Customer Churn Prediction

In [51]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
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.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Set up visualization style
sns.set(style="whitegrid")
```

In [52]:
```
df = pd.read_csv("Telco_churn.csv")
df.head(2)
```

Out[52]:

	Unnamed: 0	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	Pape
0	0	7590-VHVEG	Female	False	True	False	1	False	NaN	DSL	...	False	False	False	False	Month-to-month	
1	1	5575-GNVDE	Male	False	False	False	34	True	False	DSL	...	True	False	False	False	One year	

2 rows × 22 columns

EDA

In [53]:
```
df.shape
```

Out[53]: (5043, 22)

In [54]:
```
df.isna().sum()
```

Out[54]:
```
Unnamed: 0            0
customerID            0
gender                0
SeniorCitizen         0
Partner               0
Dependents            0
tenure                0
PhoneService          0
MultipleLines       269
InternetService       0
OnlineSecurity      651
OnlineBackup        651
DeviceProtection    651
TechSupport         651
StreamingTV         651
StreamingMovies     651
Contract              0
PaperlessBilling      0
PaymentMethod         0
MonthlyCharges        0
TotalCharges          5
Churn                 1
dtype: int64
```

Data Cleaning

In [55]:
```
# Drop all rows with any null values
df = df.dropna()
```

In [56]:
```
# Verify if all null values are removed
print(df.isna().sum())
```

```
Unnamed: 0          0
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
```

In [57]:
```
# Identify which columns contain non-numeric values.
for col in numerical_columns:
```

```
    print(f"Column: {col}, Unique Values: {df[col].unique()}")
```

```
Column: tenure, Unique Values: [34  2  8 22 28 62 13 58 49 25 69 71 10 21 30 47 72 17 27  1  5 46 11 70
 63 52 43 15 60 18 66  9  3 31 64 56  7 42 35 65 12 38 68 55 37 33 67 23
 61 14 16 20 53  4 40  6 59 44 19 54 50 41 51 32 57 45 24 29 48 36 39 26
  0]
Column: MonthlyCharges, Unique Values: [ 56.95000076  53.84999847  70.69999695 ...  78.7         60.65
 103.2       ]
Column: TotalCharges, Unique Values: ['1889.5' '108.1500015258789' '151.64999389648438' ... '346.45' '306.6'
 '6844.5']
```

In [58]:
```
# Convert Columns to Numeric:
# If the non-numeric values are due to spaces or other characters, clean the data and convert the columns to numeric.
# Convert columns to numeric, coercing errors (e.g., spaces) to NaN
for col in numerical_columns:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

In [59]:
```
# Drop or Fill NaN Values:
# After converting, handle any NaN values that were introduced during the conversion.
# Drop rows with NaN values in numerical columns
df = df.dropna(subset=numerical_columns)
```

In [60]:
```
# Display basic information about the dataset
print("Dataset Info:")
print(df.info())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
Index: 4118 entries, 1 to 5042
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        4118 non-null   int64
 1   customerID        4118 non-null   object
 2   gender            4118 non-null   object
 3   SeniorCitizen     4118 non-null   object
 4   Partner           4118 non-null   object
 5   Dependents        4118 non-null   object
 6   tenure            4118 non-null   int64
 7   PhoneService      4118 non-null   object
 8   MultipleLines     4118 non-null   object
 9   InternetService   4118 non-null   object
 10  OnlineSecurity    4118 non-null   object
 11  OnlineBackup      4118 non-null   object
 12  DeviceProtection  4118 non-null   object
 13  TechSupport       4118 non-null   object
 14  StreamingTV       4118 non-null   object
 15  StreamingMovies   4118 non-null   object
 16  Contract          4118 non-null   object
 17  PaperlessBilling  4118 non-null   object
 18  PaymentMethod     4118 non-null   object
 19  MonthlyCharges    4118 non-null   float64
 20  TotalCharges      4118 non-null   float64
 21  Churn             4118 non-null   object
dtypes: float64(2), int64(2), object(18)
memory usage: 740.0+ KB
None
```

In [61]:
```
# Display summary statistics for numerical columns
print("\nSummary Statistics for Numerical Columns:")
```

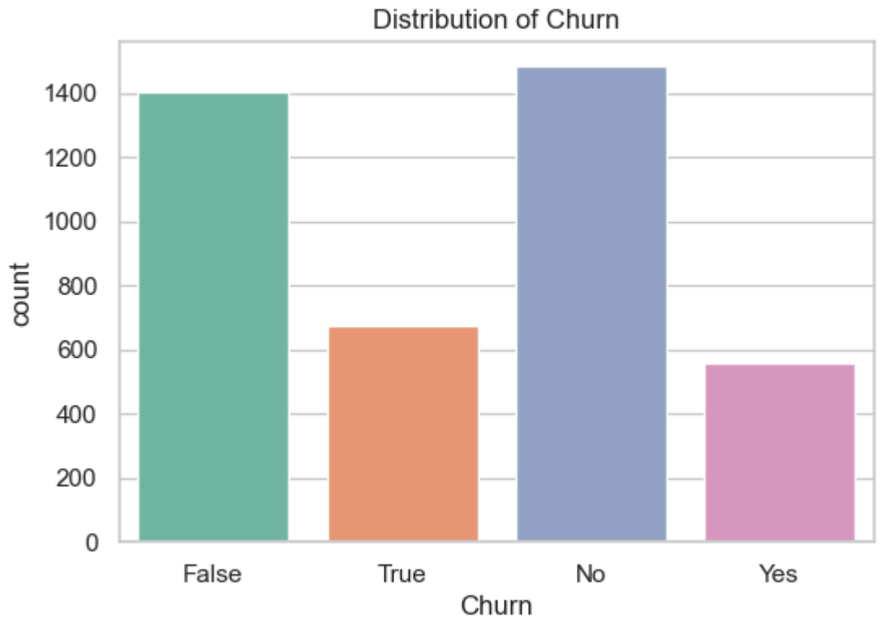
```
print(df.describe())
```

```
Summary Statistics for Numerical Columns:
        Unnamed: 0       tenure  MonthlyCharges  TotalCharges
count  4118.000000  4118.000000     4118.000000   4118.000000
mean   1253.190627    32.853813       73.520435   2607.558100
std     773.866061    24.534154       26.210337   2360.549883
min       0.000000     1.000000       18.550000     18.850000
25%     602.000000     9.000000       55.762500    520.962500
50%    1206.500000    30.000000       78.950000   1828.399988
75%    1817.000000    56.000000       94.199997   4434.700049
max    2998.000000    72.000000      118.650002   8670.100000
```

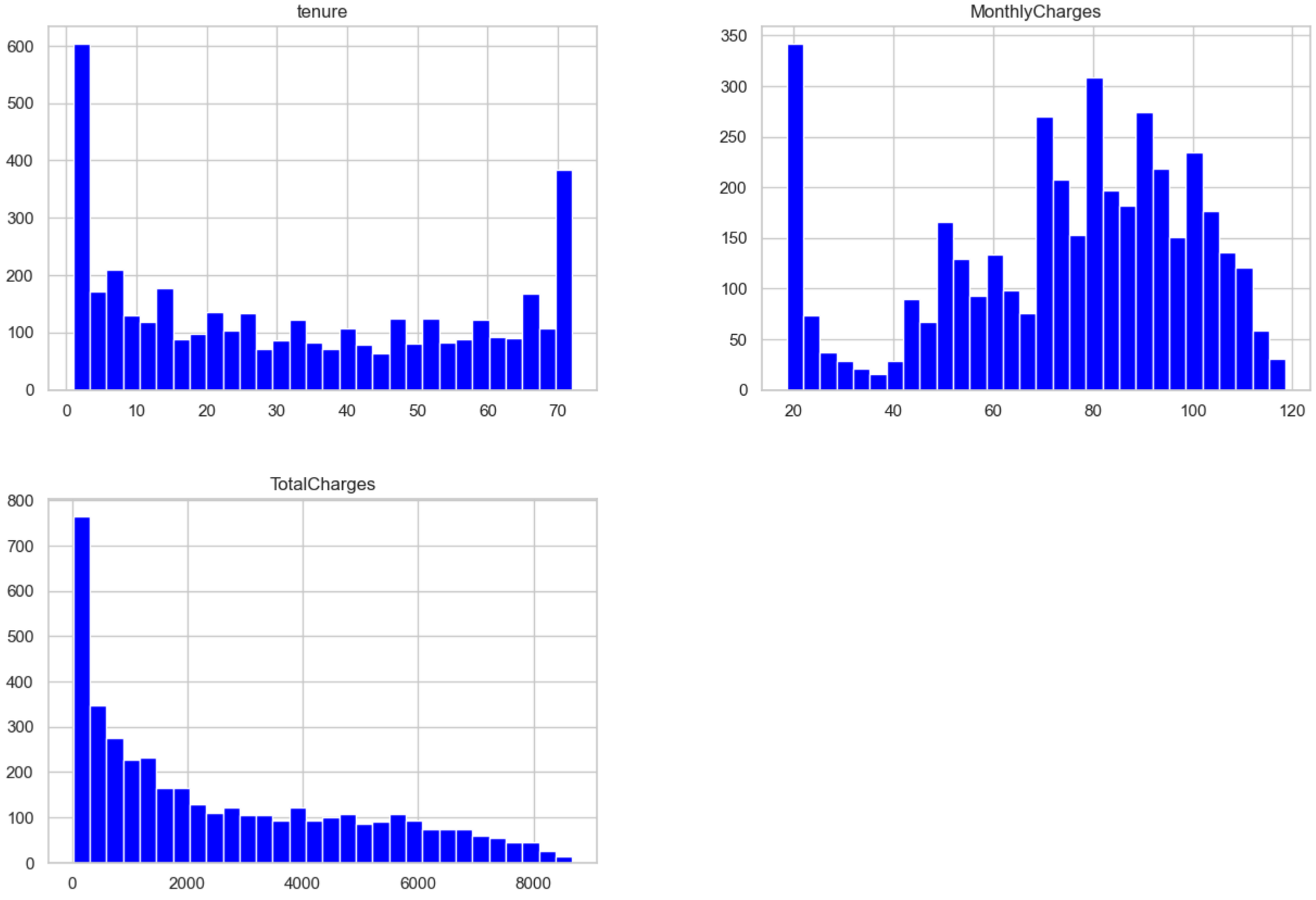
In [62]:
```
# Display summary statistics for categorical columns
print("\nSummary Statistics for Categorical Columns:")
print(df.describe(include=['object']))
```

```
Summary Statistics for Categorical Columns:
        customerID gender SeniorCitizen Partner Dependents PhoneService  \
count         4118   4118          4118    4118       4118         4118
unique        4118      2             4       4          4            3
top     5575-GNVDE   Male             0   False      False         True
freq             1   2074          1696    1059       1515         2078

       MultipleLines InternetService OnlineSecurity OnlineBackup  \
count           4118            4118           4118         4118
unique             5               3              5            5
top             True     Fiber optic          False        False
freq            1136            2247           1325         1168

       DeviceProtection TechSupport StreamingTV StreamingMovies  \
count              4118        4118        4118            4118
unique                5           5           5               5
top               False       False        True            True
freq               1141        1321        1048            1072

              Contract PaperlessBilling     PaymentMethod Churn
count             4118             4118              4118  4118
unique               3                4                 4     4
top     Month-to-month             True  Electronic check    No
freq              2381             1431              1543  1484
```

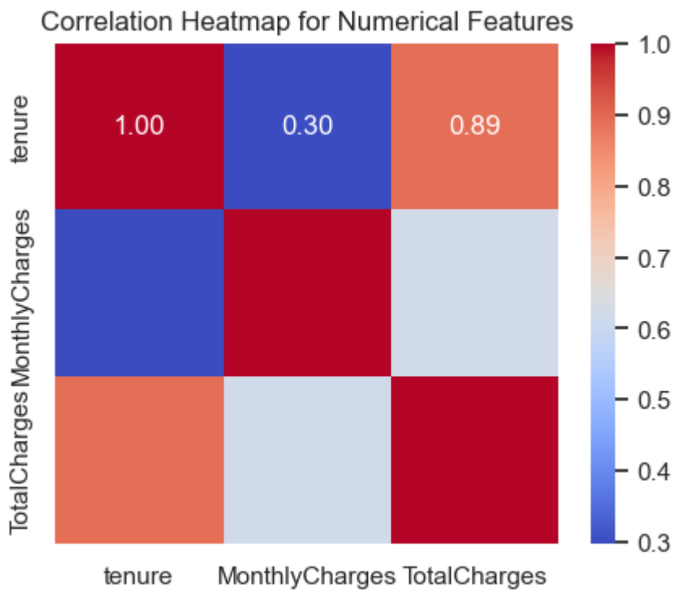
In [63]:
```
# Distribution of the target variable 'Churn'
plt.figure(figsize=(6, 4))
sns.countplot(x='Churn', data=df, palette='Set2')
plt.title('Distribution of Churn')
plt.show()
```



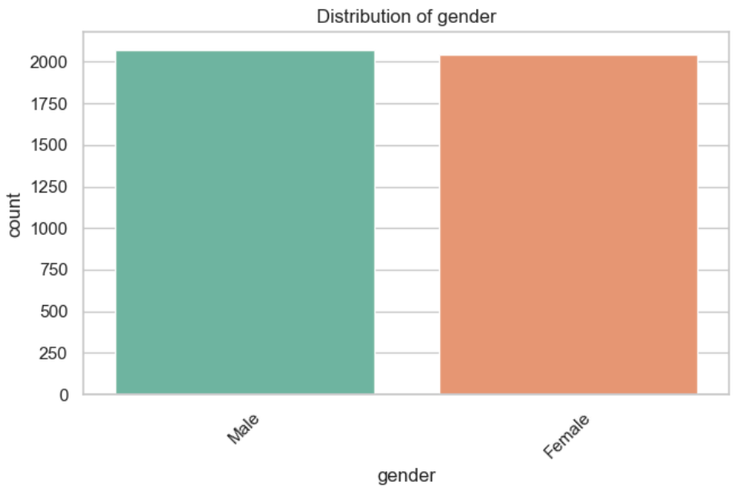
In [64]:
```
# Distribution of numerical features
numerical_columns = ['tenure', 'MonthlyCharges', 'TotalCharges']
df[numerical_columns].hist(bins=30, figsize=(15, 10), color='blue')
plt.suptitle('Distribution of Numerical Features')
plt.show()
```

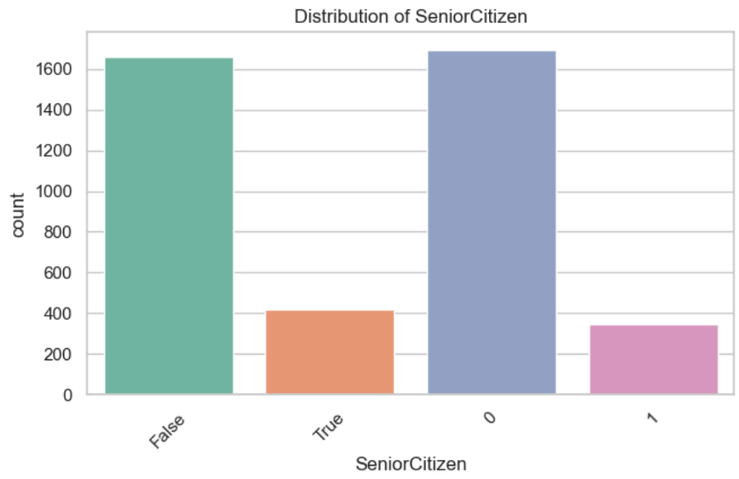


In [65]:
```
plt.figure(figsize=(5, 4))
sns.heatmap(df[numerical_columns].corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap for Numerical Features')
plt.show()
```

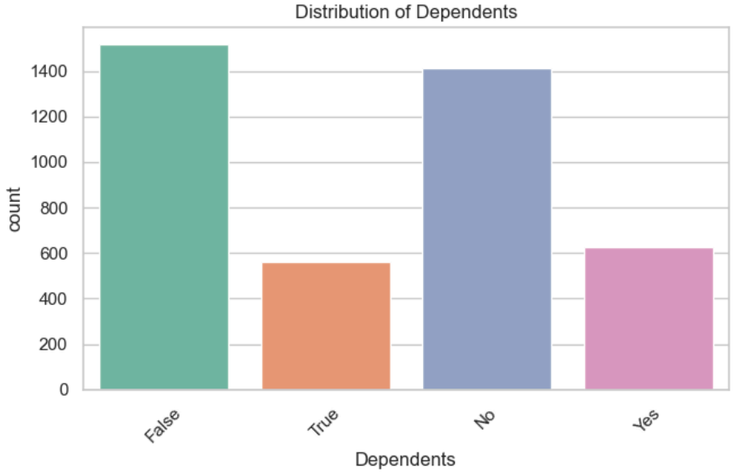


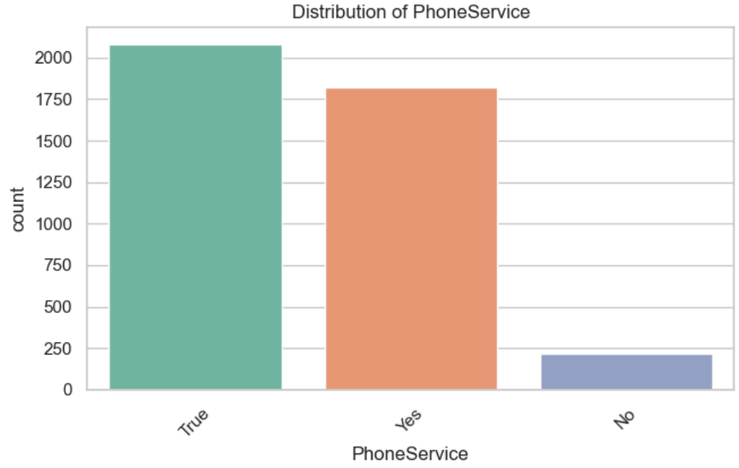
In [66]:
```
# Distribution of categorical features
categorical_columns = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService',
                       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
                       'Contract', 'PaperlessBilling', 'PaymentMethod']

plt.figure(figsize=(20, 30))
for i, column in enumerate(categorical_columns, 1):
    plt.subplot(6, 3, i)
    sns.countplot(x=column, data=df, palette='Set2')
    plt.title(f'Distribution of {column}')
    plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

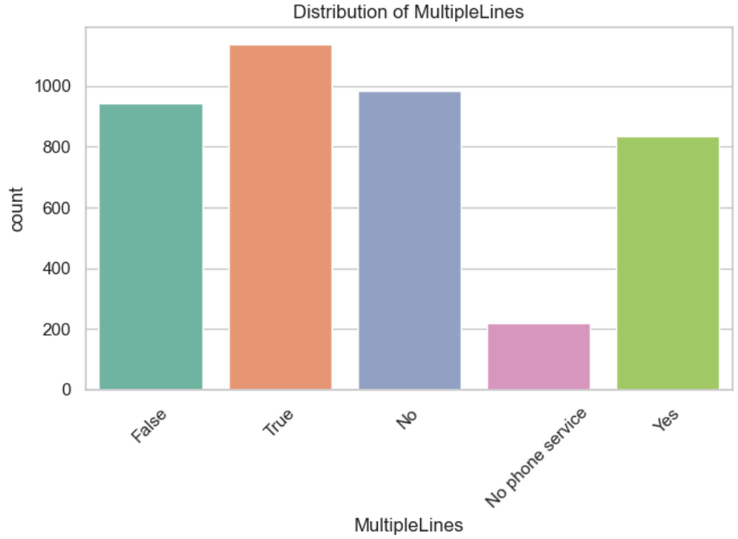

Distribution of gender
count
gender
Male
Female

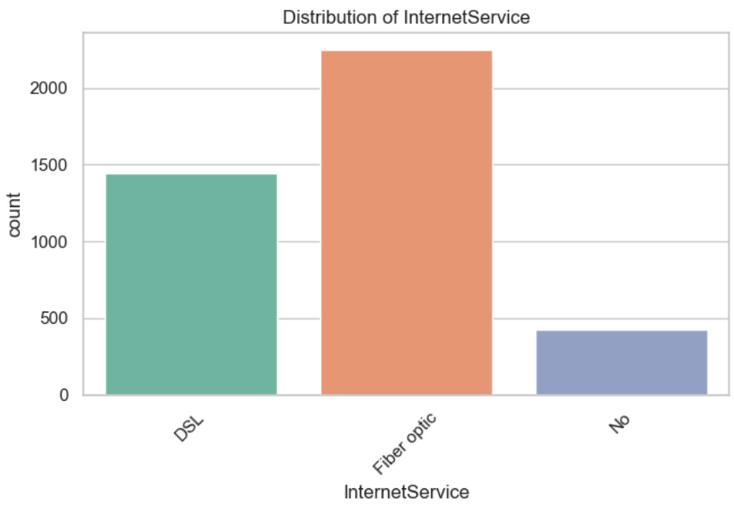

Distribution of SeniorCitizen
count
SeniorCitizen
False
True
0
1

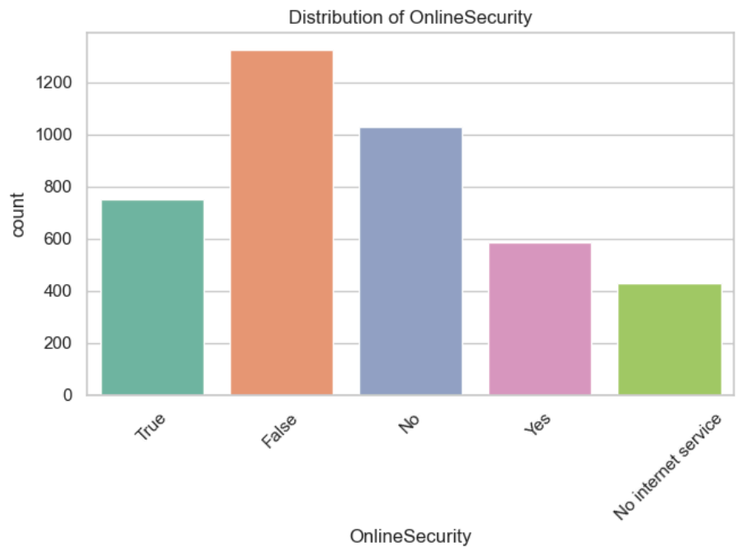

Distribution of Partner
count
Partner
False
True
No
Yes

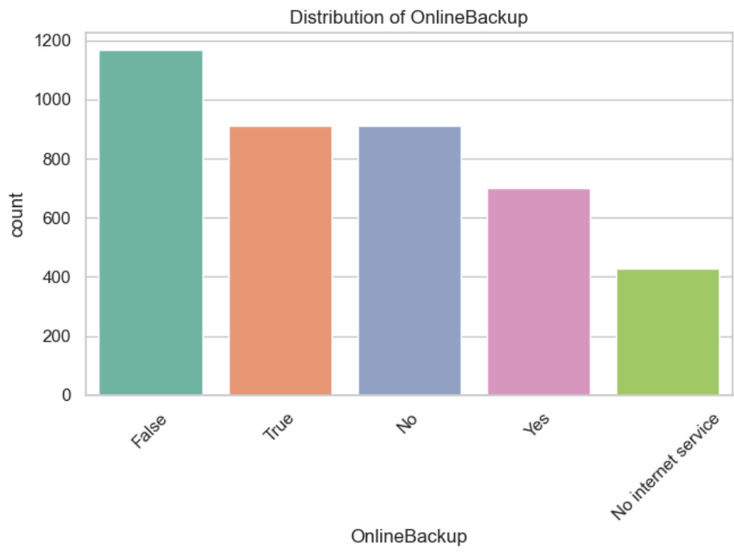

Distribution of Dependents
count
Dependents
False
True
No
Yes

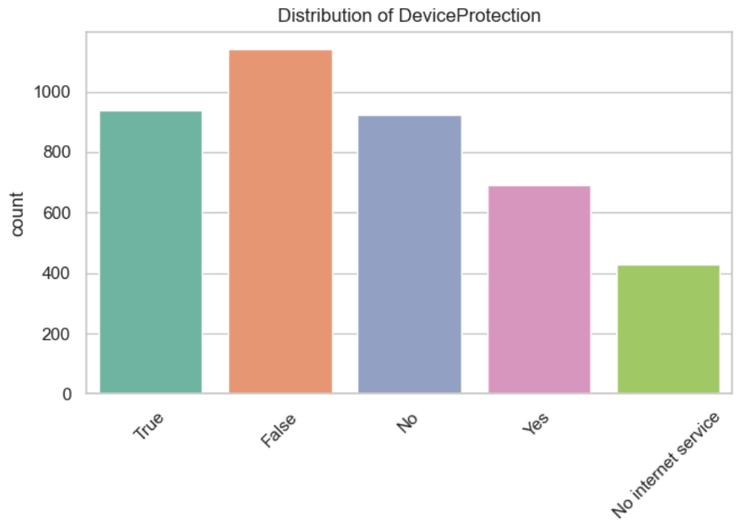

Distribution of PhoneService
count
PhoneService
True
Yes
No

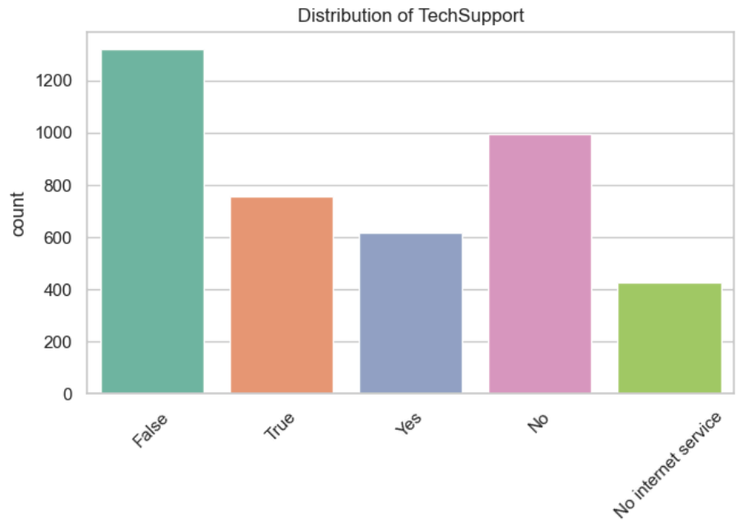

Distribution of MultipleLines
count
MultipleLines
False
True
No
No phone service
Yes

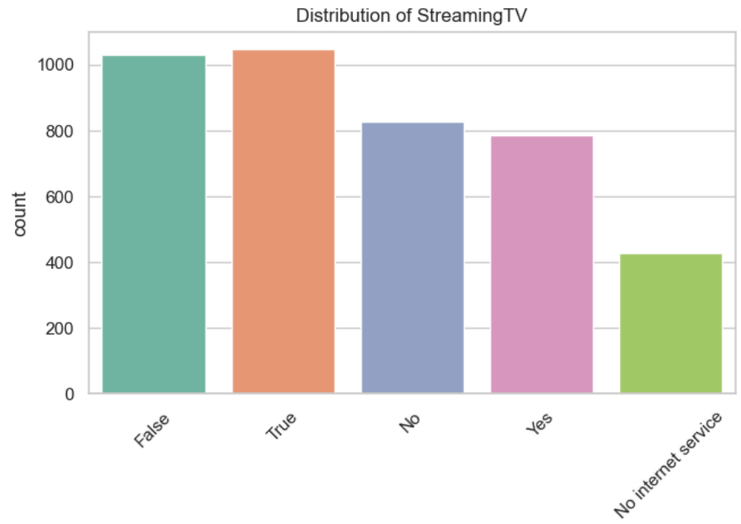

Distribution of InternetService
count
InternetService
DSL
Fiber optic
No

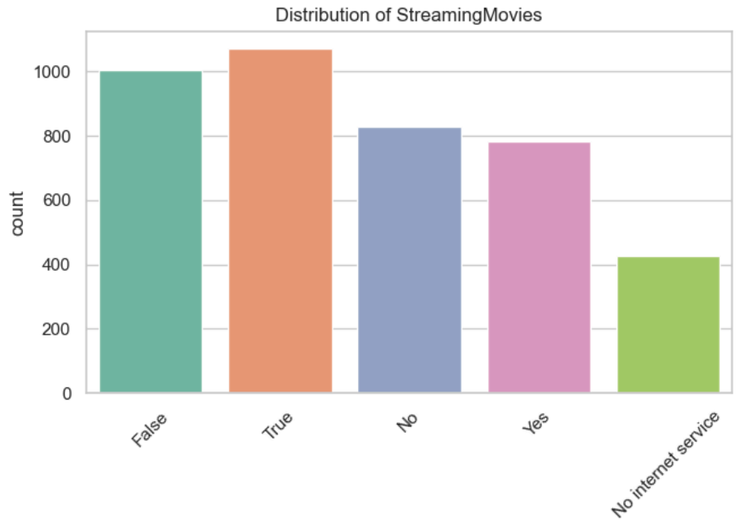

Distribution of OnlineSecurity
count
OnlineSecurity
True
False
No
Yes
No internet service

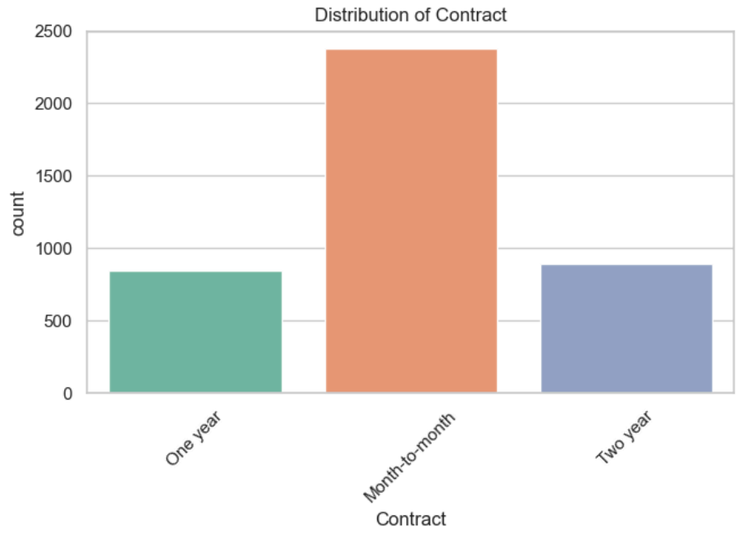

Distribution of OnlineBackup
count
OnlineBackup
False
True
No
Yes
No internet service

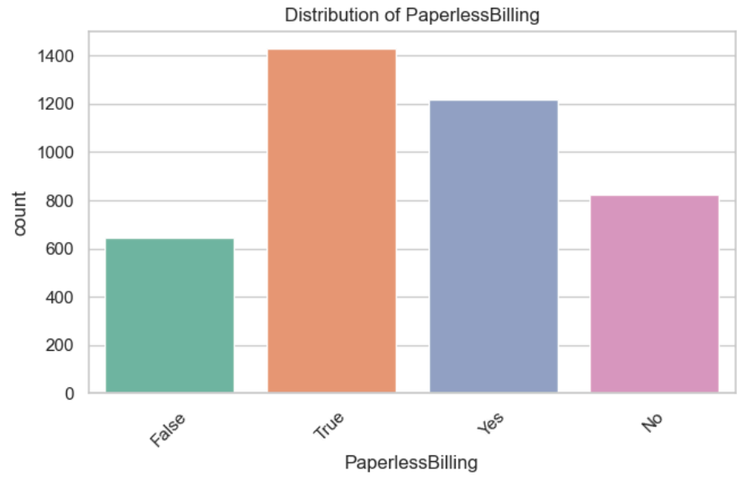

Distribution of DeviceProtection
count
0
200
400
600
800
1000
True
False
No
Yes
No internet service
DeviceProtection

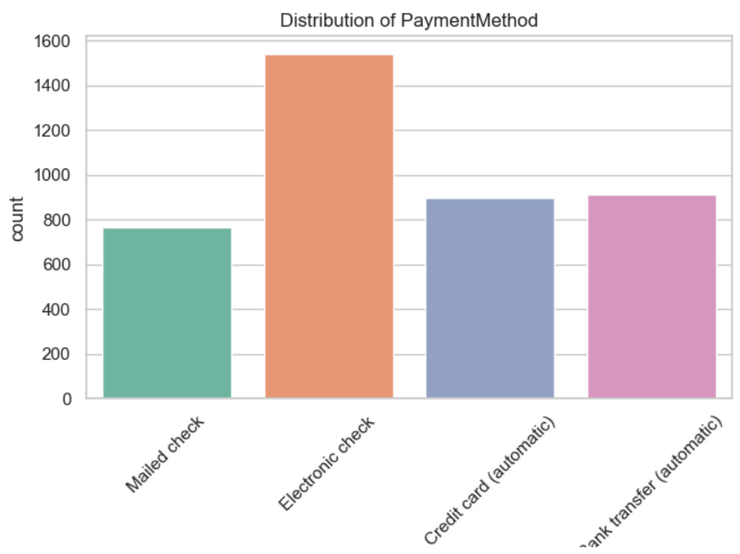

Distribution of TechSupport
count
0
200
400
600
800
1000
1200
False
True
Yes
No
No internet service
TechSupport


Distribution of StreamingTV
count
0
200
400
600
800
1000
False
True
No
Yes
No internet service
StreamingTV


Distribution of StreamingMovies
count
0
200
400
600
800
1000
False
True
No
Yes
No internet service
StreamingMovies

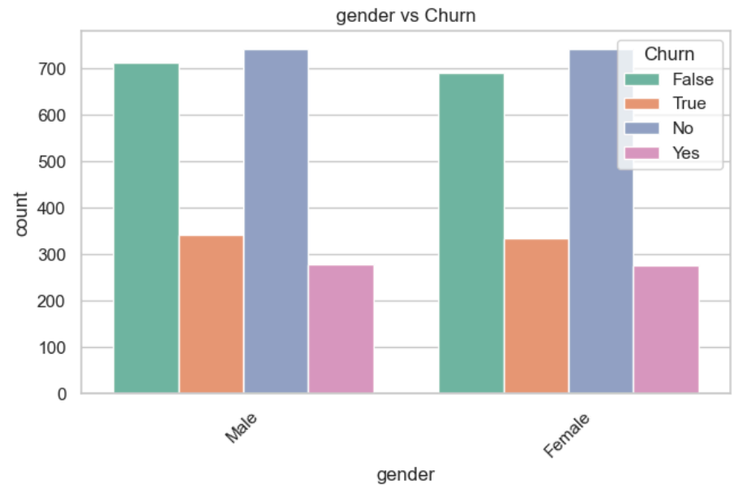

Distribution of Contract
count
0
500
1000
1500
2000
2500
One year
Month-to-month
Two year
Contract

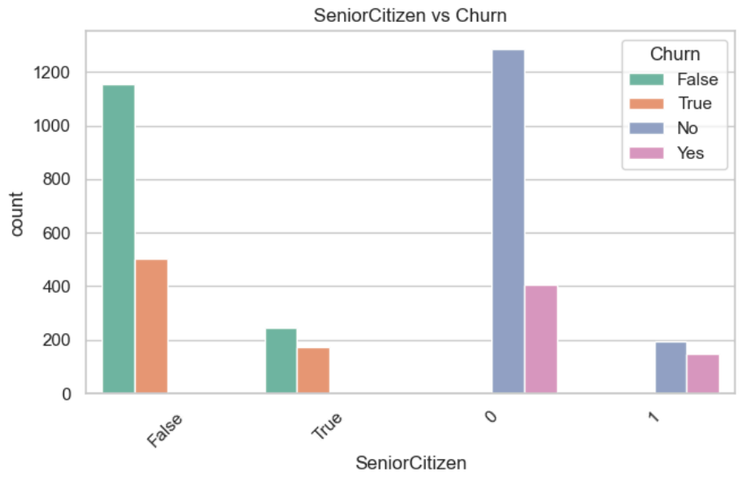

Distribution of PaperlessBilling
count
0
200
400
600
800
1000
1200
1400
False
True
Yes
No
PaperlessBilling

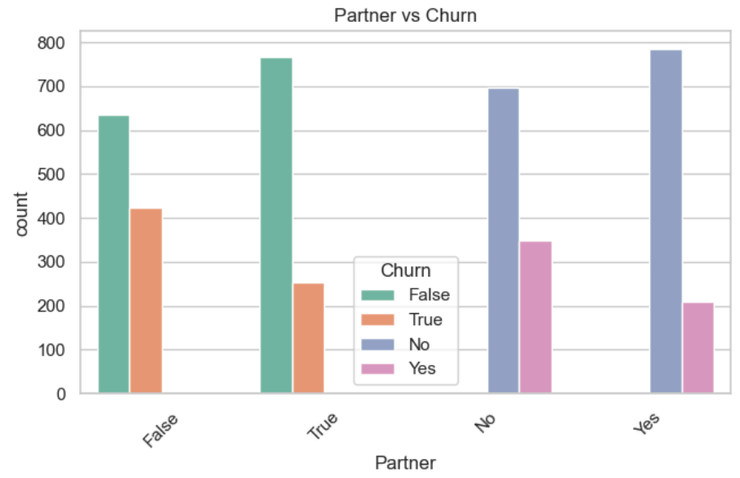

Distribution of PaymentMethod
count
0
200
400
600
800
1000
1200
1400
1600
Mailed check
Electronic check
Credit card (automatic)
Bank transfer (automatic)

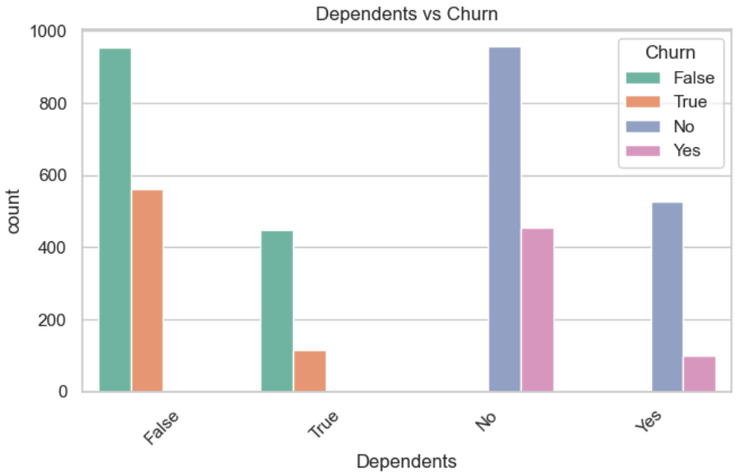
PaymentMethod

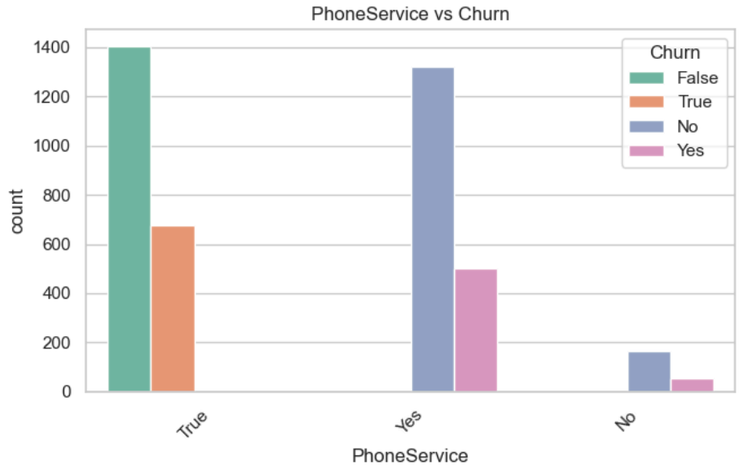
In [67]:
```
# Relationship between categorical features and Churn
plt.figure(figsize=(20, 30))
for i, column in enumerate(categorical_columns, 1):
    plt.subplot(6, 3, i)
    sns.countplot(x=column, hue='Churn', data=df, palette='Set2')
    plt.title(f'{column} vs Churn')
    plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

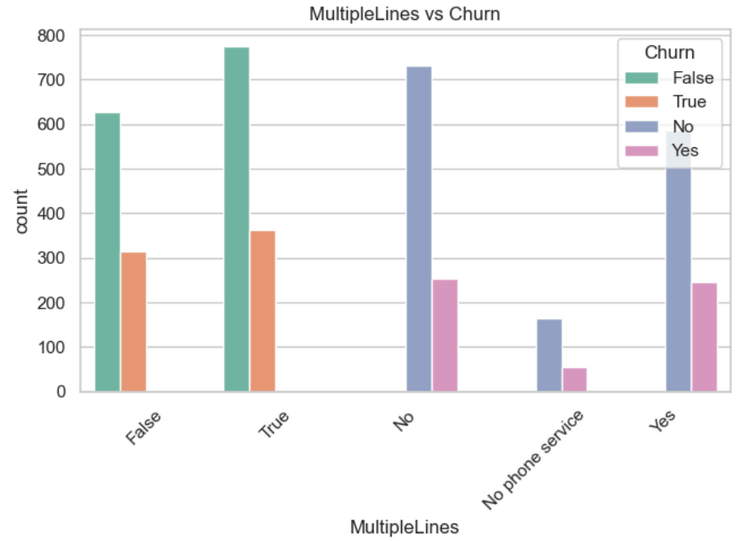

gender vs Churn
Churn
False
True
No
Yes
count
gender
Male
Female

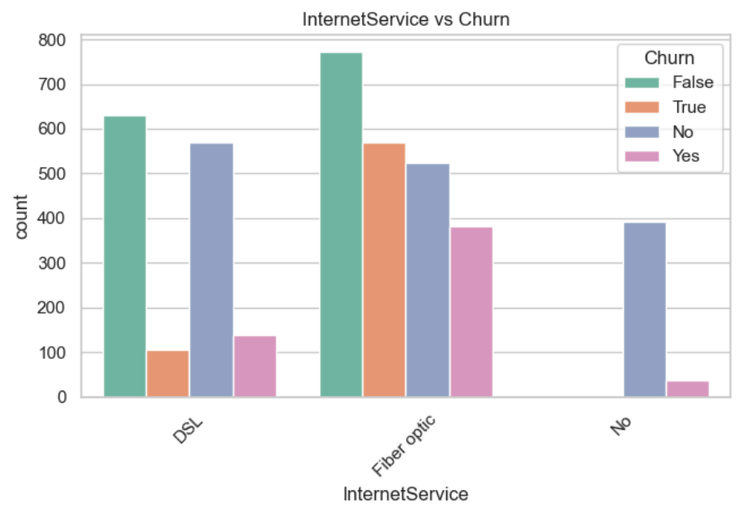

SeniorCitizen vs Churn
Churn
False
True
No
Yes
count
SeniorCitizen
False
True
0
1

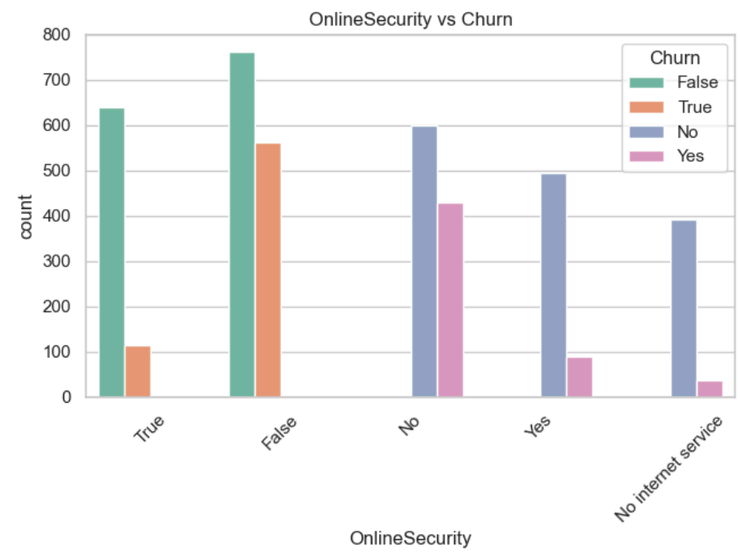

Partner vs Churn
Churn
False
True
No
Yes
count
Partner
False
True
No
Yes

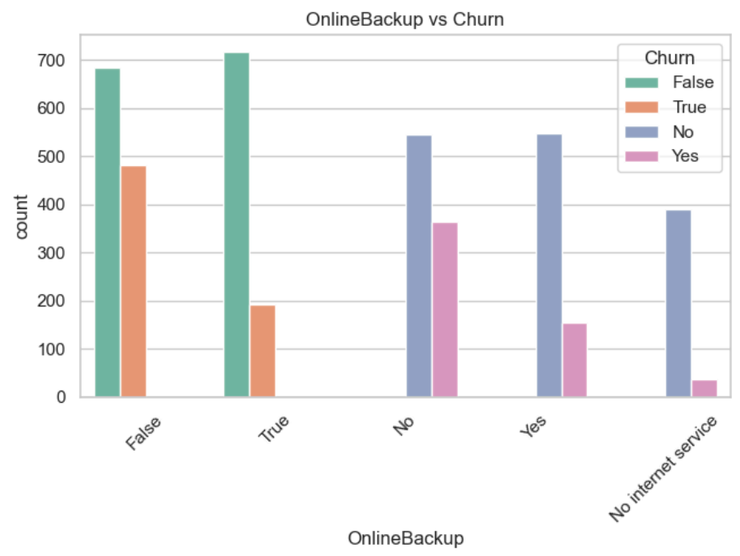

Dependents vs Churn
Churn
False
True
No
Yes
count
Dependents
False
True
No
Yes

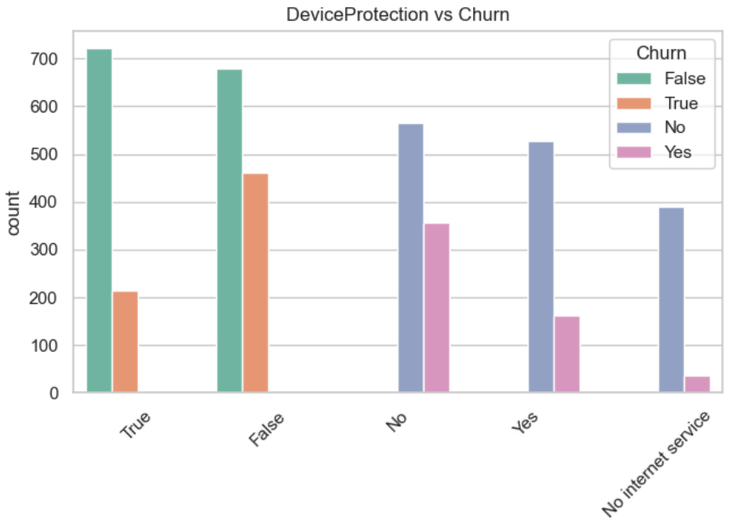

PhoneService vs Churn
Churn
False
True
No
Yes
count
PhoneService
True
Yes
No

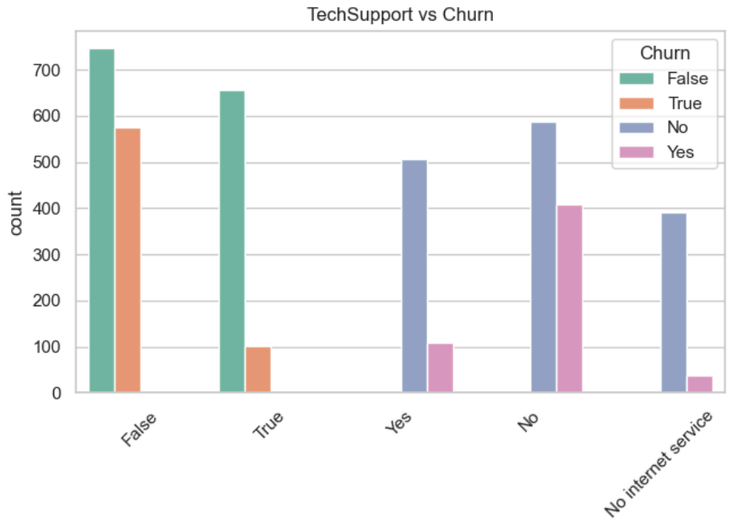

MultipleLines vs Churn
Churn
False
True
No
Yes
count
MultipleLines
False
True
No
No phone service
Yes

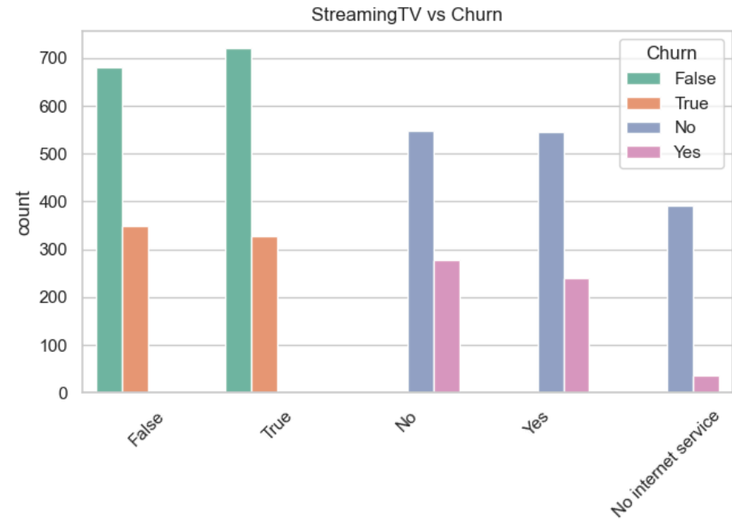

InternetService vs Churn
Churn
False
True
No
Yes
count
InternetService
DSL
Fiber optic
No

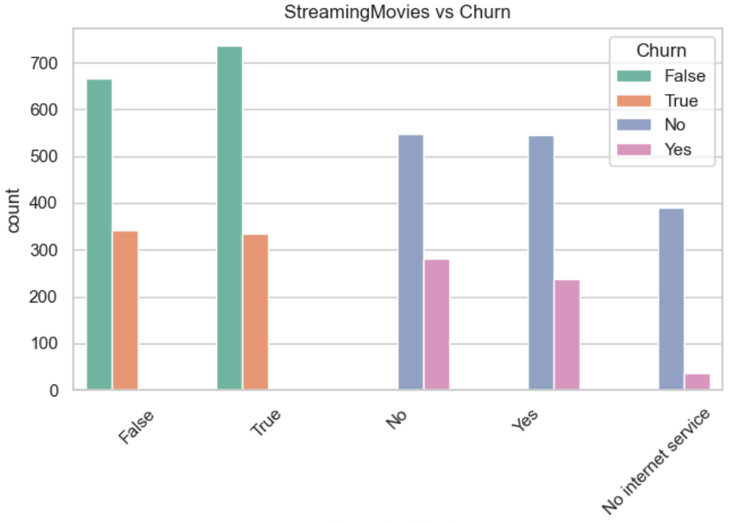

OnlineSecurity vs Churn
Churn
False
True
No
Yes
count
OnlineSecurity
True
False
No
Yes
No internet service

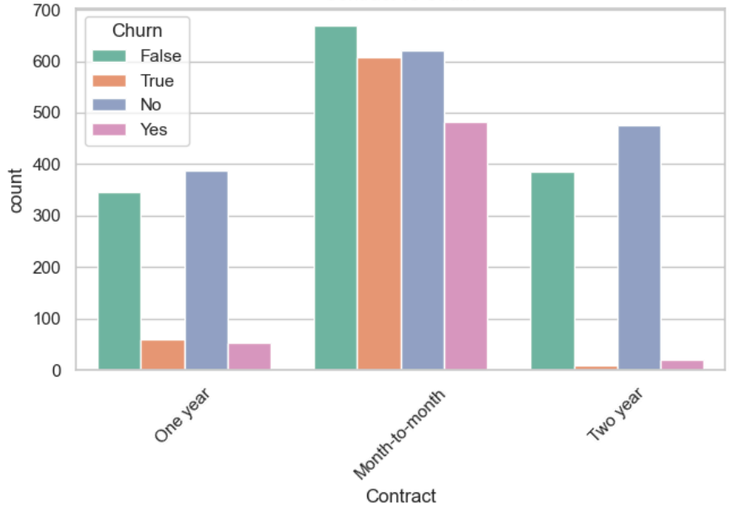

OnlineBackup vs Churn
Churn
False
True
No
Yes
count
OnlineBackup
False
True
No
Yes
No internet service

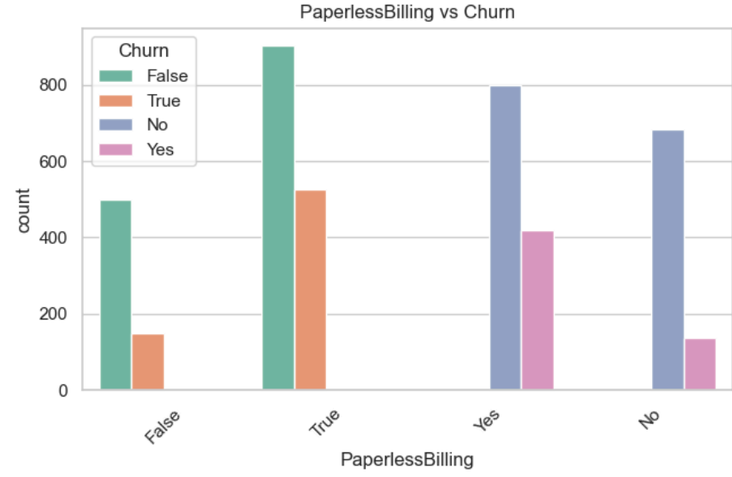

DeviceProtection vs Churn
Churn
False
True
No
Yes
count
0
100
200
300
400
500
600
700
True
False
No
Yes
No internet service
DeviceProtection

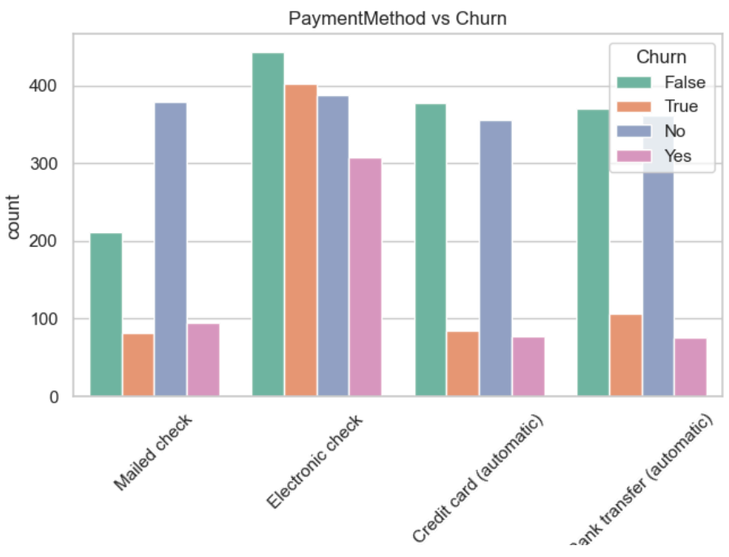

TechSupport vs Churn
Churn
False
True
No
Yes
count
0
100
200
300
400
500
600
700
False
True
Yes
No
No internet service
TechSupport


StreamingTV vs Churn
Churn
False
True
No
Yes
count
0
100
200
300
400
500
600
700
False
True
No
Yes
No internet service
StreamingTV


StreamingMovies vs Churn
Churn
False
True
No
Yes
count
0
100
200
300
400
500
600
700
False
True
No
Yes
No internet service
StreamingMovies

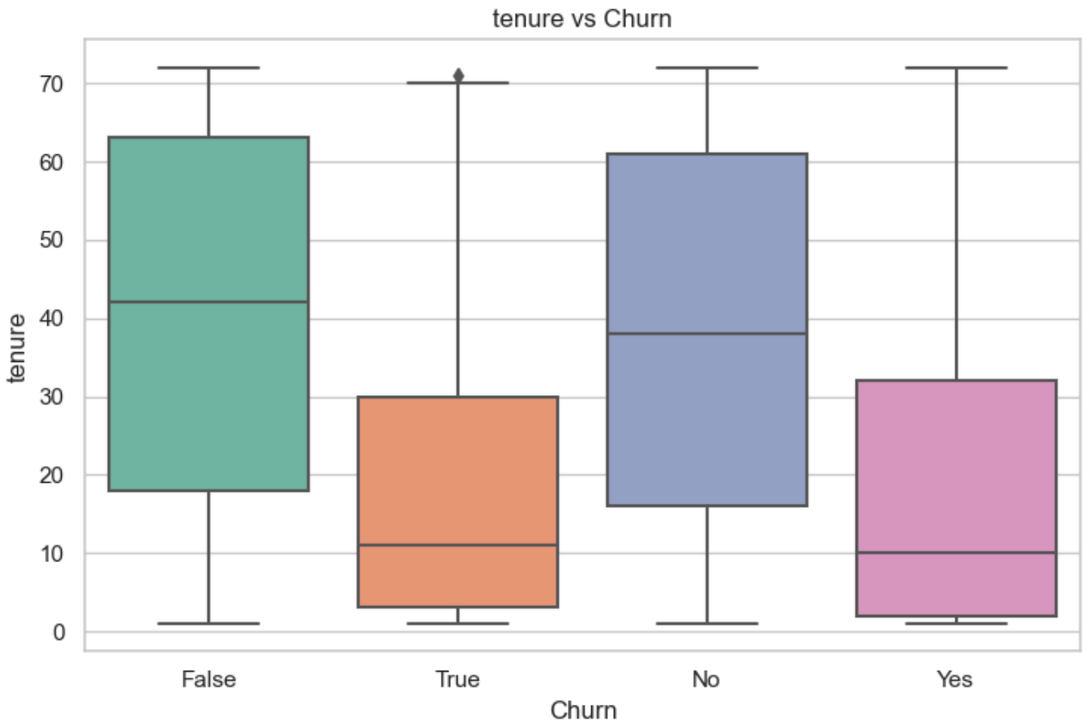

Contract vs Churn
Churn
False
True
No
Yes
count
0
100
200
300
400
500
600
700
One year
Month-to-month
Two year
Contract

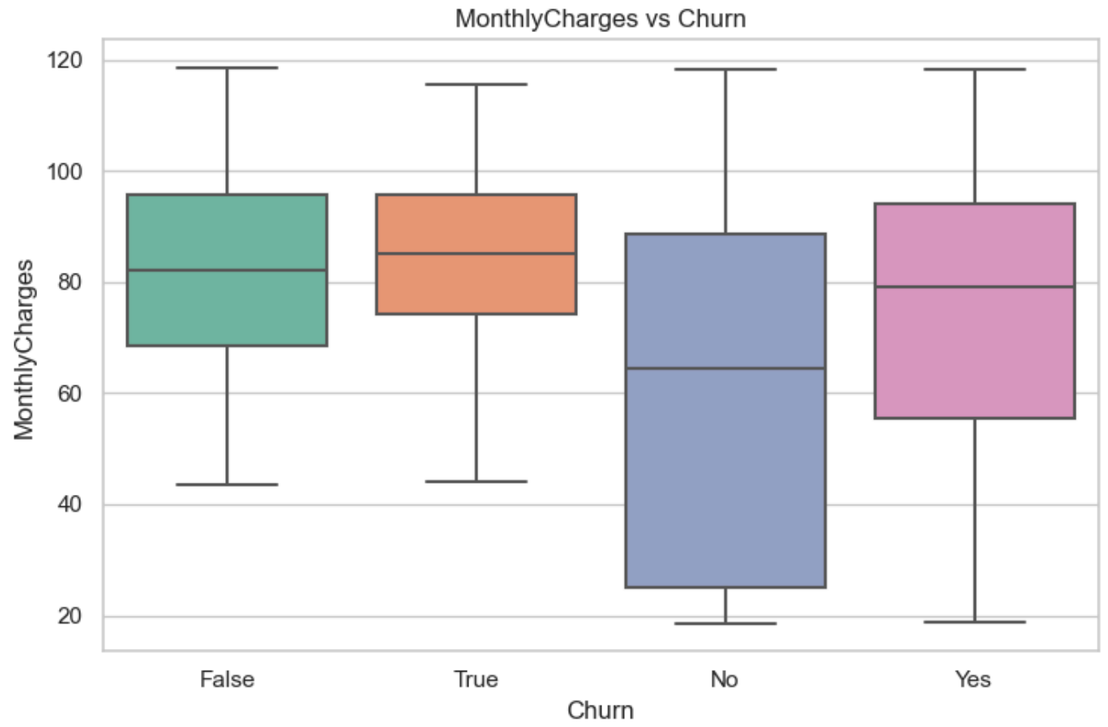

PaperlessBilling vs Churn
Churn
False
True
No
Yes
count
0
200
400
600
800
False
True
Yes
No
PaperlessBilling

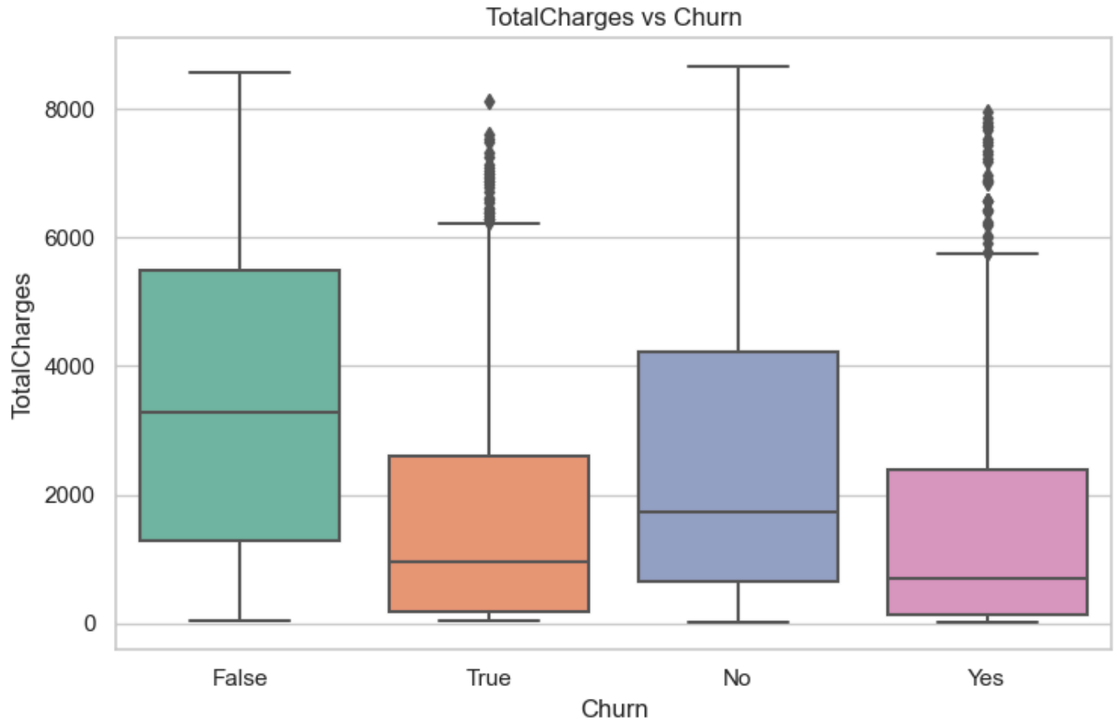

PaymentMethod vs Churn
Churn
False
True
No
Yes
count
0
100
200
300
400
Mailed check
Electronic check
Credit card (automatic)
Bank transfer (automatic)

PaymentMethod

In [68]:
```
# Relationship between numerical features and Churn
plt.figure(figsize=(15, 10))
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(2, 2, i)
    sns.boxplot(x='Churn', y=column, data=df, palette='Set2')
    plt.title(f'{column} vs Churn')
plt.tight_layout()
plt.show()
```







Step 1: Data Preprocessing

In [69]:
```
# Define categorical and numerical columns
categorical_columns = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
```

```
                      'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
                      'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod']

numerical_columns = ['tenure', 'MonthlyCharges', 'TotalCharges']
```

In [70]:
```
# Create a ColumnTransformer to preprocess categorical and numerical features
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_columns),  # Scale numerical features
        ('cat', OneHotEncoder(drop='first'), categorical_columns)  # Encode categorical features
    ])
```

Step 2: Split the data into training and testing sets

In [71]:
```
X = df.drop('Churn', axis=1)  # Features
y = df['Churn']  # Target variable

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

Step 3: Build a Pipeline for Model Training

In [72]:
```
# Model 1
# Logistic Regression Model
logreg_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=42))
])
```

In [73]:
```
# Model 2
# Random Forest Model
rf_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(random_state=42))
])
```

Step 4: Train and Evaluate Models

In [74]:
```
# Logistic Regression
logreg_pipeline.fit(X_train, y_train)
y_pred_logreg = logreg_pipeline.predict(X_test)

print("Logistic Regression Results:")
print(classification_report(y_test, y_pred_logreg))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_logreg))
print("Accuracy:", accuracy_score(y_test, y_pred_logreg))
```

```
Logistic Regression Results:
              precision    recall  f1-score   support

       False       0.81      0.84      0.82       286
          No       0.86      0.88      0.87       286
        True       0.67      0.61      0.64       149
         Yes       0.65      0.59      0.62       103

    accuracy                           0.78       824
   macro avg       0.75      0.73      0.74       824
weighted avg       0.78      0.78      0.78       824

Confusion Matrix:
[[241   0  45   0]
 [  0 253   0  33]
 [ 58   0  91   0]
 [  0  42   0  61]]
Accuracy: 0.7839805825242718
```

In [75]:
```python
# Random Forest
rf_pipeline.fit(X_train, y_train)
y_pred_rf = rf_pipeline.predict(X_test)

print("\nRandom Forest Results:")
print(classification_report(y_test, y_pred_rf))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_rf))
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
```

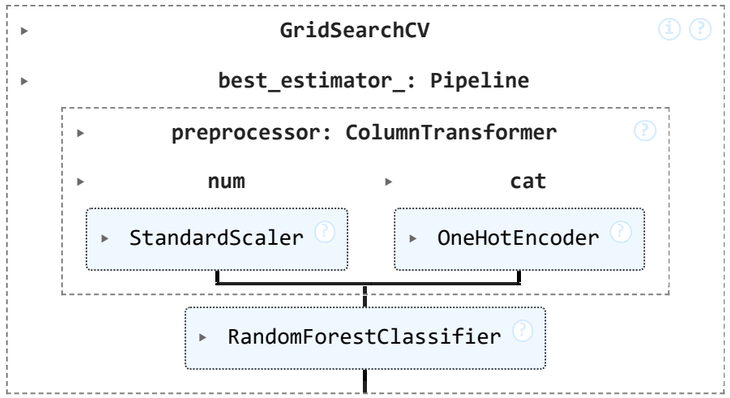
```
Random Forest Results:
              precision    recall  f1-score   support

       False       0.77      0.87      0.82       286
          No       0.83      0.90      0.86       286
        True       0.67      0.50      0.57       149
         Yes       0.64      0.49      0.55       103

    accuracy                           0.77       824
   macro avg       0.73      0.69      0.70       824
weighted avg       0.76      0.77      0.76       824

Confusion Matrix:
[[249   0  37   0]
 [  0 258   0  28]
 [ 74   0  75   0]
 [  0  53   0  50]]
Accuracy: 0.7669902912621359
```

Step 5: Model Optimization (Optional)

In [76]:
```python
from sklearn.model_selection import GridSearchCV

# Define hyperparameters for Random Forest
param_grid = {
    'classifier__n_estimators': [100, 200],
    'classifier__max_depth': [None, 10, 20],
    'classifier__min_samples_split': [2, 5]
}

# Perform Grid Search
```

```
grid_search = GridSearchCV(rf_pipeline, param_grid, cv=3, scoring='accuracy')
grid_search.fit(X_train, y_train)
```

Out[76]:



In [77]:
```
# Best parameters and score
print("\nBest Parameters:", grid_search.best_params_)
print("Best Accuracy:", grid_search.best_score_)
```

```
Best Parameters: {'classifier__max_depth': 10, 'classifier__min_samples_split': 2, 'classifier__n_estimators': 100}
Best Accuracy: 0.7777777777777778
```

In [79]:
```
# Evaluate the optimized model
y_pred_optimized = grid_search.predict(X_test)
print("\nOptimized Random Forest Results:")
print(classification_report(y_test, y_pred_optimized))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_optimized))
print("Accuracy:", accuracy_score(y_test, y_pred_optimized))
```

```
Optimized Random Forest Results:
              precision    recall  f1-score   support

       False       0.79      0.88      0.83       286
          No       0.82      0.92      0.87       286
        True       0.71      0.56      0.63       149
         Yes       0.67      0.45      0.53       103

    accuracy                           0.78       824
   macro avg       0.75      0.70      0.72       824
weighted avg       0.77      0.78      0.77       824

Confusion Matrix:
[[251   0  35   0]
 [  0 263   0  23]
 [ 65   0  84   0]
 [  0  57   0  46]]
Accuracy: 0.7815533980582524
```

Feature Importance:

Analyze feature importance from the Random Forest model to understand which features contribute most to predictions.

In [80]:
```
importances = grid_search.best_estimator_.named_steps['classifier'].feature_importances_
feature_names = numerical_columns + list(grid_search.best_estimator_.named_steps['preprocessor']
                                         .named_transformers_['cat'].get_feature_names_out(categorical_columns))
```

```
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
print(feature_importance_df.sort_values(by='Importance', ascending=False))
```

```
                                Feature  Importance
13                     PhoneService_True    0.143277
14                      PhoneService_Yes    0.084328
0                                tenure    0.074112
2                           TotalCharges    0.073987
5                     SeniorCitizen_False    0.063246
1                           MonthlyCharges    0.054204
48                 PaperlessBilling_True    0.041606
10                         Dependents_No    0.037558
49                  PaperlessBilling_Yes    0.031803
19           InternetService_Fiber optic    0.024880
9                            Partner_Yes    0.021432
21                    OnlineSecurity_No    0.020847
17                   MultipleLines_True    0.018800
46                    Contract_Two year    0.016200
39                     StreamingTV_True    0.015901
35                     TechSupport_True    0.015695
47                  PaperlessBilling_No    0.015014
8                            Partner_True    0.014829
31                DeviceProtection_True    0.014819
51        PaymentMethod_Electronic check    0.014716
27                    OnlineBackup_True    0.013868
7                             Partner_No    0.013683
45                    Contract_One year    0.013096
23                   OnlineSecurity_True    0.011289
15                     MultipleLines_No    0.011144
29                   DeviceProtection_No    0.009040
25                     OnlineBackup_No    0.008972
6                     SeniorCitizen_True    0.008935
3                             gender_Male    0.008705
41                   StreamingMovies_No    0.007500
50  PaymentMethod_Credit card (automatic)    0.007436
18                    MultipleLines_Yes    0.007153
32                  DeviceProtection_Yes    0.007096
28                      OnlineBackup_Yes    0.006404
33                        TechSupport_No    0.006390
12                        Dependents_Yes    0.006222
43                   StreamingMovies_True    0.006069
24                     OnlineSecurity_Yes    0.005426
52            PaymentMethod_Mailed check    0.005147
36                        TechSupport_Yes    0.004646
4                         SeniorCitizen_1    0.004639
37                         StreamingTV_No    0.004540
44                    StreamingMovies_Yes    0.004108
11                        Dependents_True    0.003492
16        MultipleLines_No phone service    0.003365
38       StreamingTV_No internet service    0.002888
34       TechSupport_No internet service    0.002787
40                        StreamingTV_Yes    0.002430
26      OnlineBackup_No internet service    0.002248
20                     InternetService_No    0.001735
42   StreamingMovies_No internet service    0.000908
22    OnlineSecurity_No internet service    0.000732
30  DeviceProtection_No internet service    0.000659
```

Deployment:

Save the trained model using joblib or pickle for deployment.

In [82]:
```
import joblib
joblib.dump(grid_search.best_estimator_, 'optimized_random_forest_model.pkl')
```

Out[82]:
```
['optimized_random_forest_model.pkl']
```