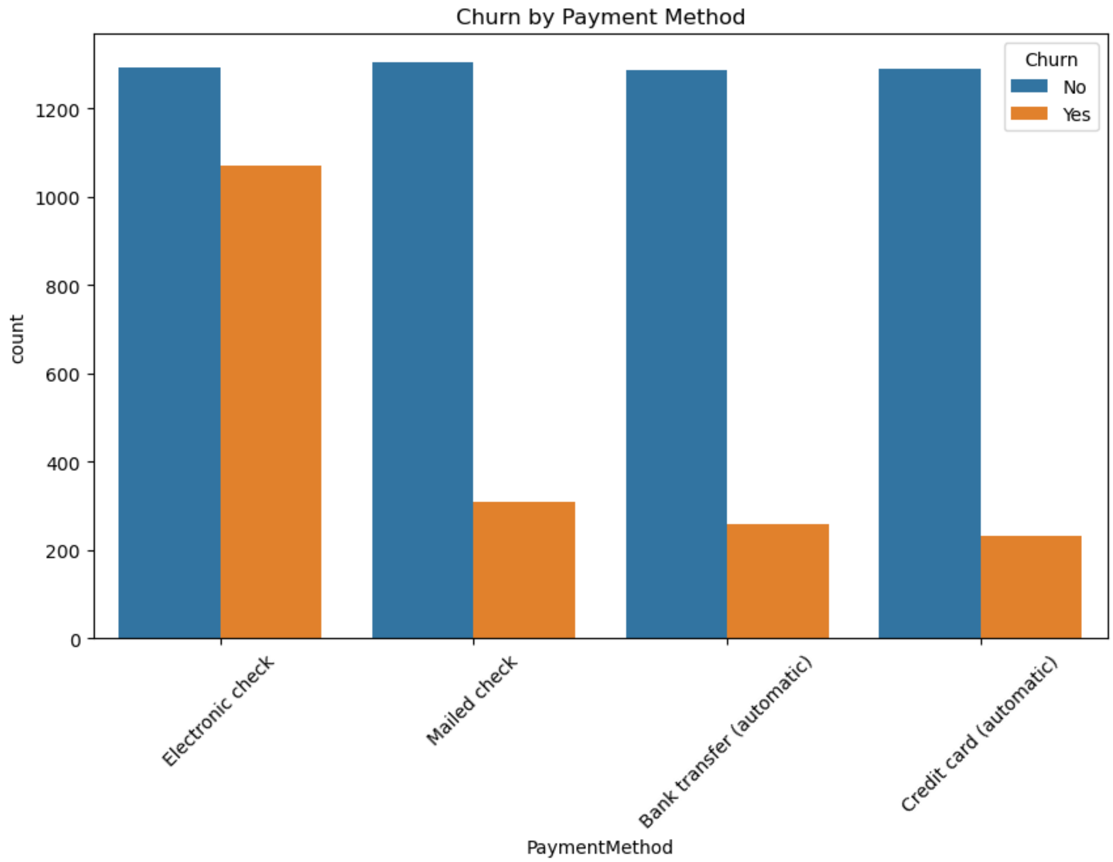
Senior citizens show a noticeably higher churn rate compared to non-senior customers.

[21]:
```
# Churn by contract type
sns.countplot(data=telecom_data, x='Contract', hue='Churn')
plt.xticks(rotation=20)
plt.title("Churn by Contract Type")
plt.show()
```



Month-to-month contracts have significantly higher churn compared to one-year and two-year contracts.

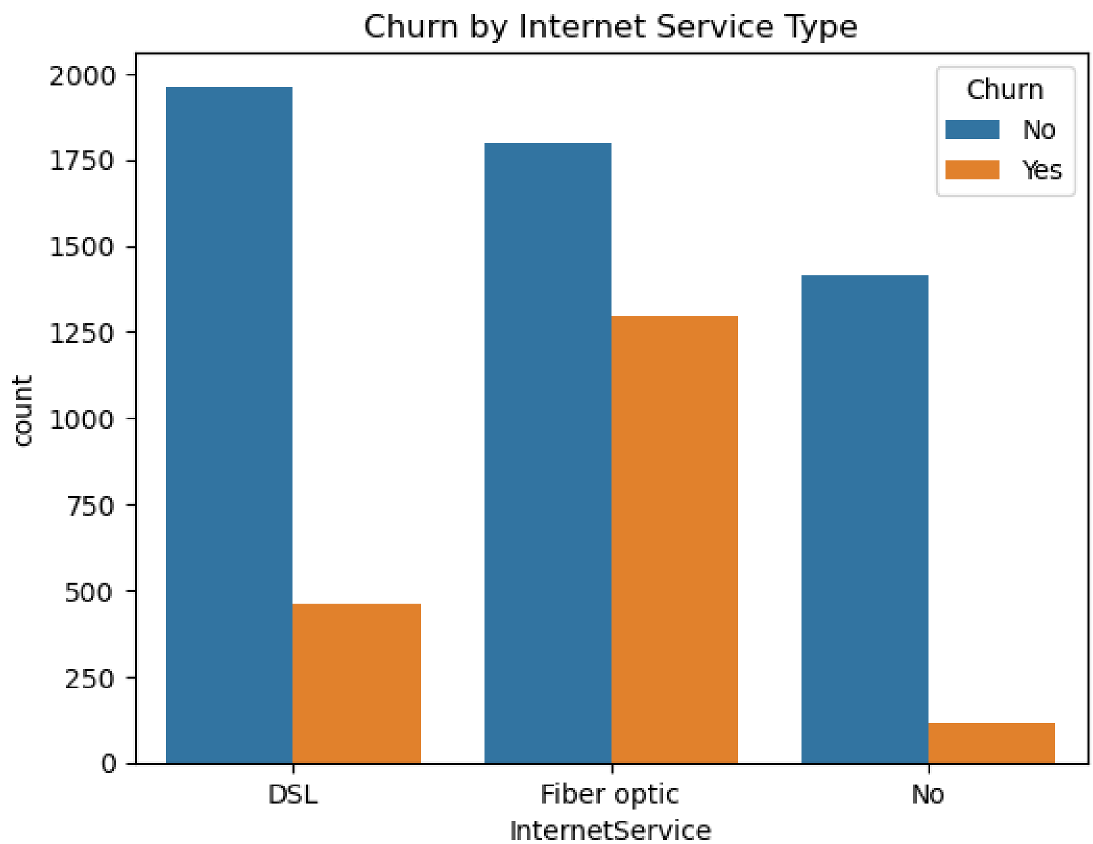
[22]:
```
# Churn by Payment Method
plt.figure(figsize=(10,6))
sns.countplot(data=telecom_data, x='PaymentMethod', hue='Churn')
plt.xticks(rotation=45)
plt.title("Churn by Payment Method")
plt.show()
```



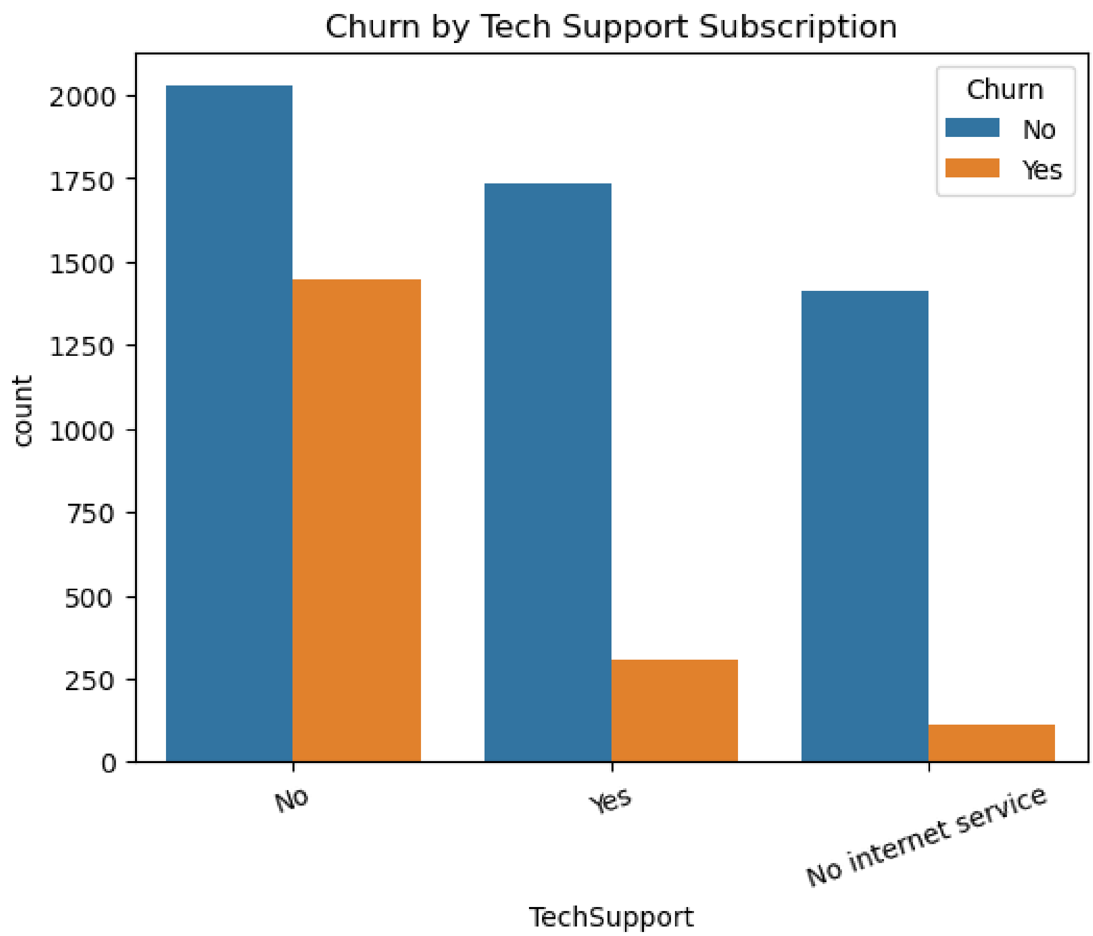
Customers using electronic checks churn more frequently than those using automatic payments.

[23]:

```
sns.countplot(data=telecom_data, x='InternetService', hue='Churn')
plt.title("Churn by Internet Service Type")
plt.show()
```

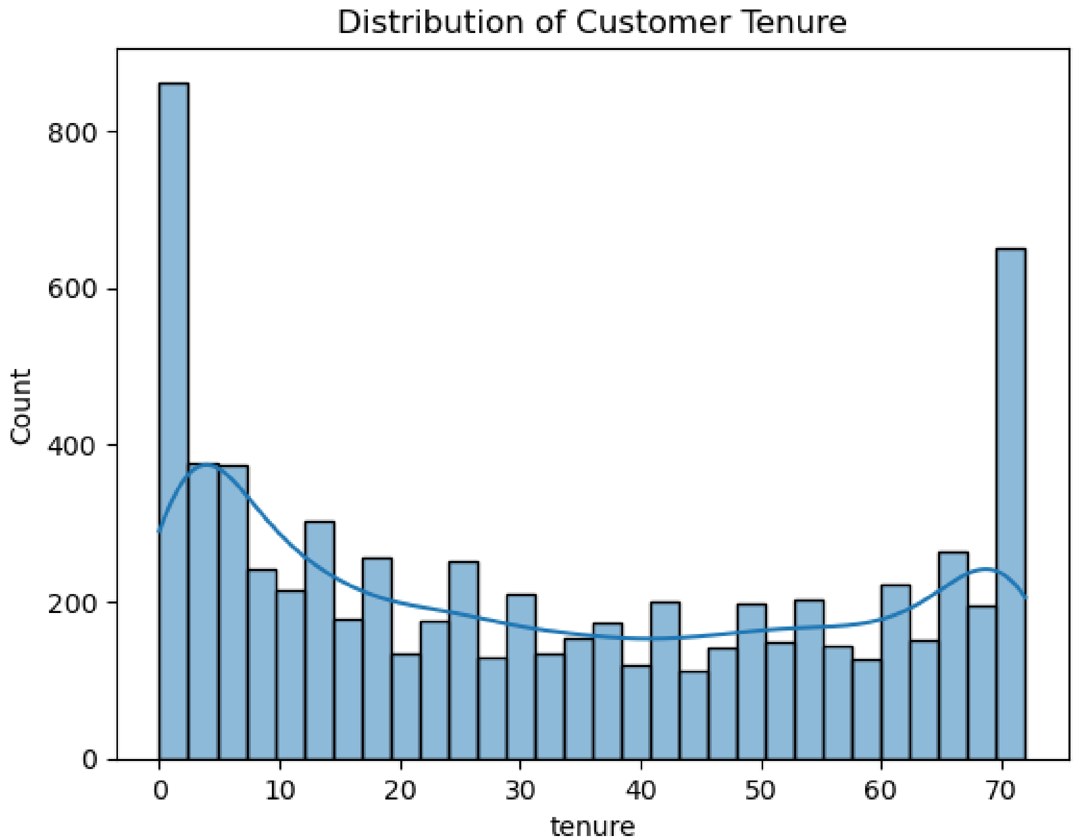


[24]:
```
sns.countplot(data=telecom_data, x='TechSupport', hue='Churn')
plt.xticks(rotation=20)
plt.title("Churn by Tech Support Subscription")
plt.show()
```

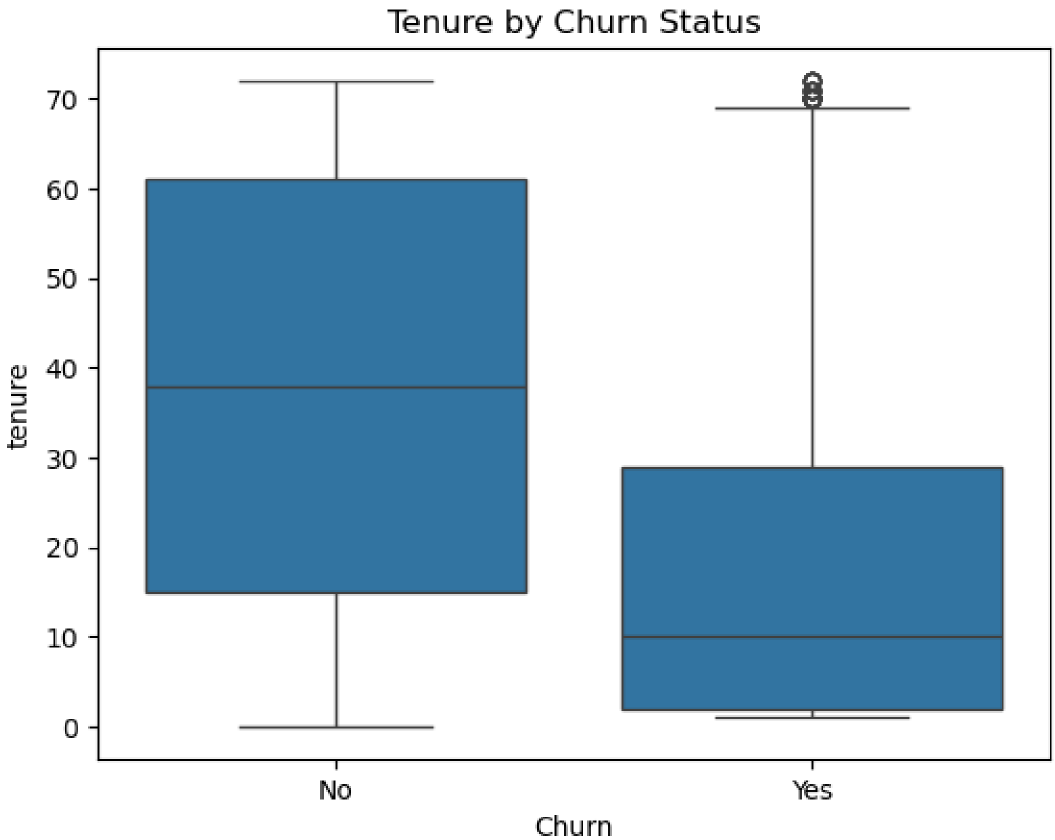


Customers without tech support churn at a much higher rate, suggesting support services improve retention.

[25]:
```
sns.histplot(telecom_data['tenure'], bins=30, kde=True)
plt.title("Distribution of Customer Tenure")
plt.show()
```

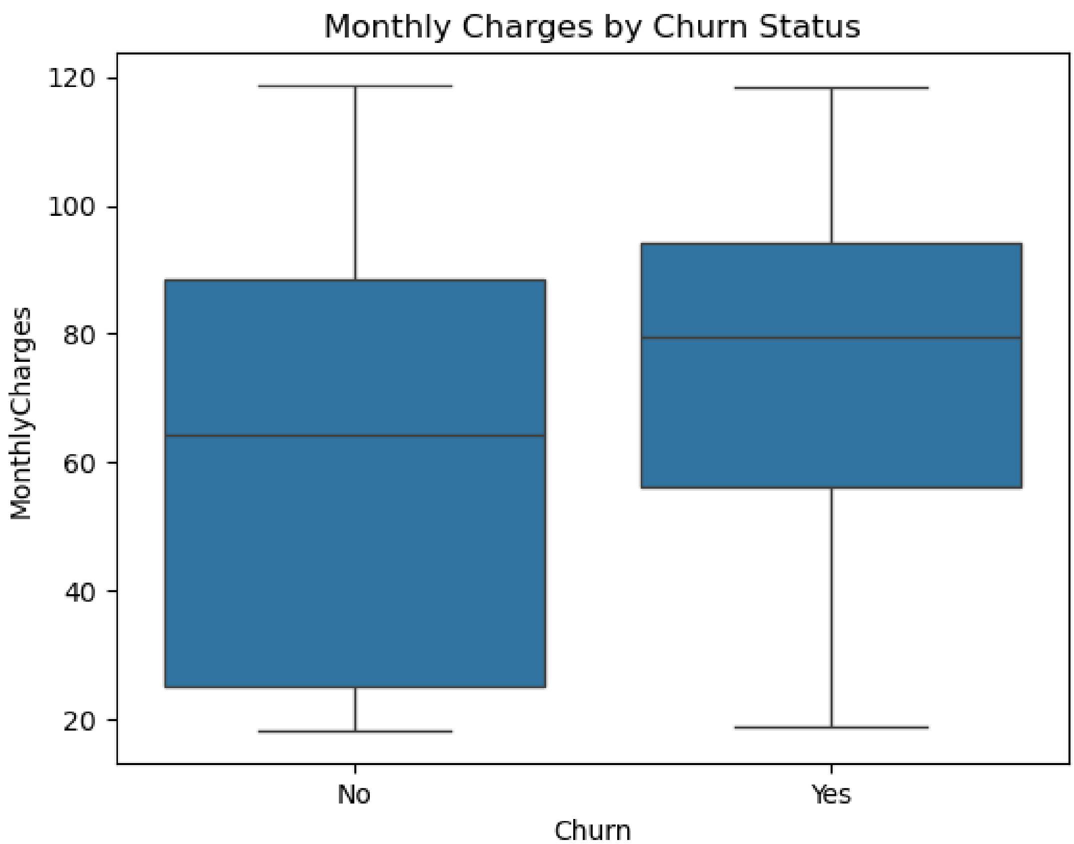


[26]:
```
sns.boxplot(data=telecom_data, x='Churn', y='tenure')
plt.title("Tenure by Churn Status")
plt.show()
```



Customers who churn tend to have much shorter tenure.

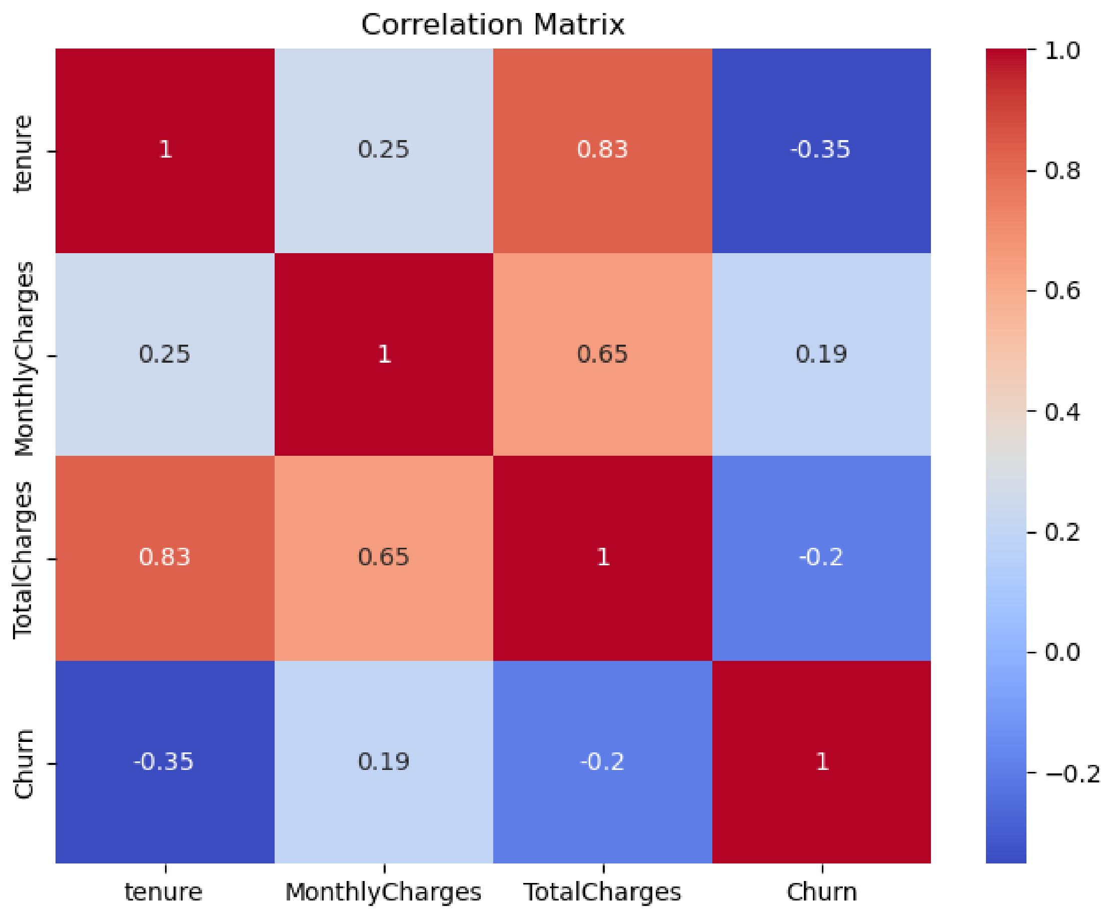
[27]:
```
# Monthly Charges vs Churn
sns.boxplot(data=telecom_data, x='Churn', y='MonthlyCharges')
plt.title("Monthly Charges by Churn Status")
plt.show()
```



Customers with higher monthly charges are more likely to churn.

[28]:
```
# Chrun Correlation analysis
tc_corr = telecom_data.copy()
tc_corr['Churn'] = tc_corr['Churn'].map({'Yes': 1, 'No': 0})
```

[29]:
```
plt.figure(figsize=(8,6))
sns.heatmap(tc_corr[['tenure', 'MonthlyCharges', 'TotalCharges', 'Churn']].
↪corr(),
            annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```



Key Findings: Tenure is strongly negatively correlated with churn. Monthly charges show a positive correlation with churn

Key Takeaways * Customers on month-to-month contracts are the most likely to churn * Short-tenure customers are significantly more likely to leave * Higher monthly charges are associated with higher churn * Customers lacking support services (e.g., tech support) churn more often * The dataset exhibits class imbalance, which must be addressed during modeling

Telco Customer Churn Prediction

[30]:
```
# Strip whitespace from categorical columns
for col in telecom_data.select_dtypes(include='object').columns:
    telecom_data[col] = telecom_data[col].str.strip()
```

[31]:
```
# Separate Features and Target
X = telecom_data.drop('Churn', axis=1)
y = telecom_data['Churn'].map({'Yes': 1, 'No': 0})
```

[32]:
```
assert y.isna().sum() == 0, "Target variable contains NaNs"
assert X.isna().sum().sum() == 0, "Feature matrix contains NaNs"
```

[33]:
```python
# Train, Test, Split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

[34]:
```python
# Feature type identification
categorical_features = X.select_dtypes(include='object').columns.tolist()
numerical_features = X.select_dtypes(include=['int64', 'float64']).columns.
 ↪tolist()
```

[35]:
```python
# Column transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'),␣
 ↪categorical_features)
    ]
)
```

[36]:
```python
log_reg_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(
        max_iter=1000,
        class_weight='balanced',
        random_state=42
    ))
])
```

[37]:
```python
# Train and evaluate
log_reg_pipeline.fit(X_train, y_train)

y_pred = log_reg_pipeline.predict(X_test)
y_prob = log_reg_pipeline.predict_proba(X_test)[:, 1]
```

[38]:
```python
from sklearn.metrics import classification_report, roc_auc_score

print(classification_report(y_test, y_pred))
print("ROC-AUC:", roc_auc_score(y_test, y_prob))
```

```
              precision    recall  f1-score   support

           0       0.90      0.72      0.80      1035
           1       0.51      0.78      0.61       374
```

```
    accuracy                           0.74      1409
   macro avg       0.70      0.75      0.71      1409
weighted avg       0.80      0.74      0.75      1409
```

```
ROC-AUC: 0.8417499806246609
```

Overall Performance * Accuracy: 74% * ROC-AUC: 0.84 → Strong discriminative power

Even though accuracy is moderate, the ROC-AUC indicates the model is very good at ranking customers by churn risk.

[40]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = roc_auc_score(y_test, y_prob)
```

[ ]:
```
# Plot the ROC Curve
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Customer Churn Prediction')
plt.legend(loc='lower right')
plt.tight_layout()
plt.show()
```