In [107…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from scipy.stats import chi2_contingency
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import (
    RandomForestClassifier,
    AdaBoostClassifier,
    VotingClassifier,
    BaggingClassifier,
    GradientBoostingClassifier,
    ExtraTreesClassifier,
)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV

from sklearn.metrics import classification_report, confusion_matrix
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:

```
df=pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Mul
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	

5 rows × 21 columns

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

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

In [5]:
```
df.columns.values
```

Out[5]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

1. **customerID**: Unique identifier for each customer.
2. **gender**: Gender of the customer (male or female).
3. **SeniorCitizen**: Indicates if the customer is a senior citizen (1 for Yes, 0 for No).
4. **Partner**: Indicates if the customer has a partner (Yes or No).
5. **Dependents**: Indicates if the customer has dependents (Yes or No).
6. **tenure**: Number of months the customer has stayed with the company.
7. **PhoneService**: Indicates if the customer has phone service (Yes or No).
8. **MultipleLines**: Indicates if the customer has multiple phone lines (Yes, No, or No phone service).
9. **InternetService**: Type of internet service the customer has (DSL, Fiber optic, or No).
10. **OnlineSecurity**: Indicates if the customer has online security service (Yes, No, or No internet service).
11. **OnlineBackup**: Indicates if the customer has online backup service (Yes, No, or No internet service).
12. **DeviceProtection**: Indicates if the customer has device protection plan (Yes, No, or No internet service).
13. **TechSupport**: Indicates if the customer has tech support service (Yes, No, or No internet service).
14. **StreamingTV**: Indicates if the customer has streaming TV service (Yes, No, or No internet service).
15. **StreamingMovies**: Indicates if the customer has streaming movies service (Yes, No, or No internet service).
16. **Contract**: Type of contract the customer has (Month-to-month, One year, or Two year).
17. **PaperlessBilling**: Indicates if the customer uses paperless billing (Yes or No).
18. **PaymentMethod**: Payment method used by the customer (Electronic check, Mailed check, Bank transfer, or Credit card).
19. **MonthlyCharges**: Monthly charges incurred by the customer.
20. **TotalCharges**: Total charges incurred by the customer.
21. **Churn**: Indicates if the customer has churned (Yes or No).

In [6]:
```
df.info(verbose=True)
```

```
<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
```

In [7]:

```
df.describe().transpose()
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
SeniorCitizen	7043.0	0.162147	0.368612	0.00	0.0	0.00	0.00	1.00
tenure	7043.0	32.371149	24.559481	0.00	9.0	29.00	55.00	72.00
MonthlyCharges	7043.0	64.761692	30.090047	18.25	35.5	70.35	89.85	118.75

Insight

1. **Senior Citizens**: Most customers are not senior citizens.
2. **Tenure**: 75% of customers have been with the company for less than 55 months.
3. **Outliers**: The mean and median for tenure and monthly charges are close, so there likely aren't any extreme values (outliers). However, using a boxplot can confirm this.
4. **Monthly Charges**: The average monthly charge for customers is about 64.76 USD.

In [8]:

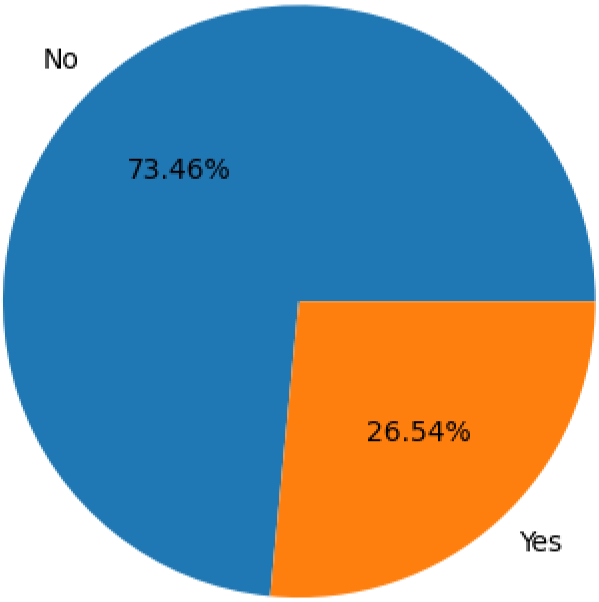
```
(df['Churn'].value_counts()/df.shape[0])*100
```

Out[8]:

```
No     73.463013
Yes    26.536987
Name: Churn, dtype: float64
```

In [9]:

```
plt.pie(x=df['Churn'].value_counts().values,labels=df['Churn'].value_counts().in
plt.show()
```

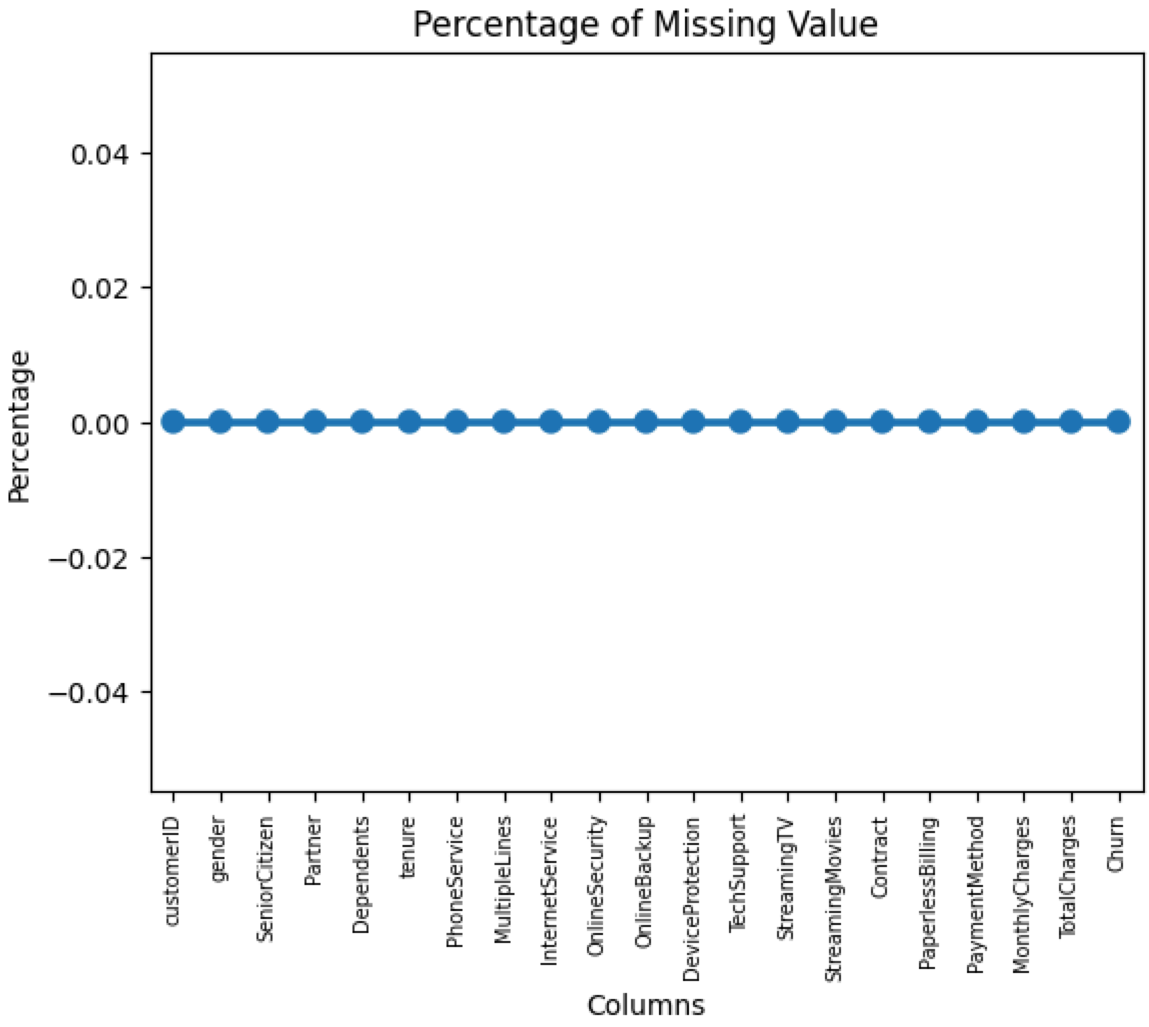


Insight

1. **Data Imbalance**: The data has a highly imbalanced ratio of 73:27 (No:Yes).
2. **Separate Analysis**: Analyze the data separately to gain more specific insights.
3. **Handling Imbalance in Model Building**: Use SMOTE (Synthetic Minority Over-sampling Technique) to address the data imbalance during model building.

Missing Value Visualization

In [10]:
```
missing=(((df.isnull().sum())*100)/df.shape[0]).reset_index()
sns.pointplot(data=missing,x='index',y=0)
plt.xticks(rotation=90,fontsize=7)
plt.title("Percentage of Missing Value")
plt.ylabel("Percentage")
plt.xlabel("Columns")
plt.show()
```



In [11]:
```
df.head(1)
```

Out[11]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Mul
0	7590-VHVEG	Female	0	Yes	No	1	No	

1 rows × 21 columns

In [12]:
```
df['TotalCharges']=pd.to_numeric(df['TotalCharges'],errors='coerce')
```

In [13]:
```
df['TotalCharges'].dtype
```

Out[13]:
```
dtype('float64')
```

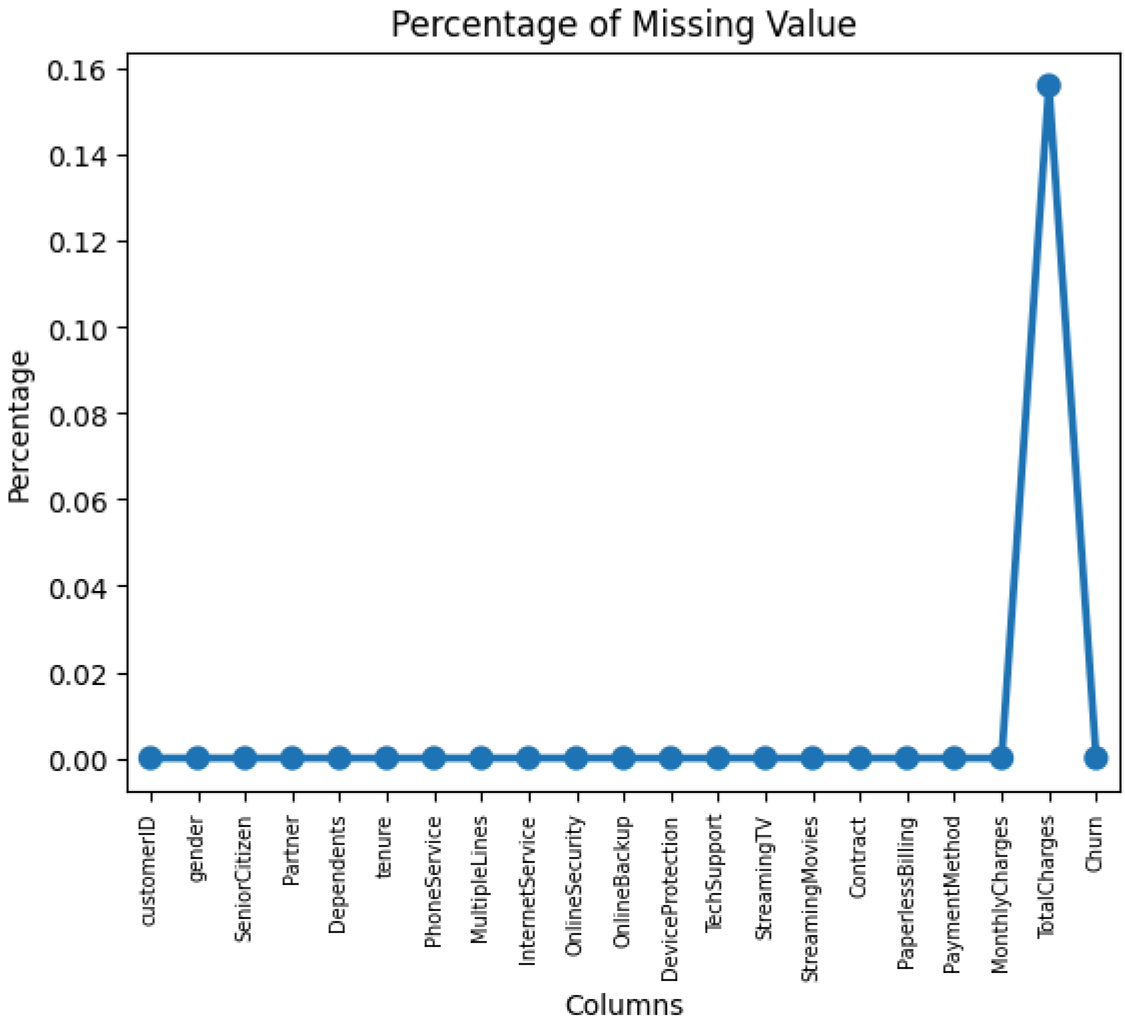
In [14]:
```
df.isnull().sum()
```

Out[14]:

```
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
```

In [15]:

```
missing=(((df.isnull().sum())*100)/df.shape[0]).reset_index()
sns.pointplot(data=missing,x='index',y=0)
plt.xticks(rotation=90,fontsize=7)
plt.title("Percentage of Missing Value")
plt.ylabel("Percentage")
plt.xlabel("Columns")
plt.show()
```



After converting the data to float, 11 NaN values were observed. These NaN values can be addressed by either removing rows containing NaN values, filling NaN values with a specific number like 0 or the mean of the column

In [16]:
```
df.loc[df['TotalCharges'].isnull()==True]
```

Out[16]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
488	4472-LVYGI	Female	0	Yes	Yes	0	No
753	3115-CZMZD	Male	0	No	Yes	0	Yes
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes
6754	2775-SEFEE	Male	0	No	Yes	0	Yes

11 rows × 21 columns

```
In [17]: df.dropna(how='any',inplace=True)
```

```
In [18]: df['TotalCharges'].isnull().sum()
```

```
Out[18]: 0
```

```
In [19]: df['tenure'].max()
```

```
Out[19]: 72
```

```
In [20]: labels = ['1-12', '13-24', '25-36', '37-48', '49-60', '61-72']

         df['tenure_group'] = pd.cut(df['tenure'], bins=range(1,80,12), labels=labels, ri
```

```
In [21]: df['tenure_group'].value_counts()
```

Out[21]:

```
1-12     2175
61-72    1407
13-24    1024
25-36     832
49-60     832
37-48     762
Name: tenure_group, dtype: int64
```

In [22]:

```
df.columns
```

Out[22]:

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

In [23]:

```
df.drop(columns=['customerID','tenure'],axis=1,inplace=True)
```

In [24]:

```
categorical_columns = df.select_dtypes(include=['object','category']).columns.tc
numerical_columns = df.select_dtypes(include=['int','float']).columns.tolist()
```

In [25]:

```
categorical_columns.append('SeniorCitizen')
numerical_columns.remove('SeniorCitizen')
```

In [26]:

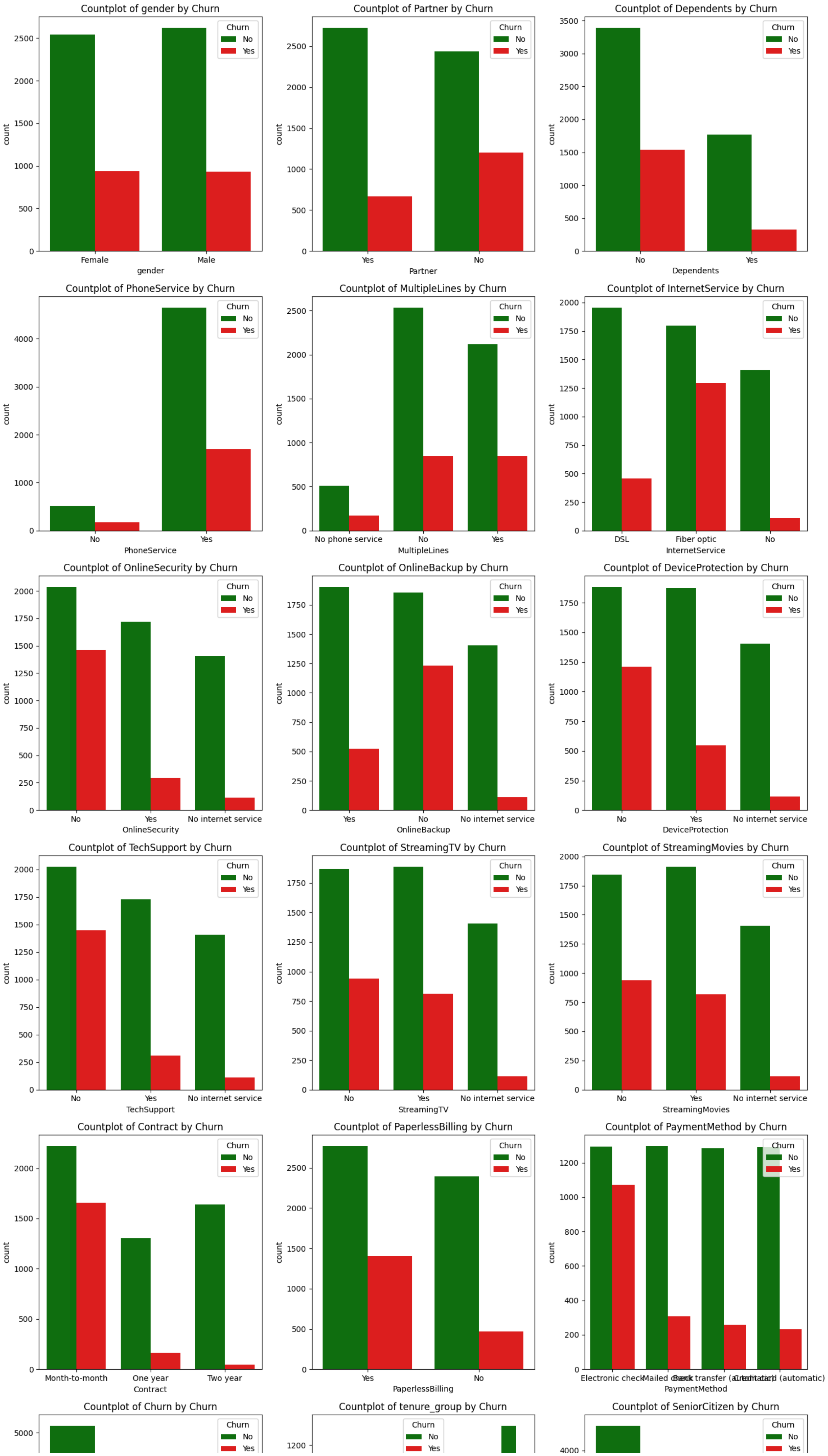
```
len(categorical_columns)
```

Out[26]:

```
18
```

In [27]:

```
cols = 3
rows = 6
fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(15, rows * 5))
axes = axes.flatten()
custom_palette = {'Yes': 'red', 'No': 'green'}
for ax, col in zip(axes, categorical_columns):
    sns.countplot(data=df, x=col, hue='Churn', ax=ax,palette=custom_palette)
    ax.set_title(f'Countplot of {col} by Churn')
plt.tight_layout()
plt.show()
```



Countplot of gender by Churn

Countplot of Partner by Churn

Countplot of Dependents by Churn

Countplot of PhoneService by Churn

Countplot of MultipleLines by Churn

Countplot of InternetService by Churn

Countplot of OnlineSecurity by Churn

Countplot of OnlineBackup by Churn

Countplot of DeviceProtection by Churn

Countplot of TechSupport by Churn

Countplot of StreamingTV by Churn

Countplot of StreamingMovies by Churn

Countplot of Contract by Churn

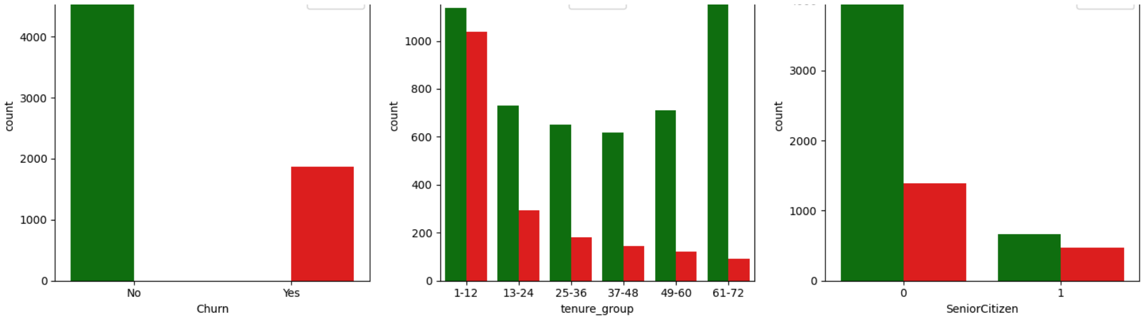
Countplot of PaperlessBilling by Churn

Countplot of PaymentMethod by Churn

Countplot of Churn by Churn

Countplot of tenure_group by Churn

Countplot of SeniorCitizen by Churn



Insights 🚀

1. **Gender:** 👫

   - The count of churn is similar between male and female customers.
   - A higher number of customers do not churn.

2. **Partner:** 💔

   - Customers without partners have a higher churn rate compared to those with partners.

3. **Dependents:** 👶

   - Customers with dependents are less likely to churn compared to those without dependents.

4. **Phone Service:** 📞

   - Customers with phone service have a higher churn rate compared to those without phone service.

5. **Multiple Lines:** 📱 📱

   - Customers with multiple lines have a higher churn rate than those without multiple lines or with no phone service.

6. **Internet Service:** 🌐

   - Fiber optic users have a higher churn rate compared to DSL users and those with no internet service.

7. **Online Security:** 🔒

   - Customers without online security have a higher churn rate compared to those with online security or no internet service.

8. **Online Backup:** 💾

   - Customers without online backup have a higher churn rate compared to those with online backup or no internet service.

9. **Device Protection:** 🛡️

   - Customers without device protection have a higher churn rate compared to those with device protection or no internet service.

10. **Tech Support:** 🧑‍💻

- Customers without tech support have a higher churn rate compared to those with tech support or no internet service.

11. **Streaming TV:** 📺

    - Customers who do not stream TV have a lower churn rate compared to those who do stream TV or have no internet service.

12. **Streaming Movies:** 🎬

    - Customers who do not stream movies have a lower churn rate compared to those who do stream movies or have no internet service.

13. **Contract:** 📄

    - Month-to-month contract customers have a significantly higher churn rate compared to those with one-year or two-year contracts.

14. **Paperless Billing:** 💻

    - Customers with paperless billing have a higher churn rate compared to those without paperless billing.

15. **Payment Method:** 💸

    - Electronic check customers have the highest churn rate, followed by mailed check, bank transfer (automatic), and credit card (automatic).

16. **Churn:** 🔄

    - The overall churn rate shows that most customers do not churn.

17. **Tenure Group:** 📅

    - Customers in the 1-12 month tenure group have the highest churn rate, which decreases with longer tenure groups.

18. **Senior Citizen:** 👴 👵

    - Senior citizens have a lower number of churns overall compared to non-senior citizens.
    - However, when considering the proportion of senior citizens and non-senior citizens, senior citizens have a higher likelihood of churning. This suggests that senior citizens churn at a higher rate relative to their representation in the customer base.

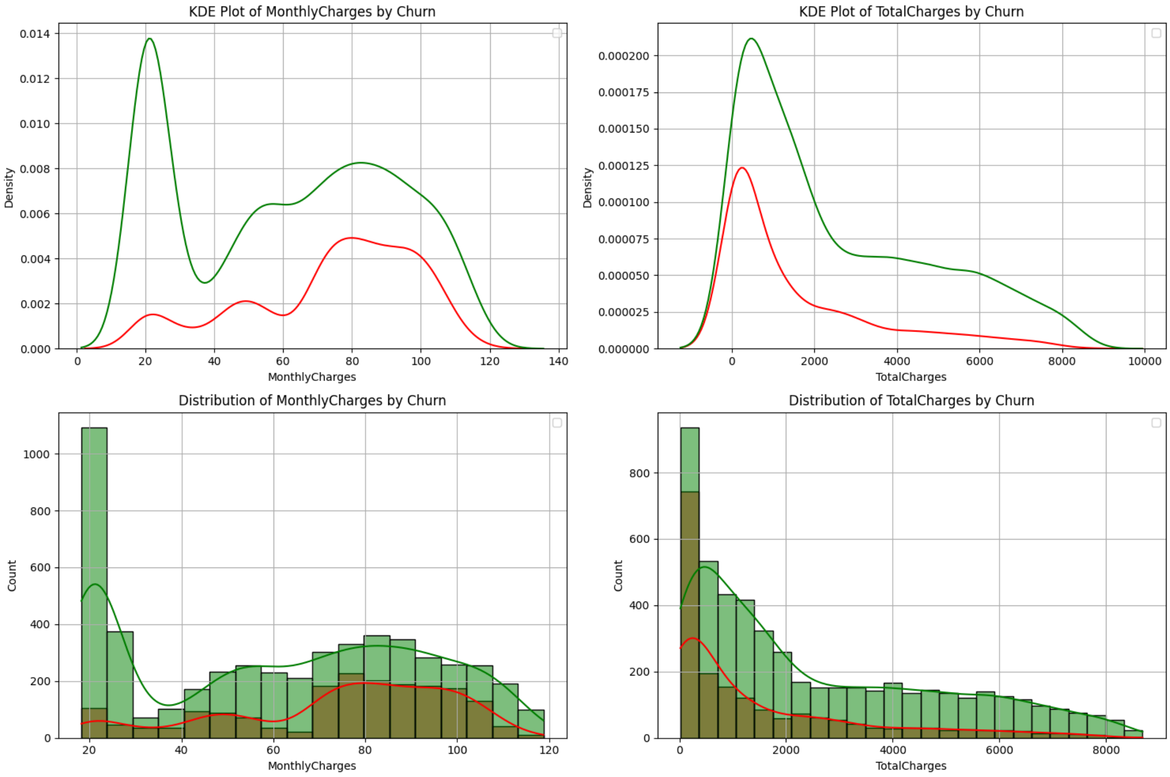
In [42]:

```
cols = 2
rows = 2
fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(15, rows * 5))
axes = axes.flatten()
custom_palette = {'Yes': 'red', 'No': 'green'}
for i, col in enumerate(numerical_columns):
    # Plot KDE plot
    sns.kdeplot(data=df, x=col, hue='Churn', ax=axes[i], palette=custom_palette)
    axes[i].set_title(f'KDE Plot of {col} by Churn')
    axes[i].legend()
    axes[i].grid(True)

    # Plot histogram with KDE overlay
    sns.histplot(data=df, x=col, hue='Churn', ax=axes[i+cols], palette=custom_pa
    axes[i+cols].set_title(f'Distribution of {col} by Churn')
    axes[i+cols].legend()
    axes[i+cols].grid(True)

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

```
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
```



Insights

Monthly Charges:

- **Non-churned Customers (Green):**
  - Most customers are paying around $20 per month.
  - There are also many customers paying between $60and80$ per month.
- **Churned Customers (Red):**
  - Customers are more likely to leave if they are paying around $80-100$ per month.

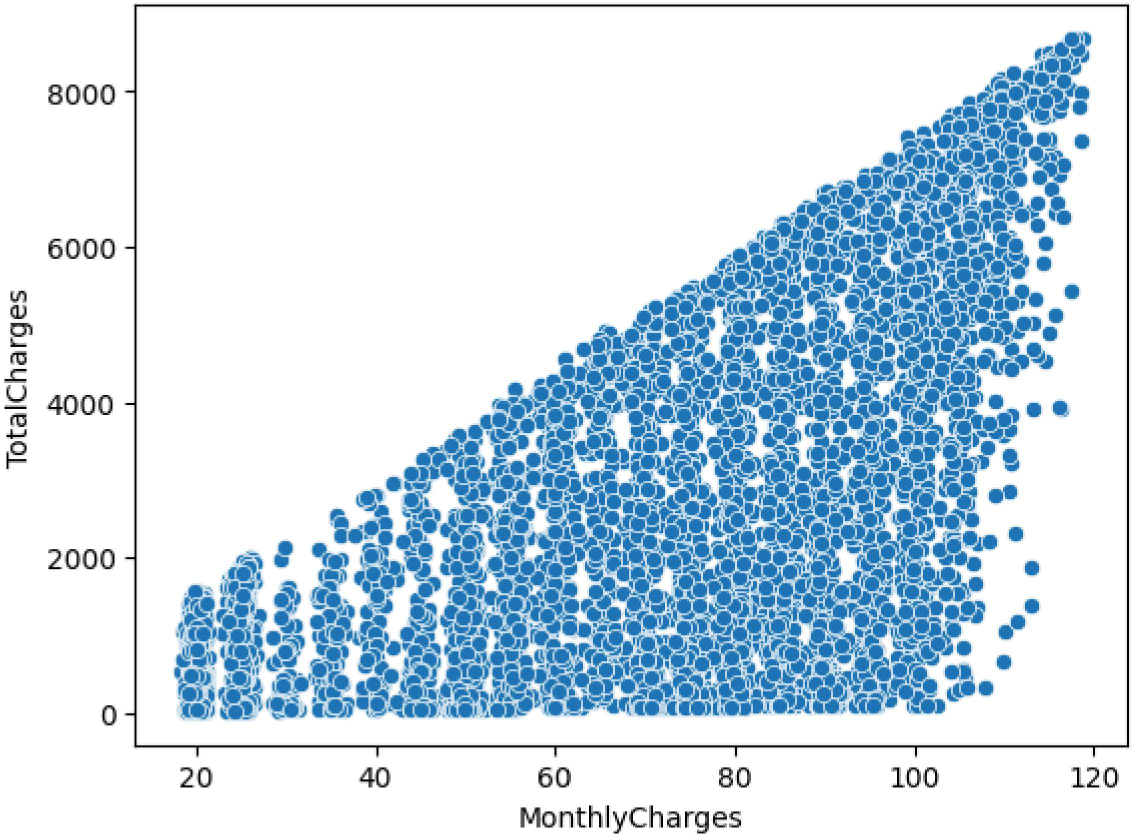
Total Charges:

- **Non-churned Customers (Green):**
  - Most customers have total charges less than $2,000.
- **Churned Customers (Red):**
  - Customers with lower total charges (less than $2,000) are more likely to leave.

`Note:` Both Monthly Charges and Total Charges are right-skewed, applying a log transformation can help normalize the data.

In [43]:
```
sns.scatterplot(data=df,x=numerical_columns[0],y=numerical_columns[1])
```

Out[43]:
```
<Axes: xlabel='MonthlyCharges', ylabel='TotalCharges'>
```



In [44]:
```
df[[numerical_columns[0],numerical_columns[1]]].corr()
```

Out[44]:

	MonthlyCharges	TotalCharges
MonthlyCharges	1.000000	0.651065
TotalCharges	0.651065	1.000000

Monthly Charges and Total Charges are highly correlated, it makes sense to select only one of these variables for your analysis to avoid multicollinearity, which can negatively impact the performance and interpretability of statistical models.

In [45]:
```
df.drop(columns=['TotalCharges'],axis=1,inplace=True)
```

In [70]:
```
def chi_square_test( x1, x2):
    contingency_table = pd.crosstab(df[x1], df[x2])
    chi_sq_result = chi2_contingency(contingency_table)
    p_value = chi_sq_result[1]
    correlation = "Correlated" if p_value < 0.05 else "Not Correlated"
    return p_value, correlation
```

In [72]:
```
for col1 in categorical_columns:
    for col2 in categorical_columns:
        if col1 != col2:
            r=chi_square_test(col1, col2)
            if r[1]=='Correlated':
                print(f"Chi-square Test for {col1} and {col2}:")
                print(r)
```

```
Chi-square Test for Partner and Dependents:
(0.0, 'Correlated')
Chi-square Test for Partner and MultipleLines:
(1.598728280420032e-32, 'Correlated')
Chi-square Test for Partner and OnlineSecurity:
(5.7832861351856704e-36, 'Correlated')
Chi-square Test for Partner and OnlineBackup:
(1.2979922776669001e-36, 'Correlated')
Chi-square Test for Partner and DeviceProtection:
(9.364677583389403e-43, 'Correlated')
Chi-square Test for Partner and TechSupport:
(1.4901687562962913e-25, 'Correlated')
Chi-square Test for Partner and StreamingTV:
(2.9557190881420945e-29, 'Correlated')
Chi-square Test for Partner and StreamingMovies:
(1.6980728723707905e-26, 'Correlated')
Chi-square Test for Partner and Contract:
(6.569797961945938e-134, 'Correlated')
Chi-square Test for Partner and PaymentMethod:
(1.098559778101642e-40, 'Correlated')
Chi-square Test for Partner and Churn:
(3.97379757451591e-36, 'Correlated')
Chi-square Test for Partner and tenure_group:
(1.161283727493916e-210, 'Correlated')
Chi-square Test for Dependents and Partner:
(0.0, 'Correlated')
Chi-square Test for Dependents and InternetService:
(2.4835965561301876e-49, 'Correlated')
Chi-square Test for Dependents and OnlineSecurity:
(1.3567889555601824e-57, 'Correlated')
Chi-square Test for Dependents and OnlineBackup:
(2.202009175635511e-40, 'Correlated')
Chi-square Test for Dependents and DeviceProtection:
(7.307390550853895e-38, 'Correlated')
Chi-square Test for Dependents and TechSupport:
(1.3230957629417935e-50, 'Correlated')
Chi-square Test for Dependents and StreamingTV:
(4.6713035166058e-33, 'Correlated')
Chi-square Test for Dependents and StreamingMovies:
(1.1283994905904793e-30, 'Correlated')
Chi-square Test for Dependents and Contract:
(6.366704051182578e-90, 'Correlated')
Chi-square Test for Dependents and PaperlessBilling:
(3.300642269458549e-20, 'Correlated')
Chi-square Test for Dependents and PaymentMethod:
(8.374235491356323e-34, 'Correlated')
Chi-square Test for Dependents and Churn:
(2.0196592017051303e-42, 'Correlated')
Chi-square Test for Dependents and tenure_group:
(2.1235135394560207e-39, 'Correlated')
Chi-square Test for Dependents and SeniorCitizen:
(1.7066137240276333e-69, 'Correlated')
Chi-square Test for PhoneService and MultipleLines:
(0.0, 'Correlated')
Chi-square Test for PhoneService and InternetService:
(0.0, 'Correlated')
Chi-square Test for PhoneService and OnlineSecurity:
(7.561203031788675e-48, 'Correlated')
Chi-square Test for PhoneService and OnlineBackup:
(4.02340259235113e-46, 'Correlated')
```

```
Chi-square Test for PhoneService and DeviceProtection:
(7.610860567010651e-46, 'Correlated')
Chi-square Test for PhoneService and TechSupport:
(3.213265149028798e-48, 'Correlated')
Chi-square Test for PhoneService and StreamingTV:
(2.1205214643698914e-50, 'Correlated')
Chi-square Test for PhoneService and StreamingMovies:
(1.5471806455336363e-48, 'Correlated')
Chi-square Test for MultipleLines and Partner:
(1.598728280420032e-32, 'Correlated')
Chi-square Test for MultipleLines and PhoneService:
(0.0, 'Correlated')
Chi-square Test for MultipleLines and InternetService:
(0.0, 'Correlated')
Chi-square Test for MultipleLines and OnlineSecurity:
(6.117090986902672e-159, 'Correlated')
Chi-square Test for MultipleLines and OnlineBackup:
(1.5951421870048882e-181, 'Correlated')
Chi-square Test for MultipleLines and DeviceProtection:
(1.962575031913262e-183, 'Correlated')
Chi-square Test for MultipleLines and TechSupport:
(1.2937199160869143e-159, 'Correlated')
Chi-square Test for MultipleLines and StreamingTV:
(1.3087009474518547e-207, 'Correlated')
Chi-square Test for MultipleLines and StreamingMovies:
(2.241457364611034e-208, 'Correlated')
Chi-square Test for MultipleLines and Contract:
(7.095855963111061e-19, 'Correlated')
Chi-square Test for MultipleLines and PaperlessBilling:
(4.542320360434221e-43, 'Correlated')
Chi-square Test for MultipleLines and PaymentMethod:
(1.2165642114365908e-81, 'Correlated')
Chi-square Test for MultipleLines and Churn:
(0.0035679273999811405, 'Correlated')
Chi-square Test for MultipleLines and tenure_group:
(2.1980480245468394e-169, 'Correlated')
Chi-square Test for MultipleLines and SeniorCitizen:
(1.3319917312846447e-33, 'Correlated')
Chi-square Test for InternetService and Dependents:
(2.4835965561301876e-49, 'Correlated')
Chi-square Test for InternetService and PhoneService:
(0.0, 'Correlated')
Chi-square Test for InternetService and MultipleLines:
(0.0, 'Correlated')
Chi-square Test for InternetService and OnlineSecurity:
(0.0, 'Correlated')
Chi-square Test for InternetService and OnlineBackup:
(0.0, 'Correlated')
Chi-square Test for InternetService and DeviceProtection:
(0.0, 'Correlated')
Chi-square Test for InternetService and TechSupport:
(0.0, 'Correlated')
Chi-square Test for InternetService and StreamingTV:
(0.0, 'Correlated')
Chi-square Test for InternetService and StreamingMovies:
(0.0, 'Correlated')
Chi-square Test for InternetService and Contract:
(6.681341589232126e-128, 'Correlated')
Chi-square Test for InternetService and PaperlessBilling:
(7.221948661149777e-219, 'Correlated')
```

```
Chi-square Test for InternetService and PaymentMethod:
(4.262322425661328e-293, 'Correlated')
Chi-square Test for InternetService and Churn:
(5.831198962237274e-159, 'Correlated')
Chi-square Test for InternetService and SeniorCitizen:
(6.801939980637394e-108, 'Correlated')
Chi-square Test for OnlineSecurity and Partner:
(5.7832861351855875e-36, 'Correlated')
Chi-square Test for OnlineSecurity and Dependents:
(1.3567889555601433e-57, 'Correlated')
Chi-square Test for OnlineSecurity and PhoneService:
(7.561203031788785e-48, 'Correlated')
Chi-square Test for OnlineSecurity and MultipleLines:
(6.117090986903019e-159, 'Correlated')
Chi-square Test for OnlineSecurity and InternetService:
(0.0, 'Correlated')
Chi-square Test for OnlineSecurity and OnlineBackup:
(0.0, 'Correlated')
Chi-square Test for OnlineSecurity and DeviceProtection:
(0.0, 'Correlated')
Chi-square Test for OnlineSecurity and TechSupport:
(0.0, 'Correlated')
Chi-square Test for OnlineSecurity and StreamingTV:
(0.0, 'Correlated')
Chi-square Test for OnlineSecurity and StreamingMovies:
(0.0, 'Correlated')
Chi-square Test for OnlineSecurity and Contract:
(2.894613836236555e-274, 'Correlated')
Chi-square Test for OnlineSecurity and PaperlessBilling:
(9.958860030591835e-179, 'Correlated')
Chi-square Test for OnlineSecurity and PaymentMethod:
(1.0318305397647769e-277, 'Correlated')
Chi-square Test for OnlineSecurity and Churn:
(1.4006867477839222e-184, 'Correlated')
Chi-square Test for OnlineSecurity and tenure_group:
(2.989946972296736e-160, 'Correlated')
Chi-square Test for OnlineSecurity and SeniorCitizen:
(1.5584109549434674e-68, 'Correlated')
Chi-square Test for OnlineBackup and Partner:
(1.2979922776669001e-36, 'Correlated')
Chi-square Test for OnlineBackup and Dependents:
(2.202009175635511e-40, 'Correlated')
Chi-square Test for OnlineBackup and PhoneService:
(4.02340259235113e-46, 'Correlated')
Chi-square Test for OnlineBackup and MultipleLines:
(1.5951421870048882e-181, 'Correlated')
Chi-square Test for OnlineBackup and InternetService:
(0.0, 'Correlated')
Chi-square Test for OnlineBackup and OnlineSecurity:
(0.0, 'Correlated')
Chi-square Test for OnlineBackup and DeviceProtection:
(0.0, 'Correlated')
Chi-square Test for OnlineBackup and TechSupport:
(0.0, 'Correlated')
Chi-square Test for OnlineBackup and StreamingTV:
(0.0, 'Correlated')
Chi-square Test for OnlineBackup and StreamingMovies:
(0.0, 'Correlated')
Chi-square Test for OnlineBackup and Contract:
(2.366319576785634e-201, 'Correlated')
```

```
Chi-square Test for OnlineBackup and PaperlessBilling:
(1.0348274720546278e-157, 'Correlated')
Chi-square Test for OnlineBackup and PaymentMethod:
(5.722022463618146e-238, 'Correlated')
Chi-square Test for OnlineBackup and Churn:
(7.776099238804965e-131, 'Correlated')
Chi-square Test for OnlineBackup and tenure_group:
(3.119591702417532e-204, 'Correlated')
Chi-square Test for OnlineBackup and SeniorCitizen:
(1.3073984759877554e-51, 'Correlated')
Chi-square Test for DeviceProtection and Partner:
(9.364677583389403e-43, 'Correlated')
Chi-square Test for DeviceProtection and Dependents:
(7.30739055085379e-38, 'Correlated')
Chi-square Test for DeviceProtection and PhoneService:
(7.610860567010651e-46, 'Correlated')
Chi-square Test for DeviceProtection and MultipleLines:
(1.9625750319133736e-183, 'Correlated')
Chi-square Test for DeviceProtection and InternetService:
(0.0, 'Correlated')
Chi-square Test for DeviceProtection and OnlineSecurity:
(0.0, 'Correlated')
Chi-square Test for DeviceProtection and OnlineBackup:
(0.0, 'Correlated')
Chi-square Test for DeviceProtection and TechSupport:
(0.0, 'Correlated')
Chi-square Test for DeviceProtection and StreamingTV:
(0.0, 'Correlated')
Chi-square Test for DeviceProtection and StreamingMovies:
(0.0, 'Correlated')
Chi-square Test for DeviceProtection and Contract:
(1.8806082103300366e-266, 'Correlated')
Chi-square Test for DeviceProtection and PaperlessBilling:
(3.110307853389781e-158, 'Correlated')
Chi-square Test for DeviceProtection and PaymentMethod:
(8.296212276266195e-246, 'Correlated')
Chi-square Test for DeviceProtection and Churn:
(1.9593887862403176e-121, 'Correlated')
Chi-square Test for DeviceProtection and tenure_group:
(3.0822081235900515e-202, 'Correlated')
Chi-square Test for DeviceProtection and SeniorCitizen:
(9.062608854854384e-52, 'Correlated')
Chi-square Test for TechSupport and Partner:
(1.4901687562962913e-25, 'Correlated')
Chi-square Test for TechSupport and Dependents:
(1.3230957629417747e-50, 'Correlated')
Chi-square Test for TechSupport and PhoneService:
(3.213265149028752e-48, 'Correlated')
Chi-square Test for TechSupport and MultipleLines:
(1.2937199160869143e-159, 'Correlated')
Chi-square Test for TechSupport and InternetService:
(0.0, 'Correlated')
Chi-square Test for TechSupport and OnlineSecurity:
(0.0, 'Correlated')
Chi-square Test for TechSupport and OnlineBackup:
(0.0, 'Correlated')
Chi-square Test for TechSupport and DeviceProtection:
(0.0, 'Correlated')
Chi-square Test for TechSupport and StreamingTV:
(0.0, 'Correlated')
```

```
Chi-square Test for TechSupport and StreamingMovies:
(0.0, 'Correlated')
Chi-square Test for TechSupport and Contract:
(0.0, 'Correlated')
Chi-square Test for TechSupport and PaperlessBilling:
(4.026697113951921e-166, 'Correlated')
Chi-square Test for TechSupport and PaymentMethod:
(2.6633723243656977e-281, 'Correlated')
Chi-square Test for TechSupport and Churn:
(7.407807748843711e-180, 'Correlated')
Chi-square Test for TechSupport and tenure_group:
(4.3544599083483415e-160, 'Correlated')
Chi-square Test for TechSupport and SeniorCitizen:
(3.810958434023567e-77, 'Correlated')
Chi-square Test for StreamingTV and Partner:
(2.955719088142052e-29, 'Correlated')
Chi-square Test for StreamingTV and Dependents:
(4.6713035166058e-33, 'Correlated')
Chi-square Test for StreamingTV and PhoneService:
(2.1205214643699213e-50, 'Correlated')
Chi-square Test for StreamingTV and MultipleLines:
(1.3087009474518547e-207, 'Correlated')
Chi-square Test for StreamingTV and InternetService:
(0.0, 'Correlated')
Chi-square Test for StreamingTV and OnlineSecurity:
(0.0, 'Correlated')
Chi-square Test for StreamingTV and OnlineBackup:
(0.0, 'Correlated')
Chi-square Test for StreamingTV and DeviceProtection:
(0.0, 'Correlated')
Chi-square Test for StreamingTV and TechSupport:
(0.0, 'Correlated')
Chi-square Test for StreamingTV and StreamingMovies:
(0.0, 'Correlated')
Chi-square Test for StreamingTV and Contract:
(2.915550413231528e-167, 'Correlated')
Chi-square Test for StreamingTV and PaperlessBilling:
(5.1517953834026255e-173, 'Correlated')
Chi-square Test for StreamingTV and PaymentMethod:
(2.8432240583151905e-223, 'Correlated')
Chi-square Test for StreamingTV and Churn:
(1.324641113169159e-81, 'Correlated')
Chi-square Test for StreamingTV and tenure_group:
(1.8017681533339575e-118, 'Correlated')
Chi-square Test for StreamingTV and SeniorCitizen:
(3.185362894070116e-53, 'Correlated')
Chi-square Test for StreamingMovies and Partner:
(1.6980728723707905e-26, 'Correlated')
Chi-square Test for StreamingMovies and Dependents:
(1.1283994905904793e-30, 'Correlated')
Chi-square Test for StreamingMovies and PhoneService:
(1.5471806455336363e-48, 'Correlated')
Chi-square Test for StreamingMovies and MultipleLines:
(2.241457364611034e-208, 'Correlated')
Chi-square Test for StreamingMovies and InternetService:
(0.0, 'Correlated')
Chi-square Test for StreamingMovies and OnlineSecurity:
(0.0, 'Correlated')
Chi-square Test for StreamingMovies and OnlineBackup:
(0.0, 'Correlated')
```

```
Chi-square Test for StreamingMovies and DeviceProtection:
(0.0, 'Correlated')
Chi-square Test for StreamingMovies and TechSupport:
(0.0, 'Correlated')
Chi-square Test for StreamingMovies and StreamingTV:
(0.0, 'Correlated')
Chi-square Test for StreamingMovies and Contract:
(2.8718346270688147e-172, 'Correlated')
Chi-square Test for StreamingMovies and PaperlessBilling:
(9.112293536780645e-169, 'Correlated')
Chi-square Test for StreamingMovies and PaymentMethod:
(7.853065206950765e-225, 'Correlated')
Chi-square Test for StreamingMovies and Churn:
(5.353560421401324e-82, 'Correlated')
Chi-square Test for StreamingMovies and tenure_group:
(1.3209906417803019e-123, 'Correlated')
Chi-square Test for StreamingMovies and SeniorCitizen:
(4.3023641295415215e-55, 'Correlated')
Chi-square Test for Contract and Partner:
(6.569797961945938e-134, 'Correlated')
Chi-square Test for Contract and Dependents:
(6.366704051182578e-90, 'Correlated')
Chi-square Test for Contract and MultipleLines:
(7.095855963111061e-19, 'Correlated')
Chi-square Test for Contract and InternetService:
(6.681341589232126e-128, 'Correlated')
Chi-square Test for Contract and OnlineSecurity:
(2.894613836236884e-274, 'Correlated')
Chi-square Test for Contract and OnlineBackup:
(2.366319576785634e-201, 'Correlated')
Chi-square Test for Contract and DeviceProtection:
(1.8806082103298224e-266, 'Correlated')
Chi-square Test for Contract and TechSupport:
(0.0, 'Correlated')
Chi-square Test for Contract and StreamingTV:
(2.915550413231528e-167, 'Correlated')
Chi-square Test for Contract and StreamingMovies:
(2.8718346270688147e-172, 'Correlated')
Chi-square Test for Contract and PaperlessBilling:
(2.7450456699387723e-48, 'Correlated')
Chi-square Test for Contract and PaymentMethod:
(6.908556640756664e-213, 'Correlated')
Chi-square Test for Contract and Churn:
(7.326182186265472e-257, 'Correlated')
Chi-square Test for Contract and tenure_group:
(0.0, 'Correlated')
Chi-square Test for Contract and SeniorCitizen:
(5.048394653966384e-32, 'Correlated')
Chi-square Test for PaperlessBilling and Dependents:
(3.300642269458549e-20, 'Correlated')
Chi-square Test for PaperlessBilling and MultipleLines:
(4.542320360434221e-43, 'Correlated')
Chi-square Test for PaperlessBilling and InternetService:
(7.221948661149777e-219, 'Correlated')
Chi-square Test for PaperlessBilling and OnlineSecurity:
(9.958860030591835e-179, 'Correlated')
Chi-square Test for PaperlessBilling and OnlineBackup:
(1.0348274720546278e-157, 'Correlated')
Chi-square Test for PaperlessBilling and DeviceProtection:
(3.110307853389604e-158, 'Correlated')
```

```
Chi-square Test for PaperlessBilling and TechSupport:
(4.0266971139516925e-166, 'Correlated')
Chi-square Test for PaperlessBilling and StreamingTV:
(5.1517953834026255e-173, 'Correlated')
Chi-square Test for PaperlessBilling and StreamingMovies:
(9.112293536780645e-169, 'Correlated')
Chi-square Test for PaperlessBilling and Contract:
(2.7450456699387723e-48, 'Correlated')
Chi-square Test for PaperlessBilling and PaymentMethod:
(3.6617526905214006e-93, 'Correlated')
Chi-square Test for PaperlessBilling and Churn:
(8.236203353962564e-58, 'Correlated')
Chi-square Test for PaperlessBilling and SeniorCitizen:
(4.860566886559332e-39, 'Correlated')
Chi-square Test for PaymentMethod and Partner:
(1.098559778101642e-40, 'Correlated')
Chi-square Test for PaymentMethod and Dependents:
(8.374235491356448e-34, 'Correlated')
Chi-square Test for PaymentMethod and MultipleLines:
(1.216564211436556e-81, 'Correlated')
Chi-square Test for PaymentMethod and InternetService:
(4.262322425661328e-293, 'Correlated')
Chi-square Test for PaymentMethod and OnlineSecurity:
(1.0318305397647769e-277, 'Correlated')
Chi-square Test for PaymentMethod and OnlineBackup:
(5.722022463618146e-238, 'Correlated')
Chi-square Test for PaymentMethod and DeviceProtection:
(8.296212276265248e-246, 'Correlated')
Chi-square Test for PaymentMethod and TechSupport:
(2.6633723243656977e-281, 'Correlated')
Chi-square Test for PaymentMethod and StreamingTV:
(2.8432240583151905e-223, 'Correlated')
Chi-square Test for PaymentMethod and StreamingMovies:
(7.85306520694987e-225, 'Correlated')
Chi-square Test for PaymentMethod and Contract:
(6.908556640756664e-213, 'Correlated')
Chi-square Test for PaymentMethod and PaperlessBilling:
(3.6617526905214006e-93, 'Correlated')
Chi-square Test for PaymentMethod and Churn:
(1.4263098511063342e-139, 'Correlated')
Chi-square Test for PaymentMethod and tenure_group:
(9.815053054315011e-224, 'Correlated')
Chi-square Test for PaymentMethod and SeniorCitizen:
(5.77509807076409e-58, 'Correlated')
Chi-square Test for Churn and Partner:
(3.97379757451591e-36, 'Correlated')
Chi-square Test for Churn and Dependents:
(2.0196592017051303e-42, 'Correlated')
Chi-square Test for Churn and MultipleLines:
(0.0035679273999811405, 'Correlated')
Chi-square Test for Churn and InternetService:
(5.831198962236941e-159, 'Correlated')
Chi-square Test for Churn and OnlineSecurity:
(1.4006867477839222e-184, 'Correlated')
Chi-square Test for Churn and OnlineBackup:
(7.776099238804965e-131, 'Correlated')
Chi-square Test for Churn and DeviceProtection:
(1.9593887862403176e-121, 'Correlated')
Chi-square Test for Churn and TechSupport:
(7.407807748843288e-180, 'Correlated')
```

```
Chi-square Test for Churn and StreamingTV:
(1.324641131691968e-81, 'Correlated')
Chi-square Test for Churn and StreamingMovies:
(5.353560421401323e-82, 'Correlated')
Chi-square Test for Churn and Contract:
(7.326182186264635e-257, 'Correlated')
Chi-square Test for Churn and PaperlessBilling:
(8.236203353962564e-58, 'Correlated')
Chi-square Test for Churn and PaymentMethod:
(1.4263098511062525e-139, 'Correlated')
Chi-square Test for Churn and tenure_group:
(2.3583950639333285e-188, 'Correlated')
Chi-square Test for Churn and SeniorCitizen:
(2.4792557203954705e-36, 'Correlated')
Chi-square Test for tenure_group and Partner:
(1.1612837274939824e-210, 'Correlated')
Chi-square Test for tenure_group and Dependents:
(2.1235135394559894e-39, 'Correlated')
Chi-square Test for tenure_group and MultipleLines:
(2.1980480245468394e-169, 'Correlated')
Chi-square Test for tenure_group and OnlineSecurity:
(2.989946972296736e-160, 'Correlated')
Chi-square Test for tenure_group and OnlineBackup:
(3.119591702417532e-204, 'Correlated')
Chi-square Test for tenure_group and DeviceProtection:
(3.0822081235900515e-202, 'Correlated')
Chi-square Test for tenure_group and TechSupport:
(4.3544599083483415e-160, 'Correlated')
Chi-square Test for tenure_group and StreamingTV:
(1.8017681533339575e-118, 'Correlated')
Chi-square Test for tenure_group and StreamingMovies:
(1.3209906417803019e-123, 'Correlated')
Chi-square Test for tenure_group and Contract:
(0.0, 'Correlated')
Chi-square Test for tenure_group and PaymentMethod:
(9.815053054315011e-224, 'Correlated')
Chi-square Test for tenure_group and Churn:
(2.358395063933462e-188, 'Correlated')
Chi-square Test for SeniorCitizen and Dependents:
(1.7066137240276333e-69, 'Correlated')
Chi-square Test for SeniorCitizen and MultipleLines:
(1.3319917312846447e-33, 'Correlated')
Chi-square Test for SeniorCitizen and InternetService:
(6.8019399806372e-108, 'Correlated')
Chi-square Test for SeniorCitizen and OnlineSecurity:
(1.5584109549434674e-68, 'Correlated')
Chi-square Test for SeniorCitizen and OnlineBackup:
(1.3073984759877554e-51, 'Correlated')
Chi-square Test for SeniorCitizen and DeviceProtection:
(9.062608854854384e-52, 'Correlated')
Chi-square Test for SeniorCitizen and TechSupport:
(3.810958434023567e-77, 'Correlated')
Chi-square Test for SeniorCitizen and StreamingTV:
(3.1853628940701154e-53, 'Correlated')
Chi-square Test for SeniorCitizen and StreamingMovies:
(4.3023641295415215e-55, 'Correlated')
Chi-square Test for SeniorCitizen and Contract:
(5.048394653966384e-32, 'Correlated')
Chi-square Test for SeniorCitizen and PaperlessBilling:
(4.860566886559332e-39, 'Correlated')
```

```
Chi-square Test for SeniorCitizen and PaymentMethod:
(5.77509807076409e-58, 'Correlated')
Chi-square Test for SeniorCitizen and Churn:
(2.4792557203954705e-36, 'Correlated')
```

1. **Strong Associations (Correlated):**

    - Many pairs of variables show very low p-values (close to zero), indicating strong evidence against the null hypothesis of independence.
    - Examples include:
        - Partner and Dependents
        - Partner and MultipleLines
        - Partner and OnlineSecurity
        - Partner and Contract
        - PaymentMethod and InternetService
        - Churn and Contract
        - SeniorCitizen and InternetService
2. **Moderate Associations (Potentially Correlated):**

    - Some pairs have higher but still significant p-values, suggesting moderate evidence against the null hypothesis.
    - Examples include:
        - MultipleLines and Churn
        - Contract and SeniorCitizen
3. **No Significant Associations:**

    - Variables like PhoneService and MultipleLines show a p-value of 0.0, indicating a significant association, but this could also mean they are directly related in terms of service provision.

Based on the findings from the chi-square tests, here are the columns you should consider dropping for selecting the best features for model building, along with brief reasons:

1. **MultipleLines**

    - Strong association with PhoneService(p-value close to zero), indicating redundancy in information.
2. **OnlineSecurity**

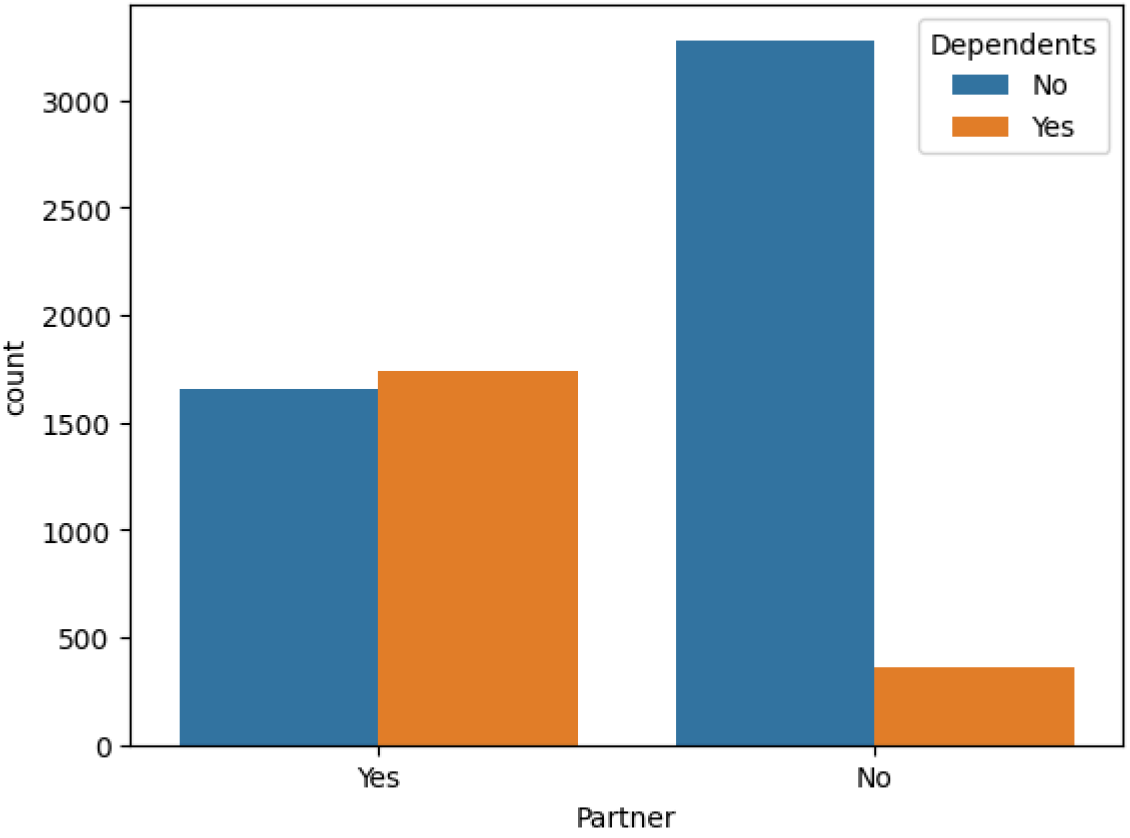
    - Strong association with Partner (low p-value), suggesting correlated information.
3. **InternetService**

    - Strong association with PaymentMethod (low p-value), indicating overlap in predictive power.
4. **Dependents**

    - Strongly associated with Partner, indicating redundancy in predictive information.

```
In [93]: sns.countplot(x=df['Partner'],hue=df['Dependents'])
```

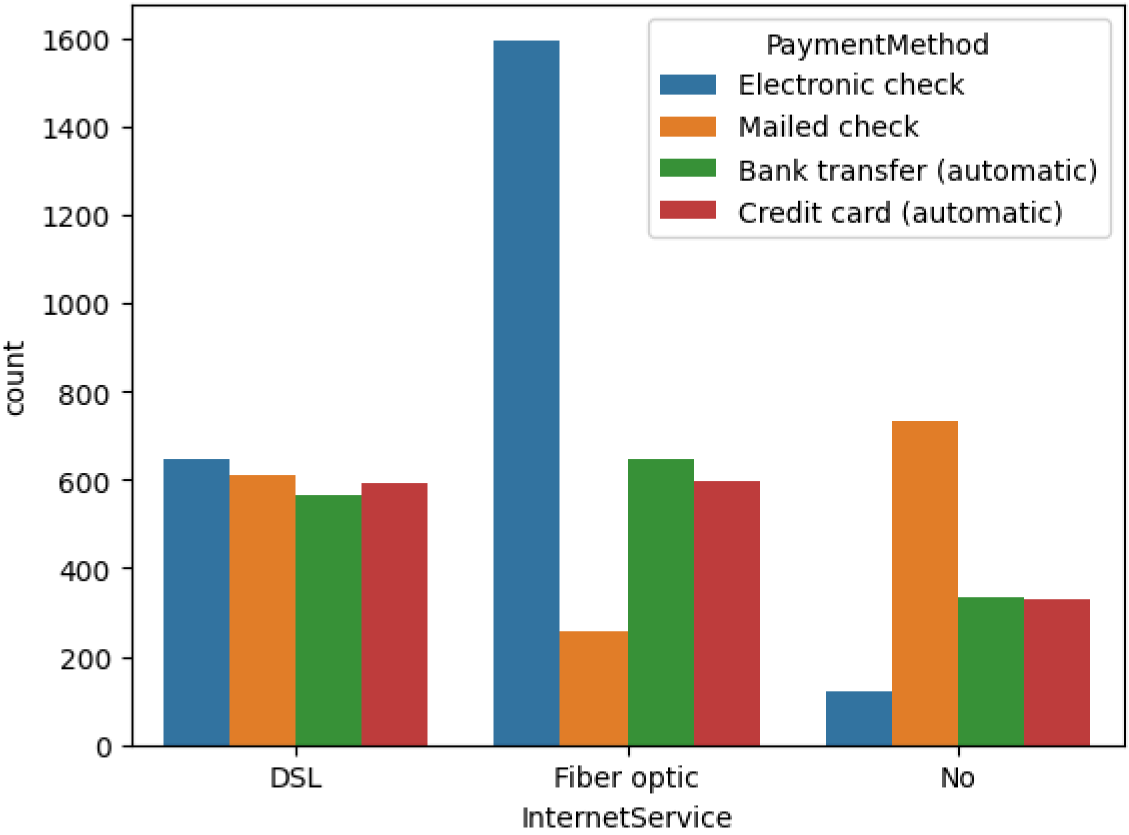
```
Out[93]: <Axes: xlabel='Partner', ylabel='count'>
```



If a person is married, they are dependent on someone else.

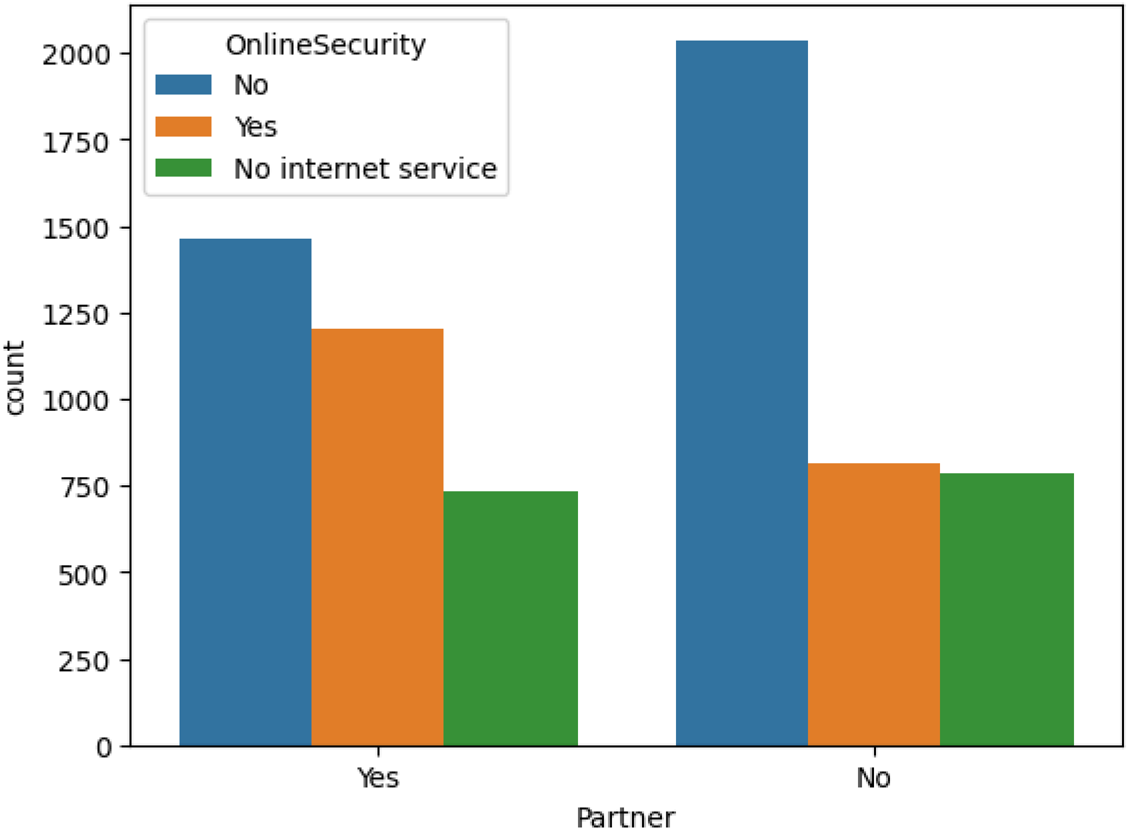
```
In [95]: sns.countplot(x=df['InternetService'],hue=df['PaymentMethod'])
```

```
Out[95]: <Axes: xlabel='InternetService', ylabel='count'>
```



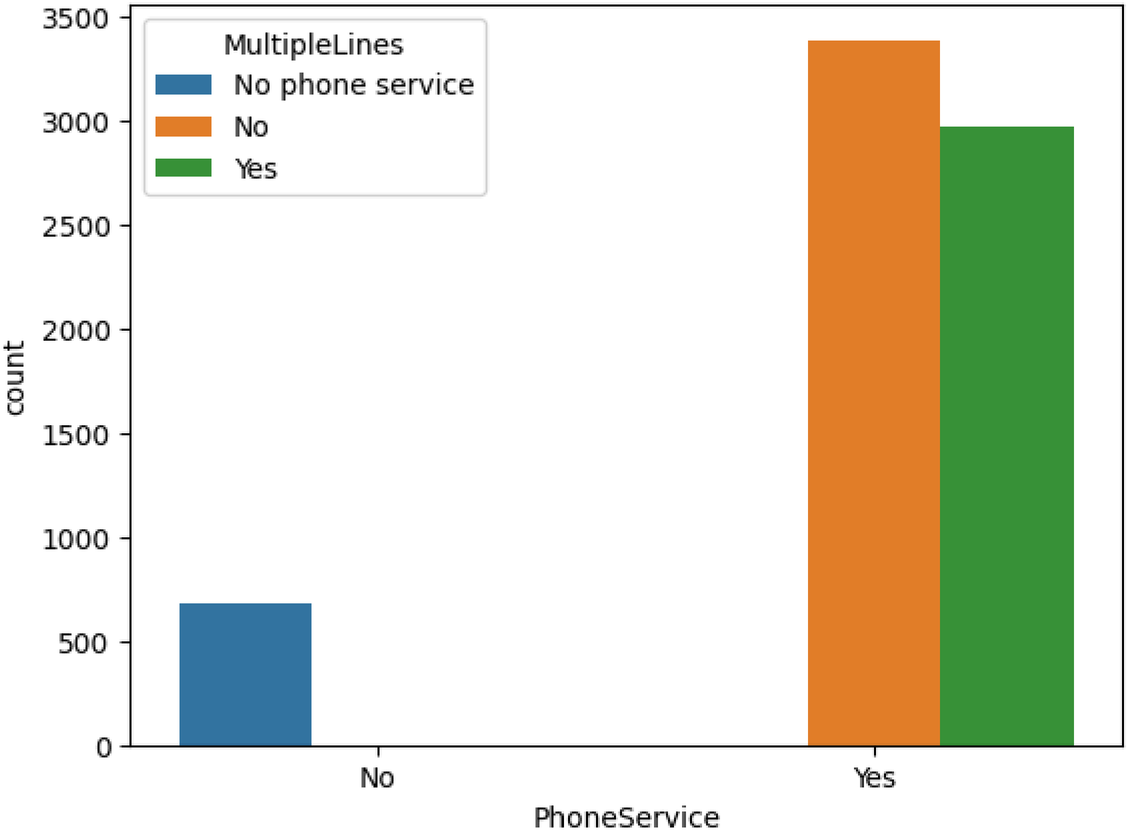
```
In [97]: sns.countplot(x=df['Partner'],hue=df['OnlineSecurity'])
```

```
Out[97]: <Axes: xlabel='Partner', ylabel='count'>
```



```
In [98]: sns.countplot(x=df['PhoneService'],hue=df['MultipleLines'])
```

```
Out[98]: <Axes: xlabel='PhoneService', ylabel='count'>
```



Mostly costumer are use Phone service so. i remove Multiple line

```
In [101…  df.columns.__len__()
```

```
Out[101]: 19
```

```
In [102…  df.drop(columns=['MultipleLines','Dependents','InternetService','OnlineSecurity'
```

```
In [103…  df.columns.__len__()
```

```
Out[103]: 15
```

```
In [129…  df['MonthlyCharges'] = np.log(df['MonthlyCharges'] + 1)
```

```
In [151…  X=df.drop('Churn' ,axis=1)
          y=df['Churn'].map({'No': 0, 'Yes': 1})
```

```
In [152…  categorical_columns = X.select_dtypes(include=['object','category']).columns.tol
          numerical_columns = X.select_dtypes(include=['int','float']).columns.tolist()
```

```
In [153…  preprocessor = ColumnTransformer(transformers=[
              ('cat', OneHotEncoder(handle_unknown='ignore',sparse=False,drop='first'), ca
          ])
```

```
In [154…  smote = SMOTE(random_state=42)
```

```
In [155…  models = [
              ("Logistic Regression", LogisticRegression()),
              ("Support Vector Machines (SVM)", SVC()),
              ("Random Forest", RandomForestClassifier()),
              ("Gradient Boosting Machines (XGBoost)", XGBClassifier()),
              ("K-Nearest Neighbors (KNN)", KNeighborsClassifier()),
              ("Decision Trees", DecisionTreeClassifier()),
              ("Random Forest", RandomForestClassifier()),
              ("Extra Trees", ExtraTreesClassifier()),
              ("Ensemble Methods (AdaBoost)", AdaBoostClassifier()),
              (
                  "Ensemble Methods (Voting Classifier)",
                  VotingClassifier(
                      estimators=[
                          ("lr", LogisticRegression()),
                          ("svm", SVC()),
                          ("rf", RandomForestClassifier()),
                      ]
                  ),
              ),
              (
                  "Ensemble Methods (Bagging)",
                  BaggingClassifier(base_estimator=DecisionTreeClassifier()),
              ),
              ("Ensemble Methods (Gradient Boosting)", GradientBoostingClassifier()),
          ]
```

In [156…

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [157…

```
def evaluate_classification_model(model_name, preprocessor, smote, model, X_trai
    # Fit the preprocessor to the training data
    X_train_preprocessed = preprocessor.fit_transform(X_train, y_train)

    # Apply SMOTE to the preprocessed training data
    X_train_resampled, y_train_resampled = smote.fit_resample(X_train_preprocess

    # Fit the model to the resampled data
    model.fit(X_train_resampled, y_train_resampled)

    # Predictions
    y_train_pred = model.predict(X_train_resampled)
    y_test_pred = model.predict(preprocessor.transform(X_test))

    # Print training classification report and confusion matrix
    print(f"Training Report for {model_name}:")
    print(classification_report(y_train_resampled, y_train_pred))
    print(f"Training Confusion Matrix for {model_name}:")
    print(confusion_matrix(y_train_resampled, y_train_pred))

    # Print testing classification report and confusion matrix
    print(f"Testing Report for {model_name}:")
    print(classification_report(y_test, y_test_pred))
    print(f"Testing Confusion Matrix for {model_name}:")
    print(confusion_matrix(y_test, y_test_pred))
```

In [158…

```
for model_name, model in models:
    evaluate_classification_model(model_name, preprocessor, smote, model, X_trai
Training Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.79      0.73      0.76      4130
           1       0.75      0.81      0.78      4130

    accuracy                           0.77      8260
   macro avg       0.77      0.77      0.77      8260
weighted avg       0.77      0.77      0.77      8260

Training Confusion Matrix for Logistic Regression:
[[3016 1114]
 [ 796 3334]]
Testing Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.90      0.71      0.79      1033
           1       0.49      0.78      0.60       374

    accuracy                           0.73      1407
   macro avg       0.70      0.74      0.70      1407
weighted avg       0.79      0.73      0.74      1407

Testing Confusion Matrix for Logistic Regression:
[[734 299]
 [ 83 291]]
Training Report for Support Vector Machines (SVM):
              precision    recall  f1-score   support

           0       0.84      0.80      0.82      4130
           1       0.81      0.85      0.83      4130

    accuracy                           0.83      8260
   macro avg       0.83      0.83      0.83      8260
weighted avg       0.83      0.83      0.83      8260

Training Confusion Matrix for Support Vector Machines (SVM):
[[3304  826]
 [ 614 3516]]
Testing Report for Support Vector Machines (SVM):
              precision    recall  f1-score   support

           0       0.88      0.76      0.82      1033
           1       0.52      0.72      0.61       374

    accuracy                           0.75      1407
   macro avg       0.70      0.74      0.71      1407
weighted avg       0.79      0.75      0.76      1407

Testing Confusion Matrix for Support Vector Machines (SVM):
[[789 244]
 [105 269]]
Training Report for Random Forest:
              precision    recall  f1-score   support

           0       0.93      0.90      0.92      4130
           1       0.91      0.93      0.92      4130

    accuracy                           0.92      8260
   macro avg       0.92      0.92      0.92      8260
```

```
weighted avg       0.92      0.92      0.92      8260

Training Confusion Matrix for Random Forest:
[[3737  393]
 [ 301 3829]]
Testing Report for Random Forest:
              precision    recall  f1-score   support

           0       0.84      0.81      0.82      1033
           1       0.52      0.58      0.55       374

    accuracy                           0.75      1407
   macro avg       0.68      0.69      0.69      1407
weighted avg       0.76      0.75      0.75      1407

Testing Confusion Matrix for Random Forest:
[[836 197]
 [158 216]]
Training Report for Gradient Boosting Machines (XGBoost):
              precision    recall  f1-score   support

           0       0.89      0.87      0.88      4130
           1       0.87      0.89      0.88      4130

    accuracy                           0.88      8260
   macro avg       0.88      0.88      0.88      8260
weighted avg       0.88      0.88      0.88      8260

Training Confusion Matrix for Gradient Boosting Machines (XGBoost):
[[3596  534]
 [ 450 3680]]
Testing Report for Gradient Boosting Machines (XGBoost):
              precision    recall  f1-score   support

           0       0.84      0.80      0.82      1033
           1       0.52      0.59      0.55       374

    accuracy                           0.75      1407
   macro avg       0.68      0.70      0.69      1407
weighted avg       0.76      0.75      0.75      1407

Testing Confusion Matrix for Gradient Boosting Machines (XGBoost):
[[831 202]
 [153 221]]
Training Report for K-Nearest Neighbors (KNN):
              precision    recall  f1-score   support

           0       0.84      0.80      0.82      4130
           1       0.81      0.85      0.83      4130

    accuracy                           0.83      8260
   macro avg       0.83      0.83      0.83      8260
weighted avg       0.83      0.83      0.83      8260

Training Confusion Matrix for K-Nearest Neighbors (KNN):
[[3320  810]
 [ 614 3516]]
Testing Report for K-Nearest Neighbors (KNN):
              precision    recall  f1-score   support
           0       0.84      0.72      0.78      1033
           1       0.44      0.62      0.52       374

    accuracy                           0.69      1407
   macro avg       0.64      0.67      0.65      1407
weighted avg       0.73      0.69      0.71      1407

Testing Confusion Matrix for K-Nearest Neighbors (KNN):
[[744 289]
 [143 231]]
Training Report for Decision Trees:
              precision    recall  f1-score   support

           0       0.91      0.92      0.92      4130
           1       0.92      0.91      0.92      4130

    accuracy                           0.92      8260
   macro avg       0.92      0.92      0.92      8260
weighted avg       0.92      0.92      0.92      8260

Training Confusion Matrix for Decision Trees:
[[3814  316]
 [ 378 3752]]
Testing Report for Decision Trees:
              precision    recall  f1-score   support

           0       0.83      0.79      0.81      1033
           1       0.49      0.54      0.51       374

    accuracy                           0.73      1407
   macro avg       0.66      0.67      0.66      1407
weighted avg       0.74      0.73      0.73      1407

Testing Confusion Matrix for Decision Trees:
[[819 214]
 [171 203]]
Training Report for Random Forest:
              precision    recall  f1-score   support

           0       0.93      0.90      0.91      4130
           1       0.90      0.93      0.92      4130

    accuracy                           0.92      8260
   macro avg       0.92      0.92      0.92      8260
weighted avg       0.92      0.92      0.92      8260

Training Confusion Matrix for Random Forest:
[[3727  403]
 [ 291 3839]]
Testing Report for Random Forest:
              precision    recall  f1-score   support

           0       0.84      0.82      0.83      1033
           1       0.53      0.57      0.55       374

    accuracy                           0.75      1407
   macro avg       0.69      0.69      0.69      1407
weighted avg       0.76      0.75      0.75      1407

Testing Confusion Matrix for Random Forest:
```

```
[[846 187]
 [162 212]]
Training Report for Extra Trees:
              precision    recall  f1-score   support

           0       0.91      0.92      0.92      4130
           1       0.92      0.91      0.92      4130

    accuracy                           0.92      8260
   macro avg       0.92      0.92      0.92      8260
weighted avg       0.92      0.92      0.92      8260

Training Confusion Matrix for Extra Trees:
[[3814  316]
 [ 378 3752]]
Testing Report for Extra Trees:
              precision    recall  f1-score   support

           0       0.83      0.83      0.83      1033
           1       0.53      0.54      0.54       374

    accuracy                           0.75      1407
   macro avg       0.68      0.68      0.68      1407
weighted avg       0.75      0.75      0.75      1407

Testing Confusion Matrix for Extra Trees:
[[853 180]
 [171 203]]
Training Report for Ensemble Methods (AdaBoost):
              precision    recall  f1-score   support

           0       0.83      0.76      0.79      4130
           1       0.78      0.84      0.81      4130

    accuracy                           0.80      8260
   macro avg       0.80      0.80      0.80      8260
weighted avg       0.80      0.80      0.80      8260

Training Confusion Matrix for Ensemble Methods (AdaBoost):
[[3119 1011]
 [ 646 3484]]
Testing Report for Ensemble Methods (AdaBoost):
              precision    recall  f1-score   support

           0       0.89      0.74      0.81      1033
           1       0.51      0.74      0.60       374

    accuracy                           0.74      1407
   macro avg       0.70      0.74      0.71      1407
weighted avg       0.79      0.74      0.75      1407

Testing Confusion Matrix for Ensemble Methods (AdaBoost):
[[768 265]
 [ 99 275]]
Training Report for Ensemble Methods (Voting Classifier):
              precision    recall  f1-score   support

           0       0.86      0.81      0.83      4130
           1       0.82      0.87      0.84      4130
    accuracy                           0.84      8260
   macro avg       0.84      0.84      0.84      8260
weighted avg       0.84      0.84      0.84      8260

Training Confusion Matrix for Ensemble Methods (Voting Classifier):
[[3330  800]
 [ 538 3592]]
Testing Report for Ensemble Methods (Voting Classifier):
              precision    recall  f1-score   support

           0       0.89      0.77      0.82      1033
           1       0.53      0.72      0.61       374

    accuracy                           0.76      1407
   macro avg       0.71      0.75      0.72      1407
weighted avg       0.79      0.76      0.77      1407

Testing Confusion Matrix for Ensemble Methods (Voting Classifier):
[[795 238]
 [103 271]]
Training Report for Ensemble Methods (Bagging):
              precision    recall  f1-score   support

           0       0.92      0.90      0.91      4130
           1       0.90      0.92      0.91      4130

    accuracy                           0.91      8260
   macro avg       0.91      0.91      0.91      8260
weighted avg       0.91      0.91      0.91      8260

Training Confusion Matrix for Ensemble Methods (Bagging):
[[3705  425]
 [ 316 3814]]
Testing Report for Ensemble Methods (Bagging):
              precision    recall  f1-score   support

           0       0.84      0.80      0.82      1033
           1       0.52      0.58      0.55       374

    accuracy                           0.74      1407
   macro avg       0.68      0.69      0.69      1407
weighted avg       0.76      0.74      0.75      1407

Testing Confusion Matrix for Ensemble Methods (Bagging):
[[830 203]
 [156 218]]
Training Report for Ensemble Methods (Gradient Boosting):
              precision    recall  f1-score   support

           0       0.86      0.79      0.82      4130
           1       0.81      0.87      0.84      4130

    accuracy                           0.83      8260
   macro avg       0.83      0.83      0.83      8260
weighted avg       0.83      0.83      0.83      8260

Training Confusion Matrix for Ensemble Methods (Gradient Boosting):
[[3268  862]
 [ 532 3598]]
Testing Report for Ensemble Methods (Gradient Boosting):
```

```
              precision    recall  f1-score   support

           0       0.87      0.76      0.81      1033
           1       0.51      0.70      0.59       374

    accuracy                           0.74      1407
   macro avg       0.69      0.73      0.70      1407
weighted avg       0.78      0.74      0.76      1407

Testing Confusion Matrix for Ensemble Methods (Gradient Boosting):
[[787 246]
 [113 261]]
```

1. Random Forest

- **Training Accuracy:** 0.92
- **Testing Accuracy:** 0.75
- **Training F1-score:** 0.92 (Macro Average)
- **Testing F1-score:** 0.55 (Macro Average)
- **Reason:** Random Forest achieved the highest training accuracy and F1-score among the models. Despite its drop in performance on the testing data, it still maintains a relatively high accuracy compared to other models.

2. Gradient Boosting Machines (XGBoost)

- **Training Accuracy:** 0.88
- **Testing Accuracy:** 0.75
- **Training F1-score:** 0.88 (Macro Average)
- **Testing F1-score:** 0.55 (Macro Average)
- **Reason:** XGBoost has a high training accuracy and F1-score and performs similarly to Random Forest on the testing data. It is well-regarded for its effectiveness and ability to handle complex patterns.

3. Support Vector Machines (SVM)

- **Training Accuracy:** 0.83
- **Testing Accuracy:** 0.75
- **Training F1-score:** 0.83 (Macro Average)
- **Testing F1-score:** 0.61 (Macro Average)
- **Reason:** SVM performs well on both training and testing datasets, maintaining a good balance between precision and recall on the test set.

4. Ensemble Methods (Voting Classifier)

- **Training Accuracy:** 0.84
- **Testing Accuracy:** 0.76
- **Training F1-score:** 0.84 (Macro Average)
- **Testing F1-score:** 0.72 (Macro Average)
- **Reason:** The Voting Classifier shows consistent performance across both training and testing phases. Its balance of precision and recall on the test set makes it a

strong candidate.

5. **Ensemble Methods (Bagging)**

- **Training Accuracy:** 0.91
- **Testing Accuracy:** 0.74
- **Training F1-score:** 0.91 (Macro Average)
- **Testing F1-score:** 0.69 (Macro Average)
- **Reason:** Bagging also achieves high training accuracy and maintains a competitive performance on the test set. It is generally robust against overfitting due to its ensemble nature.

Summary:

- **Best Models on Training Data:** Random Forest and XGBoost show the best training metrics.
- **Best Models on Testing Data:** Voting Classifier and SVM show strong performance on the test set.
- **Most Balanced Models:** Voting Classifier and Bagging offer a good balance between training and testing performance.

`I will perform more feature selection and hyperparameter tuning to increase the test accuracy.`