mh2rjwjf5

February 20, 2025

[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, f1_score, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv(r"C:
↪\Users\Aadiluddin\Downloads\WA_Fn-UseC_-Telco-Customer-Churn.csv")
```

[3]:
```
df.head()
```

[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	Female	0	Yes	No	1	No	…	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	5575-GNVDE	Male	0	No	No	34	Yes	…	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
2	3668-QPYBK	Male	0	No	No	2	Yes	…	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	7795-CFOCW	Male	0	No	No	45	No	…	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	9237-HQITU	Female	0	No	No	2	Yes	…	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

```
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]
```

[4]:
```
df.shape
```

[4]:
```
(7043, 21)
```

[5]:
```
df.isnull().sum()
```

[5]:
```
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
```

[6]:
```
df.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
```

[7]:
```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors="coerce")
df["TotalCharges"].fillna(0, inplace=True)
```

[8]:
```
df.isnull().sum()
```

[8]:
```
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
```

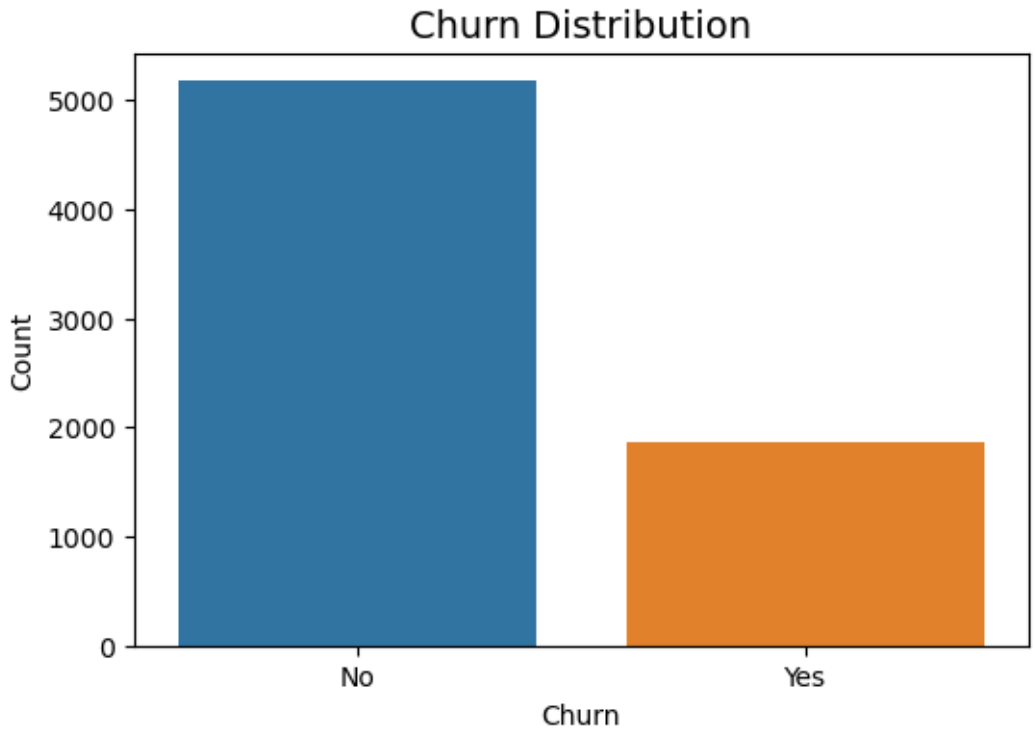
```
Churn             0
dtype: int64
```

[9]:
```
df.columns
```

[9]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

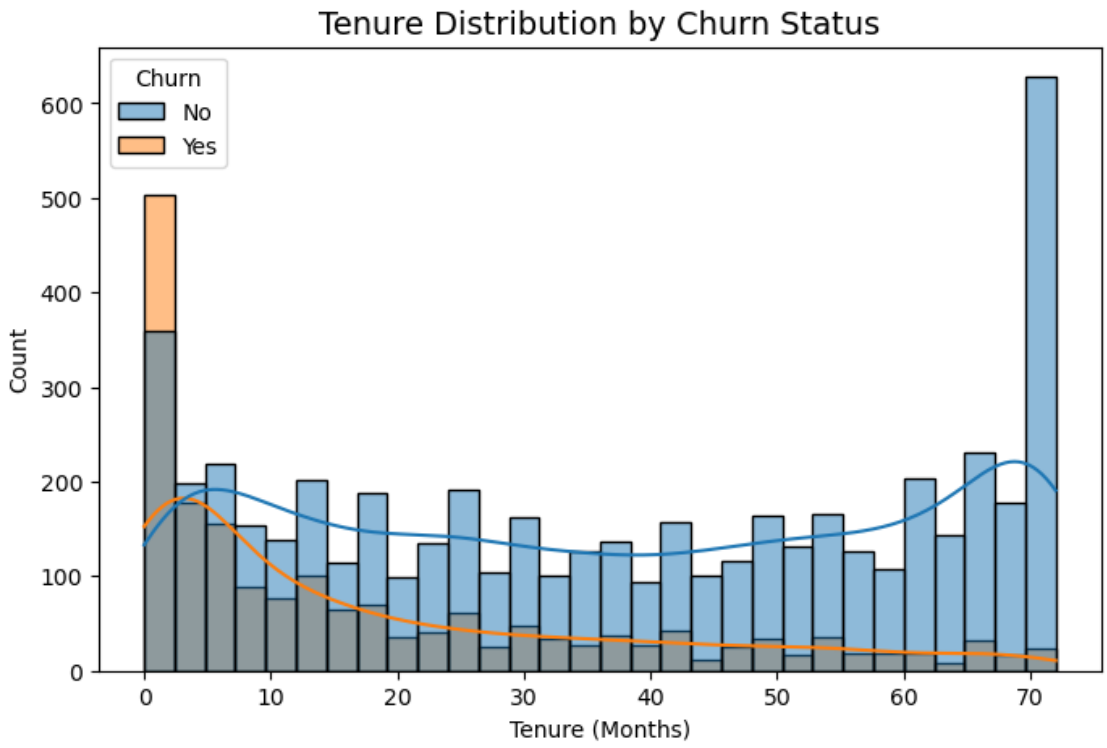
[10]:
```
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors="coerce")
df["TotalCharges"].fillna(0, inplace=True)
```

[11]:
```
plt.figure(figsize=(6, 4))
sns.countplot(x="Churn", data=df, palette=["#1f77b4", "#ff7f0e"])
plt.title("Churn Distribution", fontsize=14)
plt.xlabel("Churn")
plt.ylabel("Count")
plt.show()
```

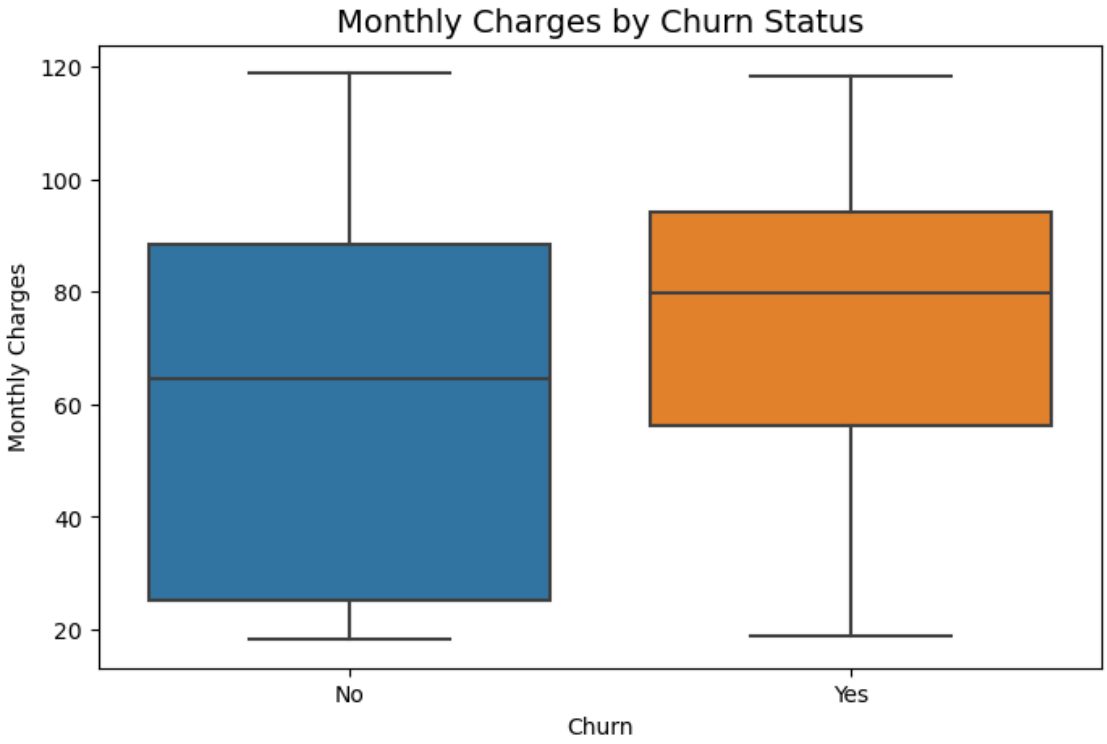


[12]:
```
plt.figure(figsize=(8, 5))
sns.histplot(df, x="tenure", hue="Churn", bins=30, kde=True,␣
 ↪palette=["#1f77b4", "#ff7f0e"])
plt.title("Tenure Distribution by Churn Status", fontsize=14)
plt.xlabel("Tenure (Months)")
plt.ylabel("Count")
plt.show()
```

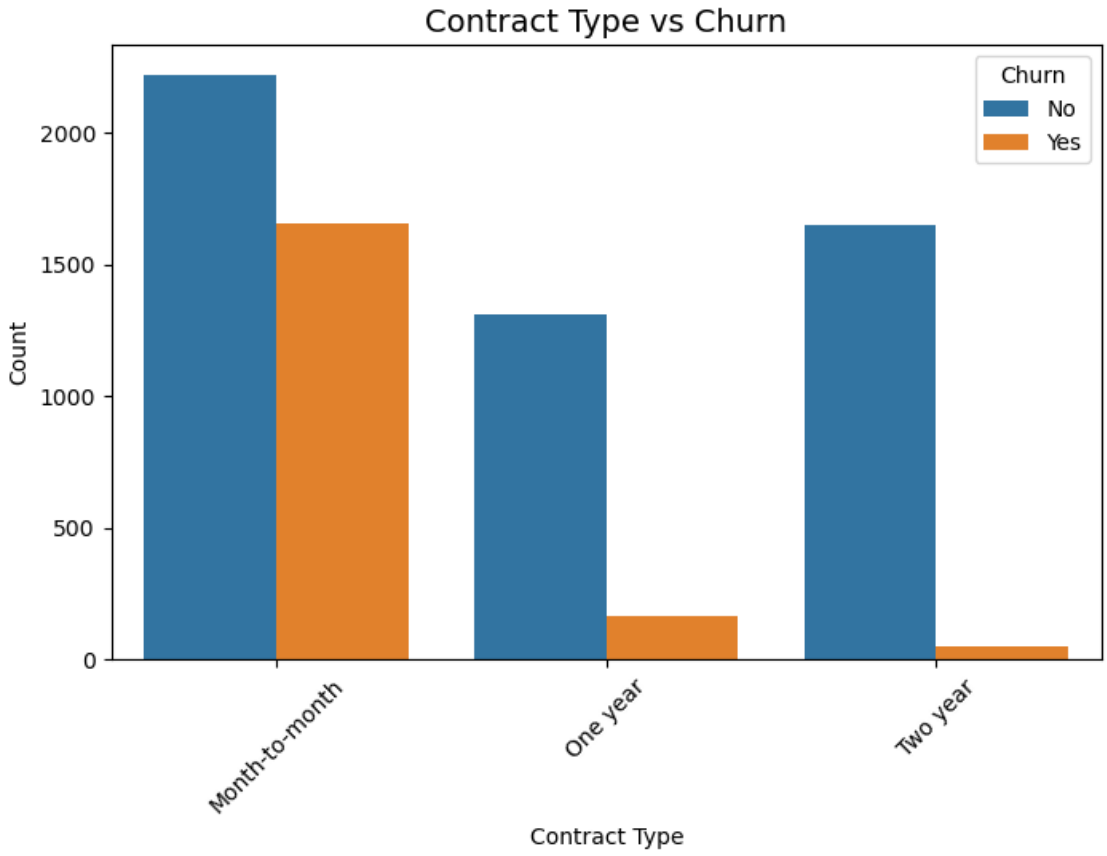
```
C:\Users\Aadiluddin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



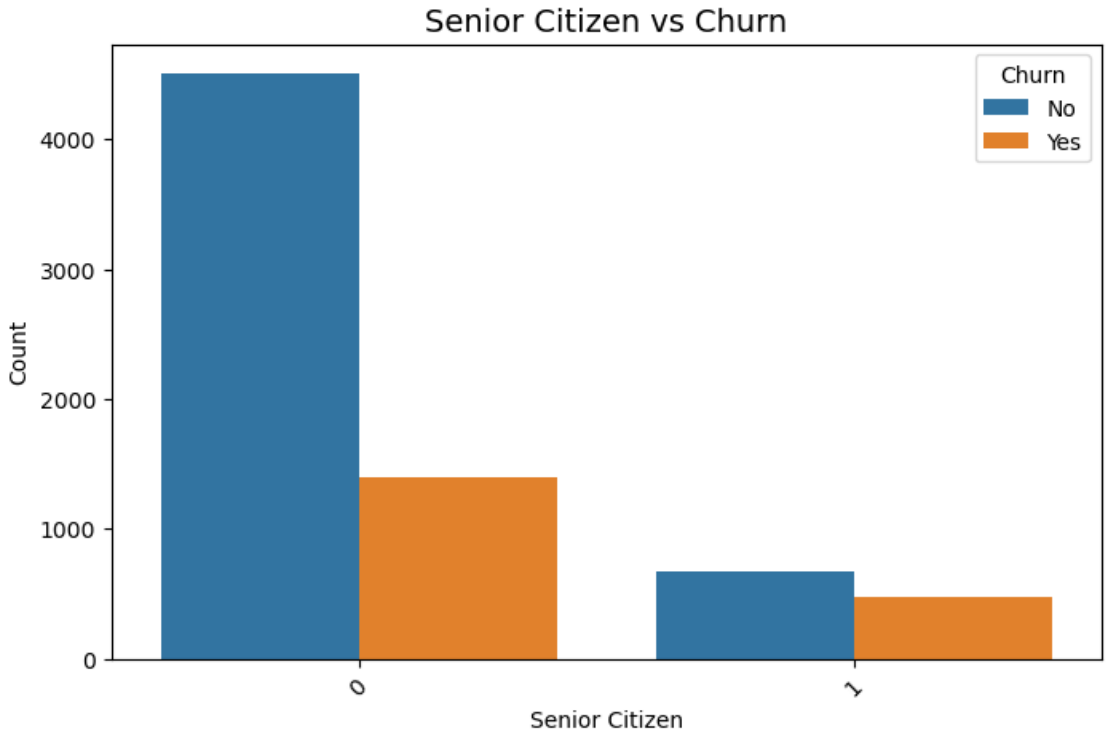
[13]:
```
plt.figure(figsize=(8, 5))
sns.boxplot(x="Churn", y="MonthlyCharges", data=df, palette=["#1f77b4",␣
 ↪"#ff7f0e"])
plt.title("Monthly Charges by Churn Status", fontsize=14)
plt.xlabel("Churn")
plt.ylabel("Monthly Charges")
plt.show()
```



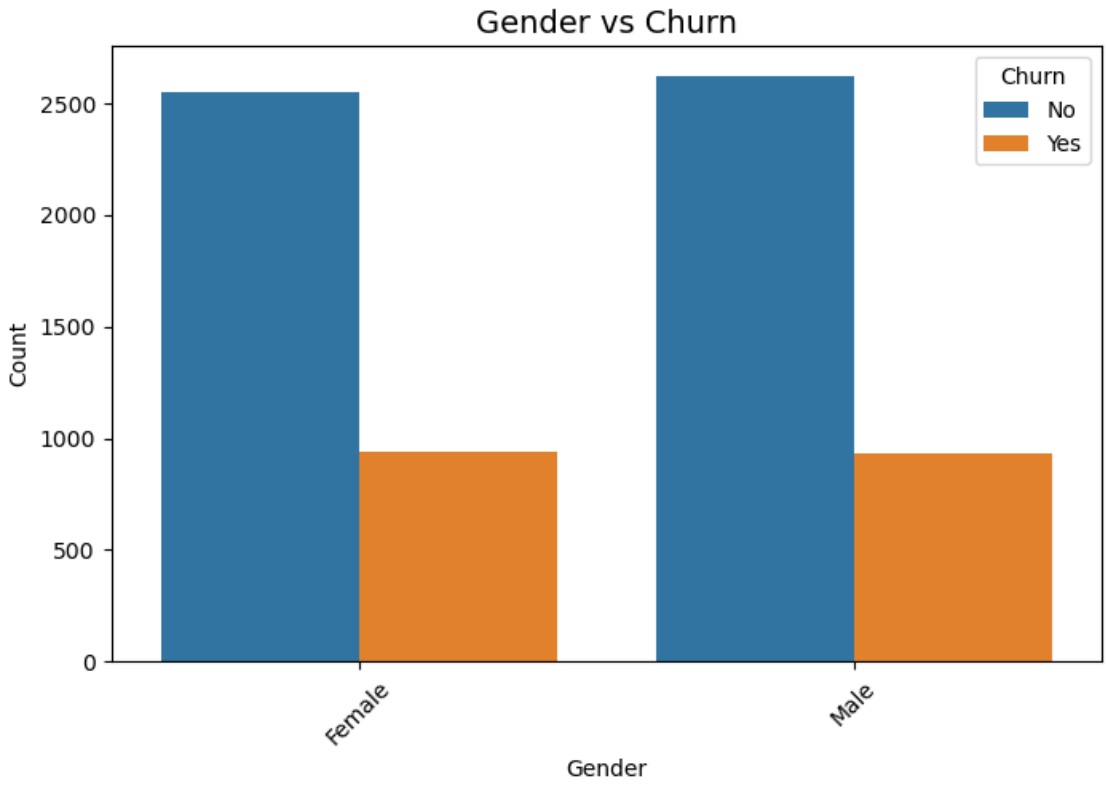
[72]:
```
plt.figure(figsize=(8, 5))
sns.countplot(x="Contract", hue="Churn", data=df, palette=["#1f77b4",␣
↪"#ff7f0e"])
plt.title("Contract Type vs Churn", fontsize=14)
plt.xlabel("Contract Type")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



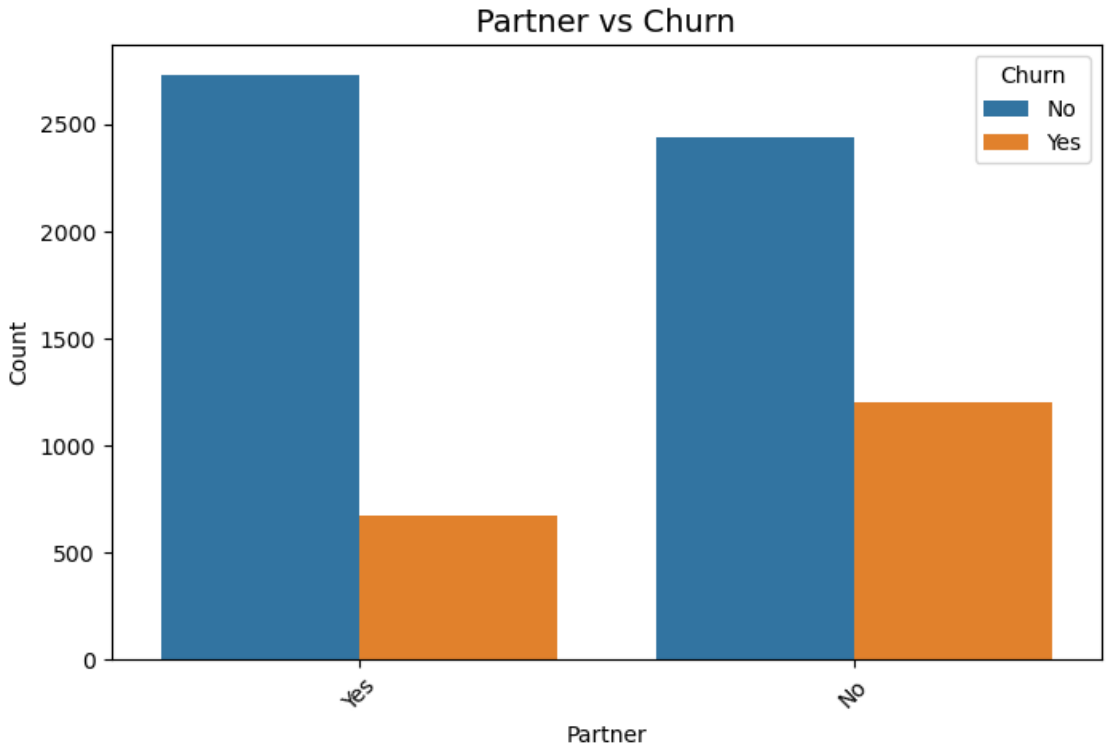
[73]:
```
plt.figure(figsize=(8, 5))
sns.countplot(x="SeniorCitizen", hue="Churn", data=df)
plt.title("Senior Citizen vs Churn", fontsize=14)
plt.xlabel("Senior Citizen")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



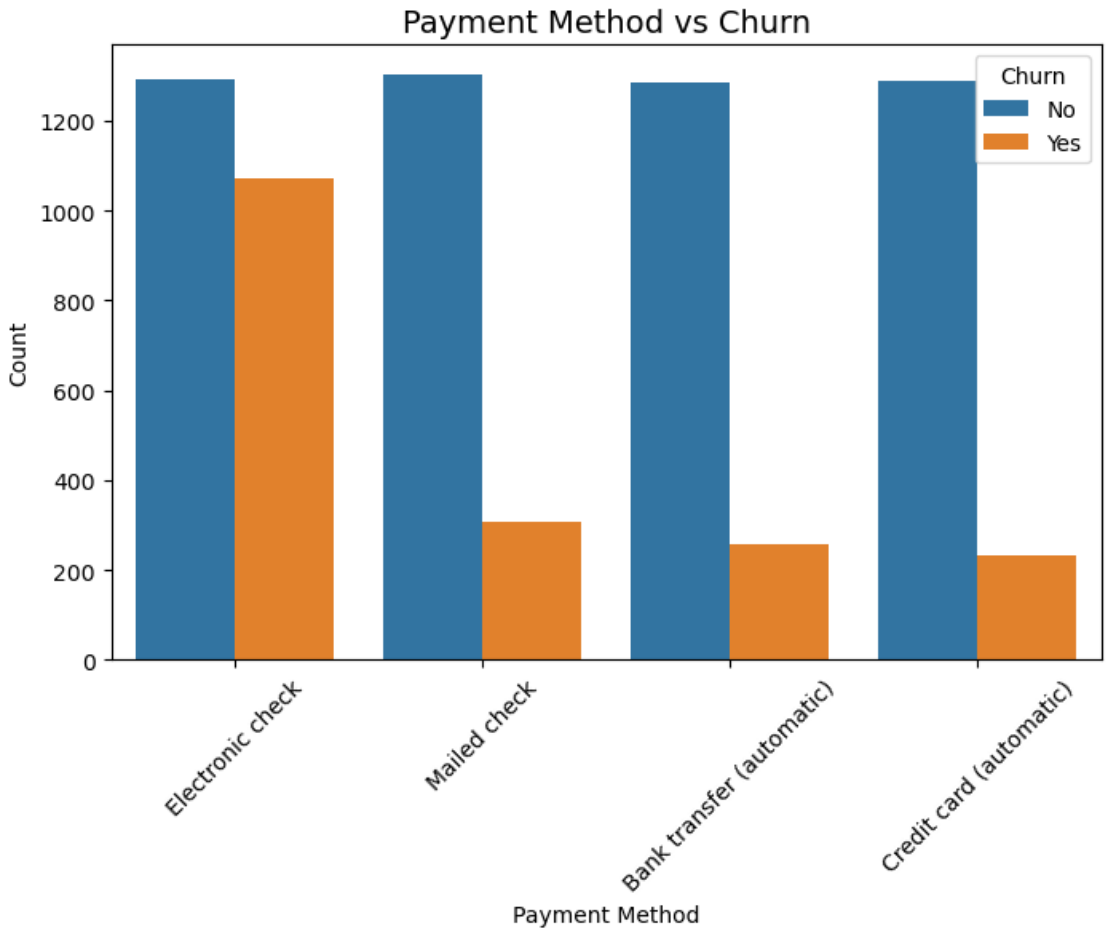
[74]:
```
plt.figure(figsize=(8, 5))
sns.countplot(x="gender", hue="Churn", data=df)
plt.title("Gender vs Churn", fontsize=14)
plt.xlabel("Gender")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



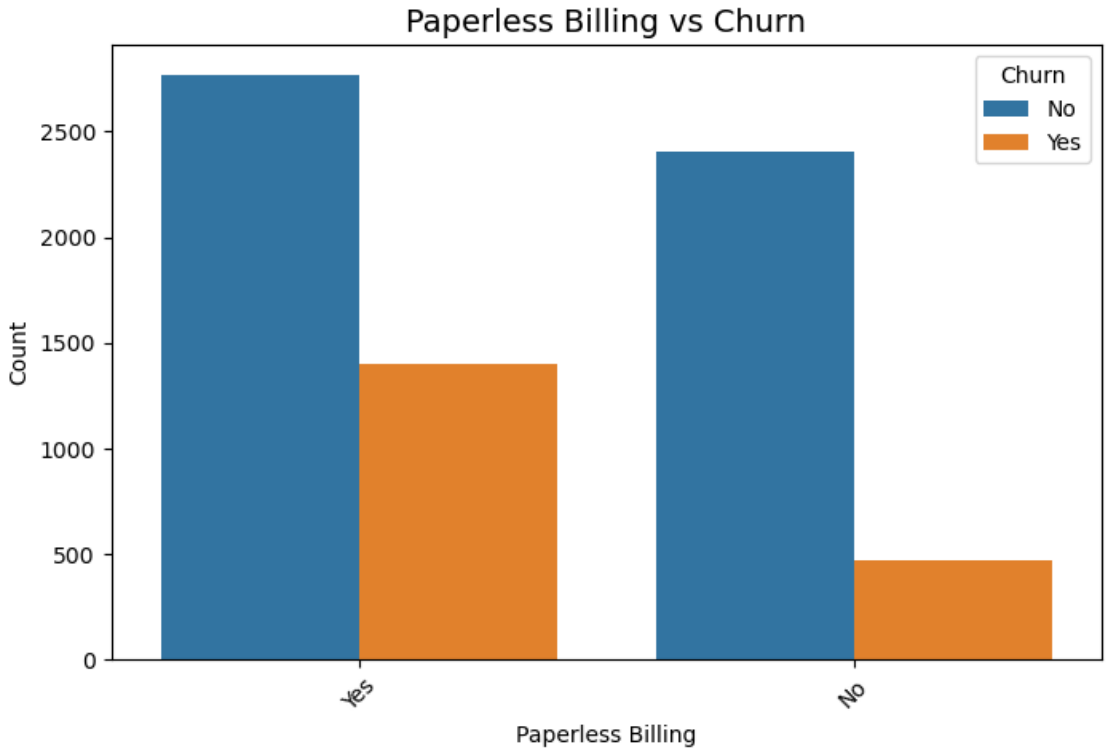
[75]:
```
plt.figure(figsize=(8, 5))
sns.countplot(x="Partner", hue="Churn", data=df)
plt.title("Partner vs Churn", fontsize=14)
plt.xlabel("Partner")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



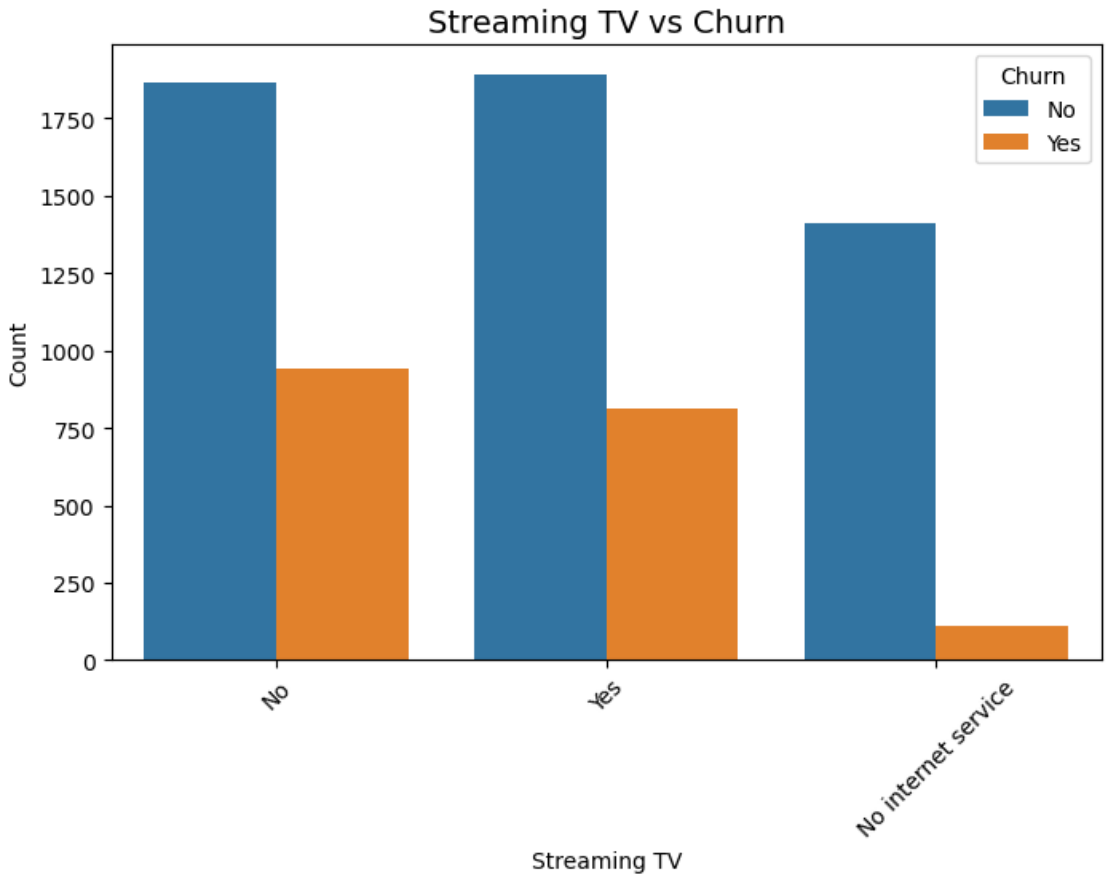
[76]:
```
plt.figure(figsize=(8, 5))
sns.countplot(x="PaymentMethod", hue="Churn", data=df)
plt.title("Payment Method vs Churn", fontsize=14)
plt.xlabel("Payment Method")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



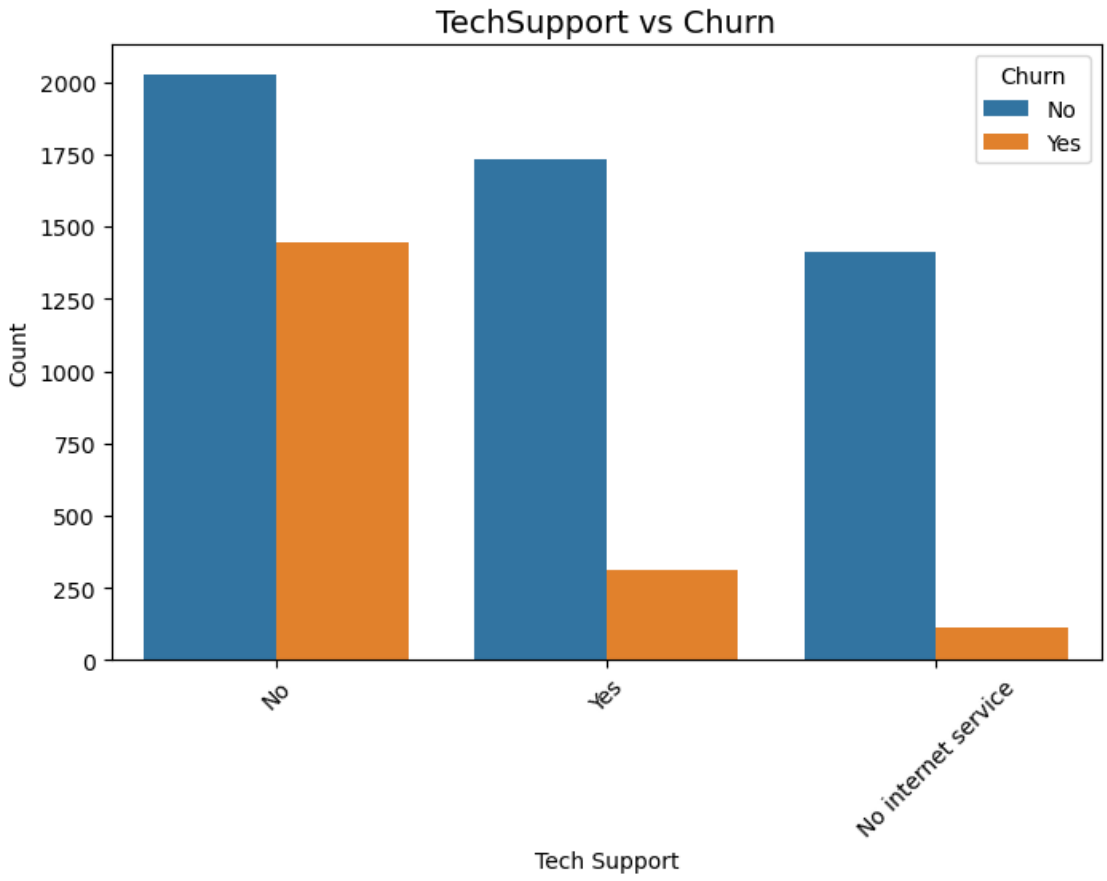
[77]:
```
plt.figure(figsize=(8, 5))
sns.countplot(x="PaperlessBilling", hue="Churn", data=df)
plt.title("Paperless Billing vs Churn", fontsize=14)
plt.xlabel("Paperless Billing")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



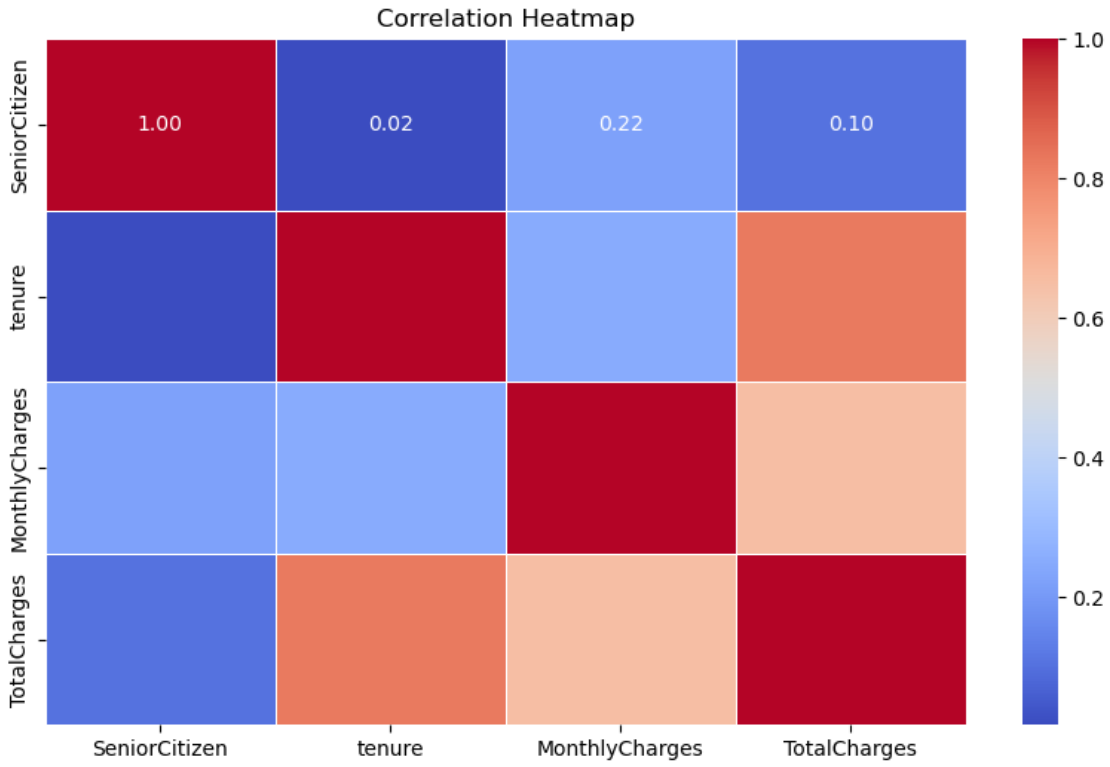
[78]:
```
plt.figure(figsize=(8, 5))
sns.countplot(x="StreamingTV", hue="Churn", data=df)
plt.title("Streaming TV vs Churn", fontsize=14)
plt.xlabel("Streaming TV")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



[79]:
```
plt.figure(figsize=(8, 5))
sns.countplot(x="TechSupport", hue="Churn", data=df)
plt.title("TechSupport vs Churn", fontsize=14)
plt.xlabel("Tech Support")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



[80]:
```
numeric_df = df.select_dtypes(include=["number"])

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(numeric_df.corr(), annot=True, fmt=".2f", cmap="coolwarm",␣
 ↪linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```



[81]:
```
df.columns
```

[81]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

[82]:
```
from sklearn.preprocessing import LabelEncoder

# List of categorical columns to encode
categorical_cols = ["gender", "Partner", "Dependents", "PhoneService",␣
 ↪"MultipleLines",
                    "InternetService", "OnlineSecurity", "OnlineBackup",␣
 ↪"DeviceProtection",
                    "TechSupport", "StreamingTV", "StreamingMovies", "Contract",
                    "PaperlessBilling", "PaymentMethod", "Churn"]

# Apply Label Encoding

for col in categorical_cols:
```

```
    encoder = LabelEncoder()
    df[col] = encoder.fit_transform(df[col])
```

[83]:
```
df["Churn"] = df["Churn"].replace({"Yes": 1, "No": 0})
```

[84]:
```
df["HasInternet"] = df["InternetService"].apply(lambda x: 0 if x == "No" else 1)
```

[85]:
```
service_cols = ["OnlineSecurity", "OnlineBackup", "DeviceProtection",
                "TechSupport", "StreamingTV", "StreamingMovies"]

df["TotalServices"] = df[service_cols].apply(lambda x: sum(x == "Yes"), axis=1)
```

[86]:
```
df["TenureGroup"] = pd.cut(df["tenure"], bins=[0, 12, 24, 48, 72],
                           labels=["0-1 Year", "1-2 Years", "2-4 Years", "4-6␣
 ↪Years"])
```

[87]:
```
df["TenureGroup"].fillna(df["TenureGroup"].mode()[0], inplace=True)
```

[88]:
```
df.columns
```

[88]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn',
       'HasInternet', 'TotalServices', 'TenureGroup'],
      dtype='object')
```

[89]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df["TenureGroup"] = le.fit_transform(df["TenureGroup"])  # Assuming␣
 ↪"tenure_group" is the column name
```

[90]:
```
print(df.head())
print(df.info())
```

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

   PhoneService  MultipleLines  InternetService  OnlineSecurity  …  \
0             0              1                0               0  …
1             1              0                0               2  …
```

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

   StreamingMovies  Contract  PaperlessBilling  PaymentMethod  MonthlyCharges  \
0                0         0                 1              2           29.85
1                0         1                 0              3           56.95
2                0         0                 1              3           53.85
3                0         1                 0              0           42.30
4                0         0                 1              2           70.70

   TotalCharges  Churn  HasInternet  TotalServices  TenureGroup
0         29.85      0            1              0            0
1       1889.50      0            1              0            2
2        108.15      1            1              0            0
3       1840.75      0            1              0            2
4        151.65      1            1              0            0

[5 rows x 24 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   int32
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   int32
 4   Dependents        7043 non-null   int32
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   int32
 7   MultipleLines     7043 non-null   int32
 8   InternetService   7043 non-null   int32
 9   OnlineSecurity    7043 non-null   int32
 10  OnlineBackup      7043 non-null   int32
 11  DeviceProtection  7043 non-null   int32
 12  TechSupport       7043 non-null   int32
 13  StreamingTV       7043 non-null   int32
 14  StreamingMovies   7043 non-null   int32
 15  Contract          7043 non-null   int32
 16  PaperlessBilling  7043 non-null   int32
 17  PaymentMethod     7043 non-null   int32
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   float64
 20  Churn             7043 non-null   int32
 21  HasInternet       7043 non-null   int64
 22  TotalServices     7043 non-null   int64
 23  TenureGroup       7043 non-null   int32
```

```
dtypes: float64(2), int32(17), int64(4), object(1)
memory usage: 853.0+ KB
None
```

[91]:
```
print(df.isnull().sum())
```

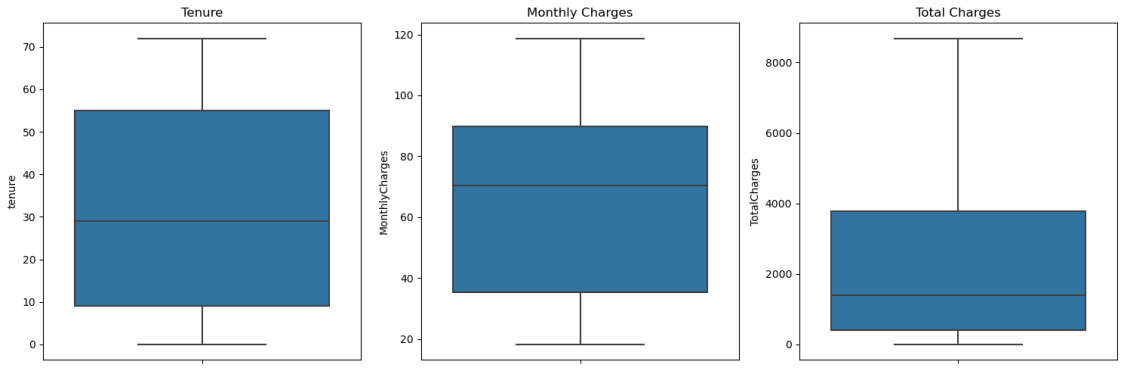
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV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

[92]:
```
df["Churn"].value_counts(normalize=True) * 100
```

[92]:
```
Churn
0    73.463013
1    26.536987
Name: proportion, dtype: float64
```

[93]:
```
import seaborn as sns
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 3, figsize=(15, 5))

sns.boxplot(y=df["tenure"], ax=axes[0])
axes[0].set_title("Tenure")
```

```
sns.boxplot(y=df["MonthlyCharges"], ax=axes[1])
axes[1].set_title("Monthly Charges")

sns.boxplot(y=df["TotalCharges"], ax=axes[2])
axes[2].set_title("Total Charges")

plt.tight_layout()
plt.show()
```



[95]:
```
import pandas as pd
import numpy as np

# Select numerical columns
num_cols = ["tenure", "MonthlyCharges", "TotalCharges"]

# Compute IQR
Q1 = df[num_cols].quantile(0.25)
Q3 = df[num_cols].quantile(0.75)
IQR = Q3 - Q1

# Define outlier thresholds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Count outliers per column
outliers_count = ((df[num_cols] < lower_bound) | (df[num_cols] > upper_bound)).
  ↪sum()
print("Outliers per column:\n", outliers_count)
```

```
Outliers per column:
 tenure            0
MonthlyCharges    0
TotalCharges      0
dtype: int64
```

[97]:
```
skewness = df[num_cols].skew()
print("Skewness per column:\n", skewness)
```

```
Skewness per column:
 tenure            0.239540
MonthlyCharges    -0.220524
TotalCharges       0.963235
dtype: float64
```

[36]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df[["MonthlyCharges", "TotalCharges"]] = scaler.
 ↪fit_transform(df[["MonthlyCharges", "TotalCharges"]])
```

[37]:
```
df = df.drop(columns=['customerID'])
```

[55]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪roc_auc_score

X = df.drop(columns=['Churn'])
y = df['Churn']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)


model = LogisticRegression(class_weight='balanced',max_iter=500)
model.fit(X_train, y_train)


y_pred = model.predict(X_test)


print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, model.predict_proba(X_test)[:,␣
 ↪1]))
```

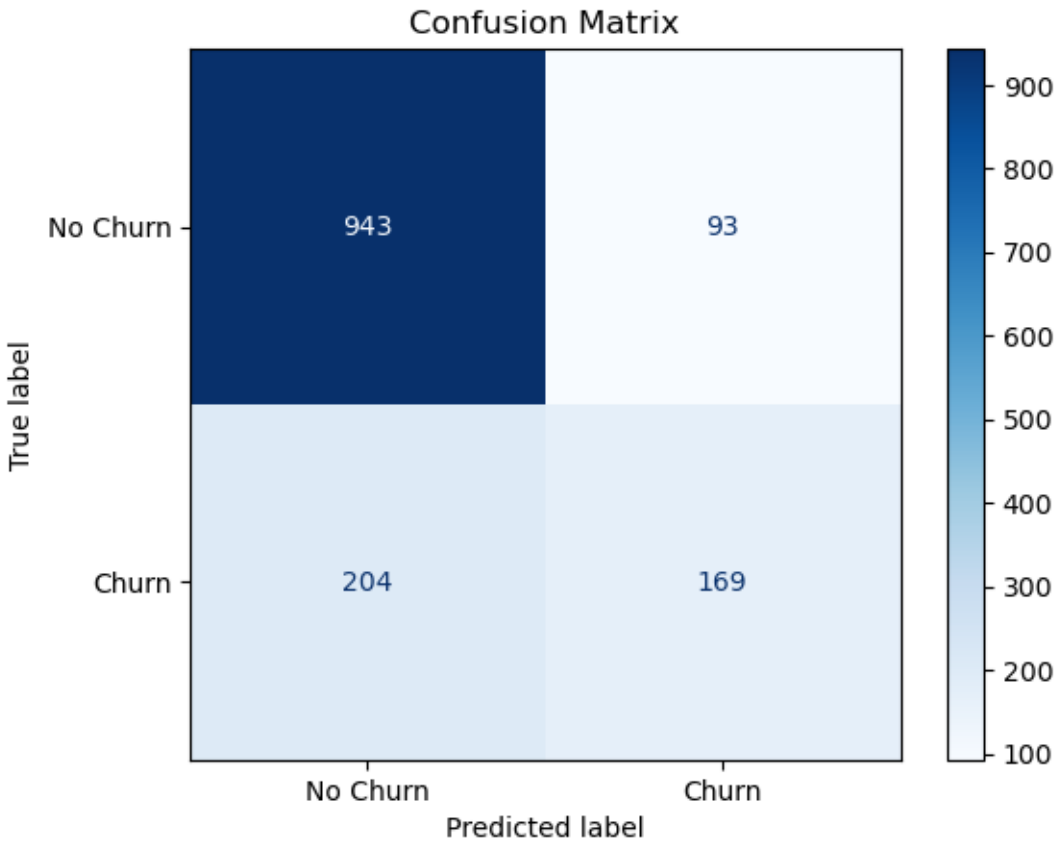
```
              precision    recall  f1-score   support

           0       0.93      0.73      0.82      1036
           1       0.53      0.84      0.65       373
```

```
    accuracy                           0.76      1409
   macro avg       0.73      0.78      0.73      1409
weighted avg       0.82      0.76      0.77      1409

[[757 279]
 [ 61 312]]
ROC-AUC Score: 0.8610776134234579
```

[56]:
```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(class_weight='balanced')
model.fit(X_train, y_train)
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪roc_auc_score


y_pred = model.predict(X_test)


print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, model.predict_proba(X_test)[:,␣
 ↪1]))
```

```
              precision    recall  f1-score   support

           0       0.82      0.92      0.87      1036
           1       0.67      0.45      0.54       373

    accuracy                           0.80      1409
   macro avg       0.75      0.69      0.71      1409
weighted avg       0.78      0.80      0.78      1409

[[954  82]
 [204 169]]
ROC-AUC Score: 0.833605484074653
```

[52]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['No Churn',␣
 ↪'Churn'])
disp.plot(cmap='Blues')
plt.title('Confusion Matrix')
plt.show()
```



[46]:
```
model.fit(X_train, y_train)
```

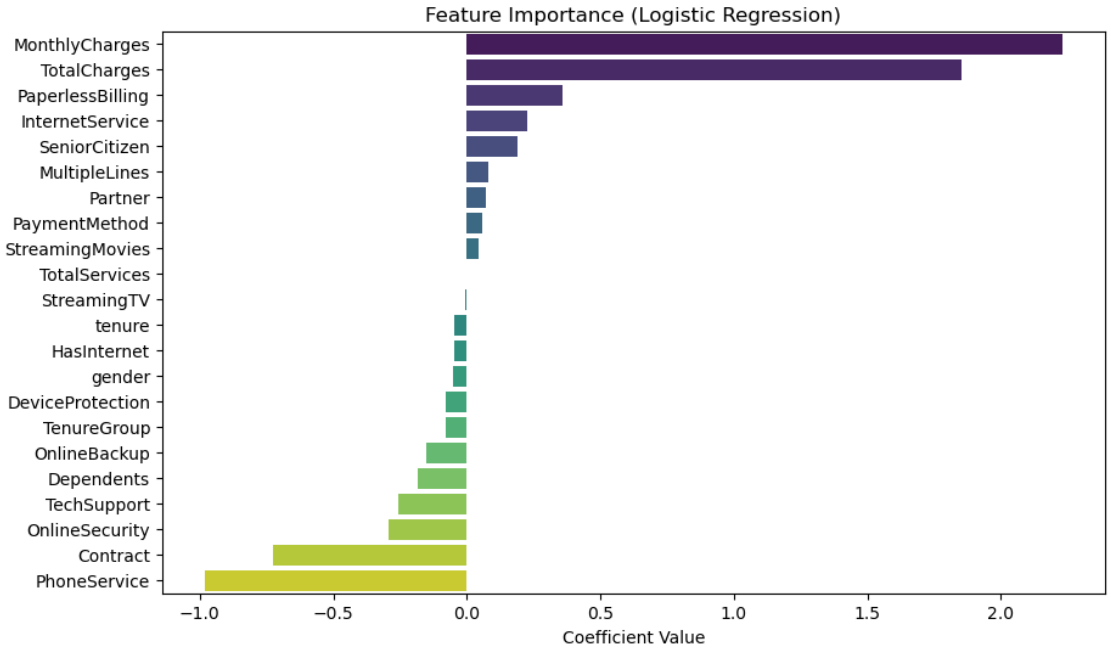
[46]:
```
RandomForestClassifier(class_weight='balanced')
```

[47]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(solver='lbfgs', max_iter=500)  # or 'saga'
model.fit(X_train, y_train)
```

[47]:
```
LogisticRegression(max_iter=500)
```

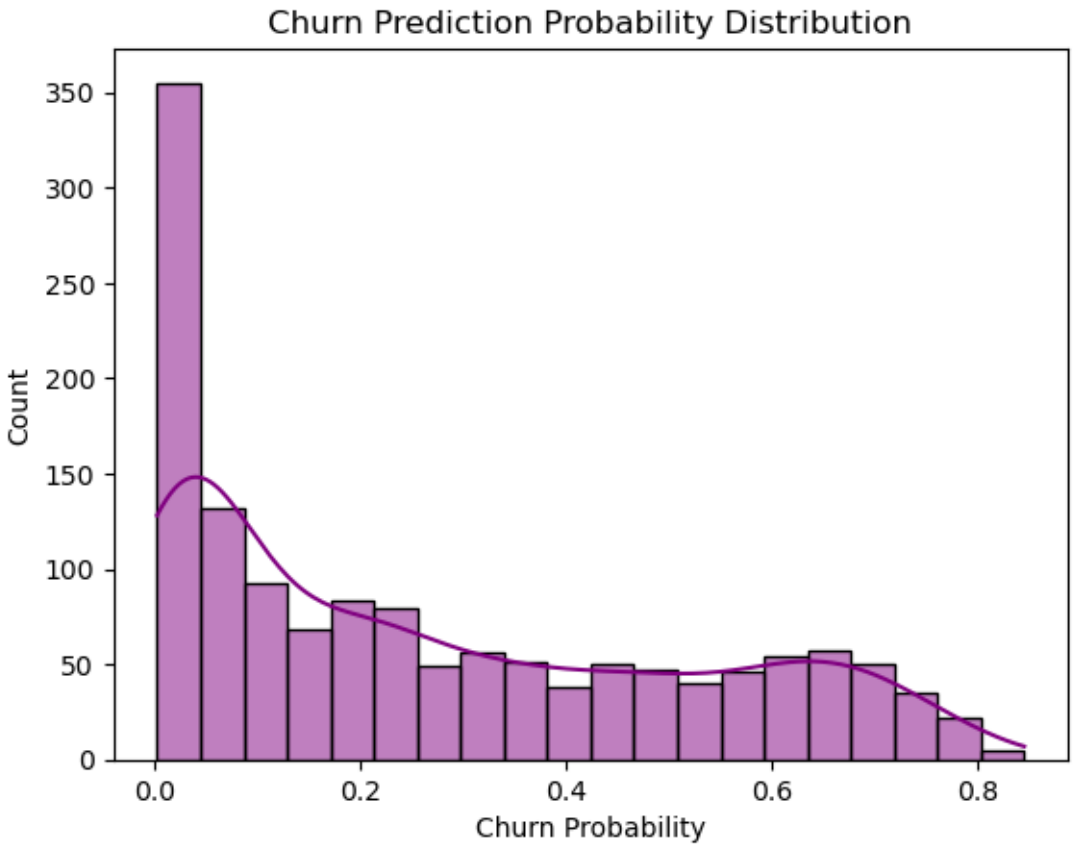
[48]:
```
import pandas as pd
import seaborn as sns

importance = pd.Series(model.coef_[0], index=X.columns).
 ↪sort_values(ascending=False)
plt.figure(figsize=(10, 6))
sns.barplot(x=importance, y=importance.index, palette="viridis")
plt.title("Feature Importance (Logistic Regression)")
plt.xlabel("Coefficient Value")
```

```
plt.show()
```



[49]:
```
import numpy as np
sns.histplot(model.predict_proba(X_test)[:, 1], bins=20, kde=True,␣
 ↪color="purple")
plt.title("Churn Prediction Probability Distribution")
plt.xlabel("Churn Probability")
plt.ylabel("Count")
plt.show()
```

```
C:\Users\Aadiluddin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

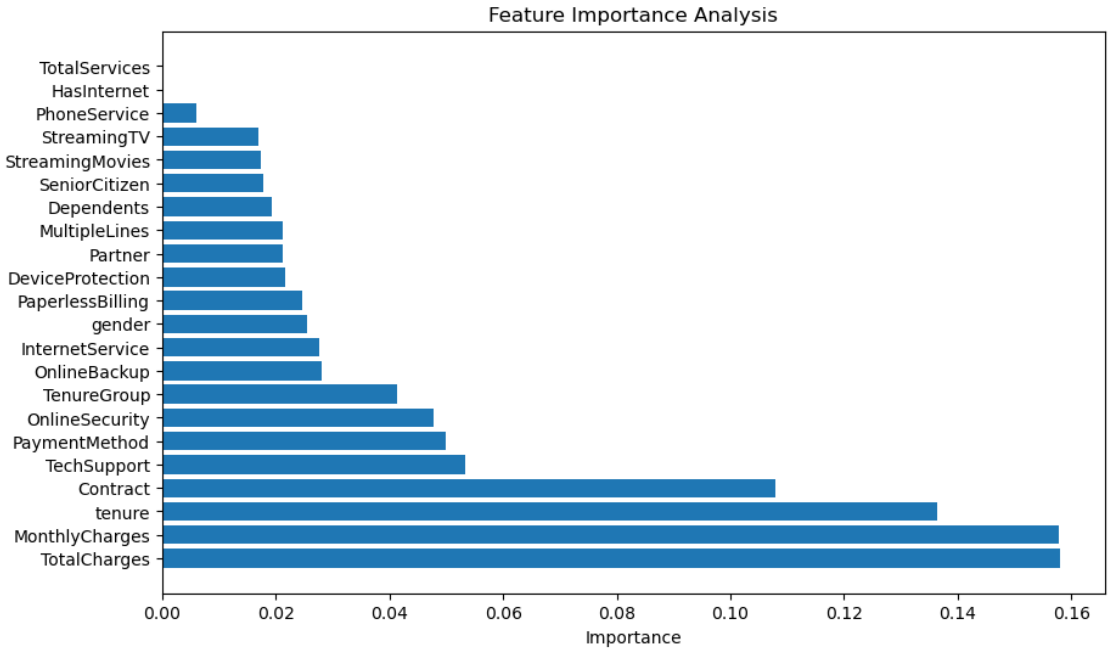


[98]:
```
feature_importances = model.feature_importances_
print("Feature Importances:", feature_importances)
```

```
Feature Importances: [0.02540288 0.01777384 0.02122942 0.01936275 0.13643619
0.0059929
 0.02115753 0.0276382  0.04771106 0.02817676 0.02170233 0.05335517
 0.0169536  0.01739445 0.10790029 0.02466902 0.04994667 0.15777509
 0.15800156 0.         0.         0.04142029]
```

[103]:
```
feature_importances = model.feature_importances_
importance_df = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': feature_importances
})

importance_df = importance_df.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'], importance_df['Importance'])
plt.xlabel('Importance')
```

```
plt.title('Feature Importance Analysis')
plt.show()
```



[ ]: