Customer Churn Classification Using ML Models

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
from google.colab import drive
drive.mount('/content/drive')

# load the data and review
df = pd.read_csv('/content/drive/My Drive/Colab Data/WA_Fn-UseC_-Telco-Customer
df.head()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, ca
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneServi
0	7590-VHVEG	Female	0	Yes	No	1	
1	5575-GNVDE	Male	0	No	No	34	`
2	3668-QPYBK	Male	0	No	No	2	`
3	7795-CFOCW	Male	0	No	No	45	
4	9237-HQITU	Female	0	No	No	2	`

5 rows × 21 columns

Read & Explore Data

```
#load the data
df.head(10)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneServi
0	7590-VHVEG	Female	0	Yes	No	1	
1	5575-GNVDE	Male	0	No	No	34	`
2	3668-QPYBK	Male	0	No	No	2	`
3	7795-CFOCW	Male	0	No	No	45	
4	9237-HQITU	Female	0	No	No	2	`
5	9305-CDSKC	Female	0	No	No	8	`
6	1452-KIOVK	Male	0	No	Yes	22	`
7	6713-OKOMC	Female	0	No	No	10	
8	7892-POOKP	Female	0	Yes	No	28	`
9	6388-TABGU	Male	0	No	Yes	62	`

10 rows × 21 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

```
df.shape
```

```
(7043, 21)
```

```
df.duplicated().sum()
```

```
0
```

```
df.isna().sum()
```

	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

```
gender_count= df['gender'].value_counts()
gender_count
```

gender	count
Male	3555
Female	3488

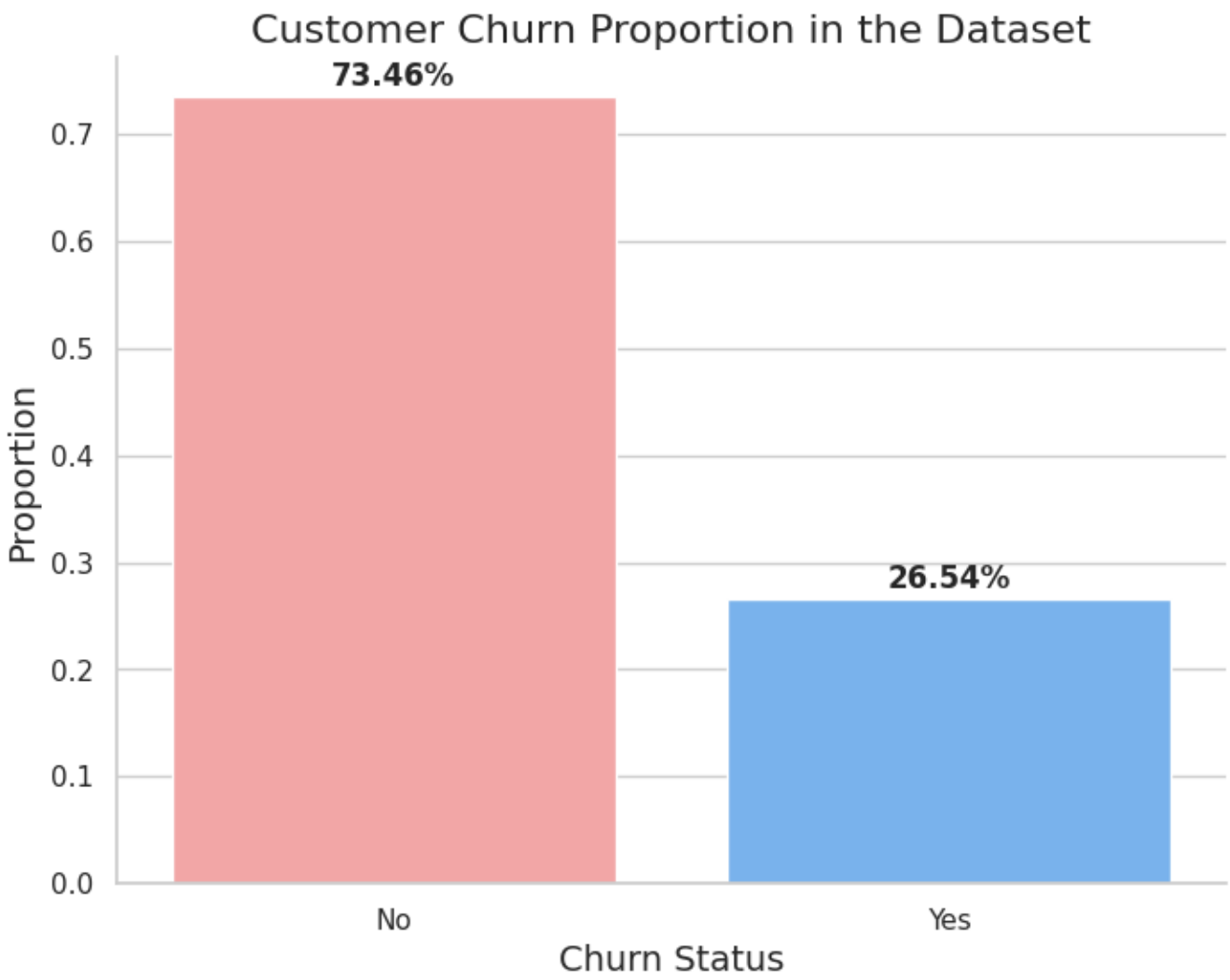
dtype: int64

```
churn_ratio = df['Churn'].value_counts(normalize=True)['Yes']
ratio_count= df['Churn'].value_counts()
print(ratio_count)

print(f"Churn Ratio: {churn_ratio:.2f}")
```

```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
Churn Ratio: 0.27
```

```
ratio_count= df['Churn'].value_counts()
churn_ratio = ratio_count / ratio_count.sum()

# Set up the plot style and color palette
sns.set(style="whitegrid", palette="pastel")

# Create a bar plot for Churn ratios
plt.figure(figsize=(8, 6))
ax = sns.barplot(x=churn_ratio.index, y=churn_ratio.values, palette=["#FF9999",

# Add value annotations
for index, value in enumerate(churn_ratio.values):
    plt.text(index, value + 0.01, f"{value:.2%}", ha='center', fontweight='bolc

# Set labels and title
plt.xlabel('Churn Status', fontsize=14)
plt.ylabel('Proportion', fontsize=14)
plt.title('Customer Churn Proportion in the Dataset', fontsize=16)

# Remove top and right borders for a cleaner look
sns.despine()
```

```
# Show the plot
plt.show()
```



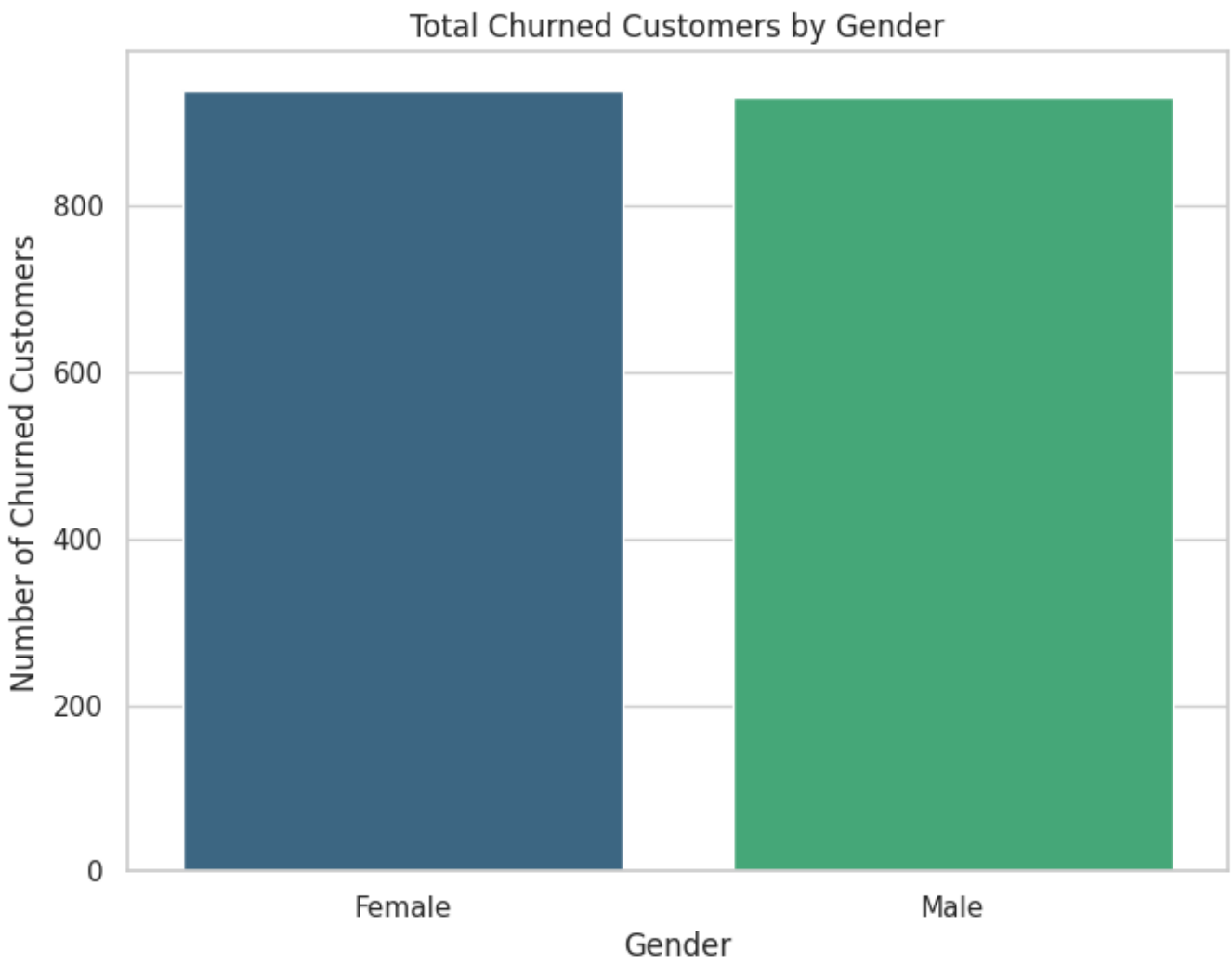
EDA

```
churned_customers = df[df['Churn'] == 'Yes']
```

```
churned_by_gender = churned_customers.groupby('gender').size().reset_index(name
```

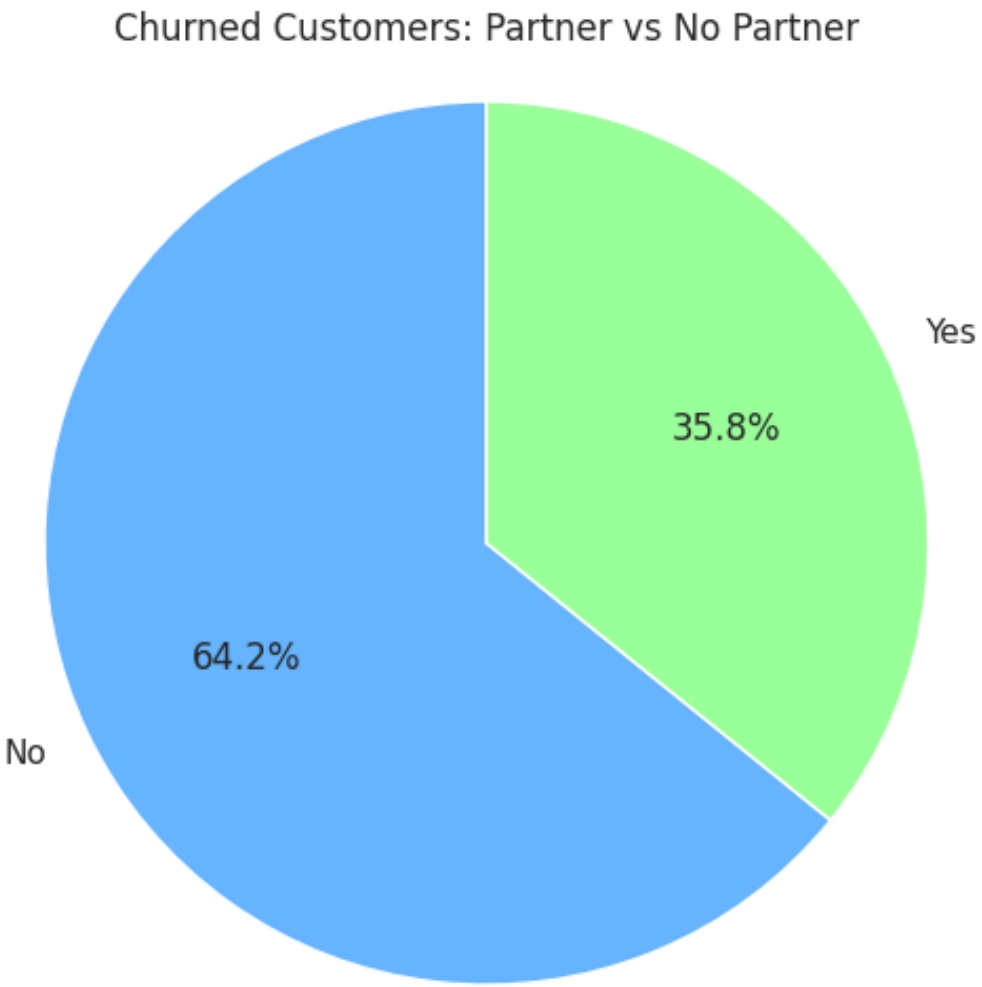
Insights for customer Churn

```
plt.figure(figsize=(8, 6))
sns.barplot(x='gender', y='Total Churned', data=churned_by_gender, palette='vir
plt.title('Total Churned Customers by Gender')
plt.xlabel('Gender')
plt.ylabel('Number of Churned Customers')
plt.show()
```




```
churned_by_partner = churned_customers['Partner'].value_counts()
```

```
plt.figure(figsize=(8, 6))
plt.pie(churned_by_partner, labels=churned_by_partner.index, autopct='%1.1f%%',
plt.title('Churned Customers: Partner vs No Partner')
plt.axis('equal')
plt.show()
```

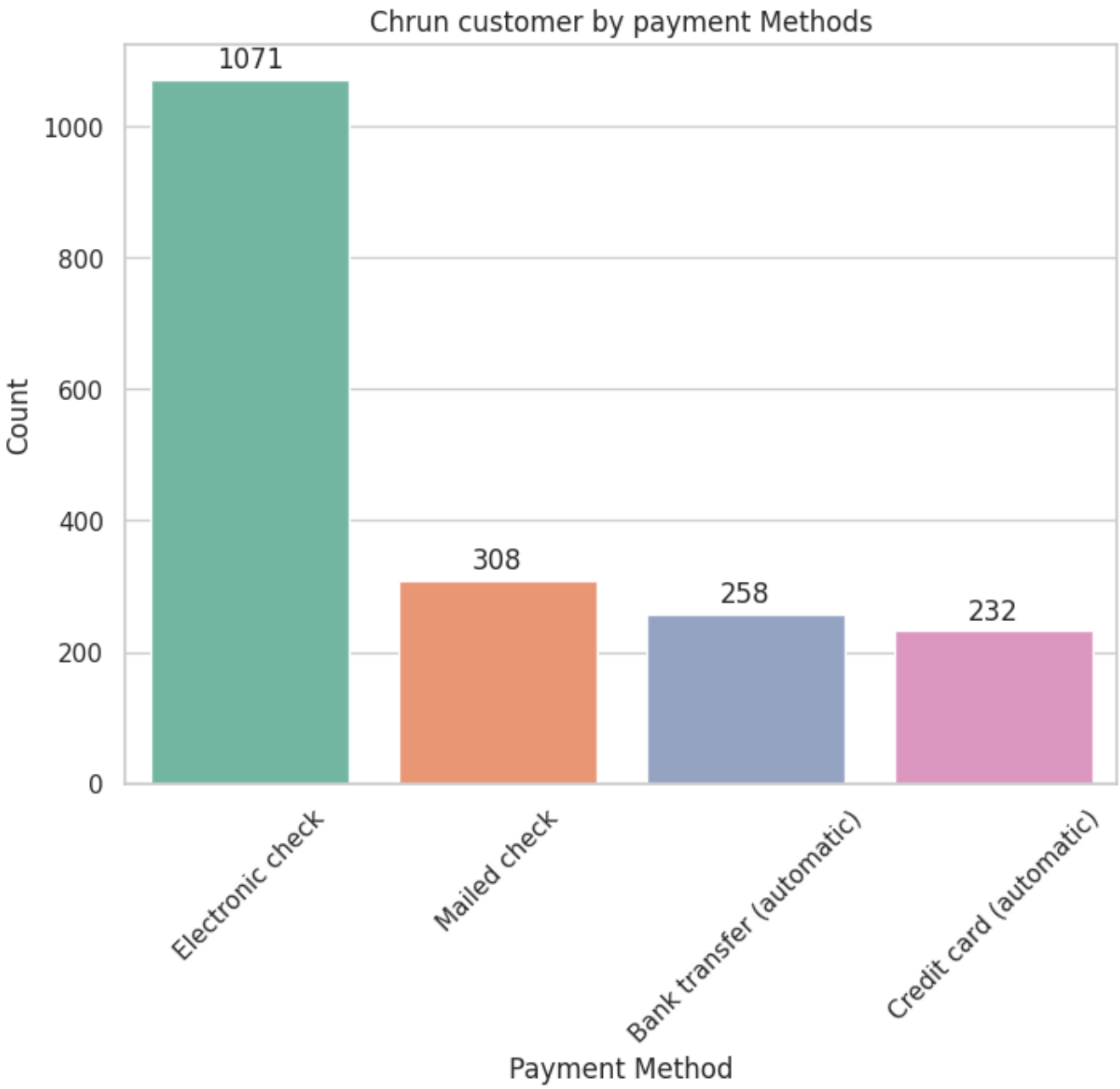


```
payment_count = churned_customers['PaymentMethod'].value_counts()
payment_count
```

PaymentMethod	count
Electronic check	1071
Mailed check	308
Bank transfer (automatic)	258
Credit card (automatic)	232

dtype: int64

```
plt.figure(figsize=(8,6))
ax = sns.barplot(x=payment_count.index,y=payment_count.values,palette='Set2')
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9),
                textcoords='offset points')

plt.title('Chrun customer by payment Methods')
plt.xlabel('Payment Method')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

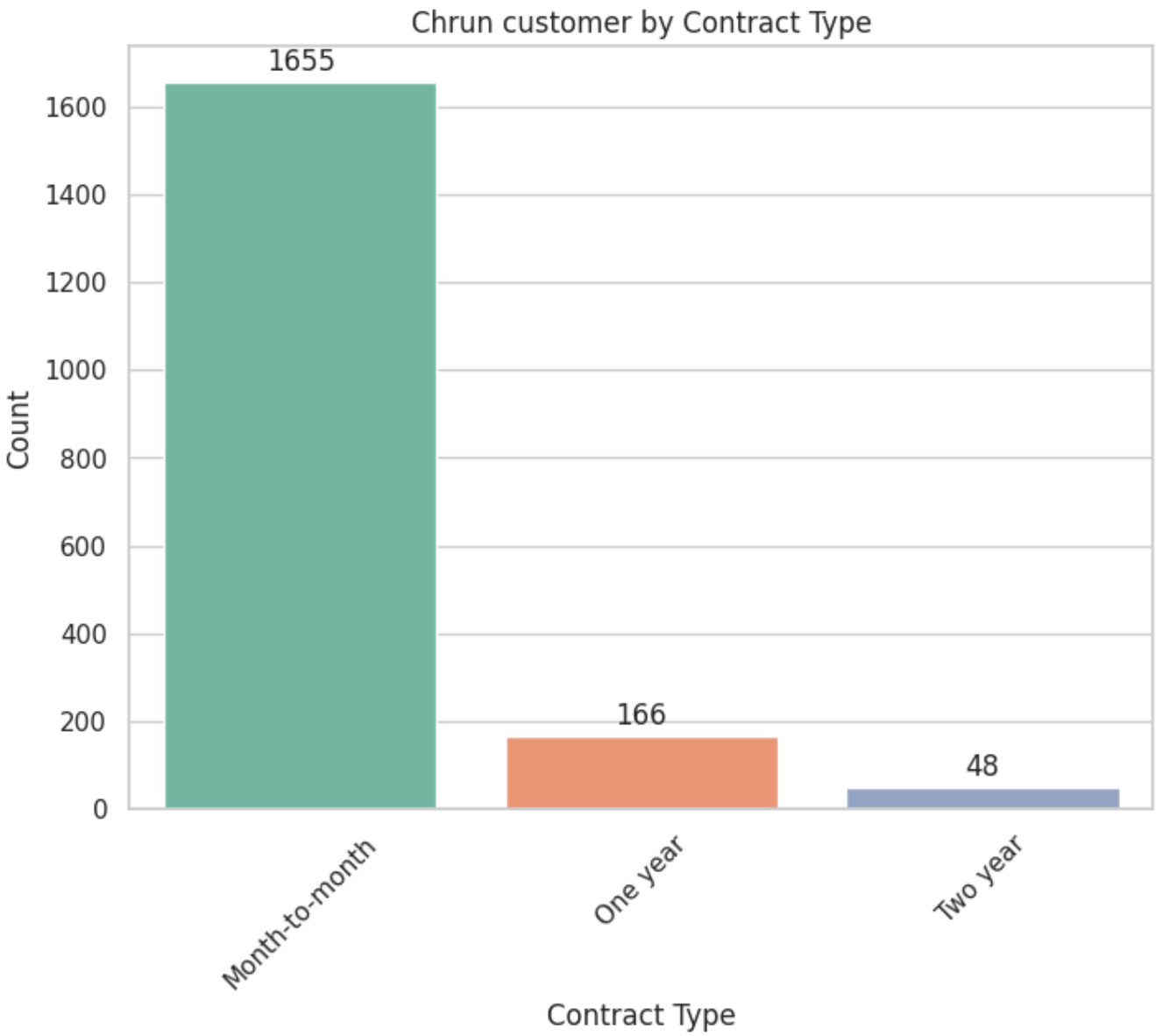

Chrun customer by payment Methods
1071
308
258
232
1000
800
600
400
200
0
Count
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
Payment Method

```
contract_count = churned_customers['Contract'].value_counts()
contract_count
```

Contract	count
Month-to-month	1655
One year	166
Two year	48

dtype: int64

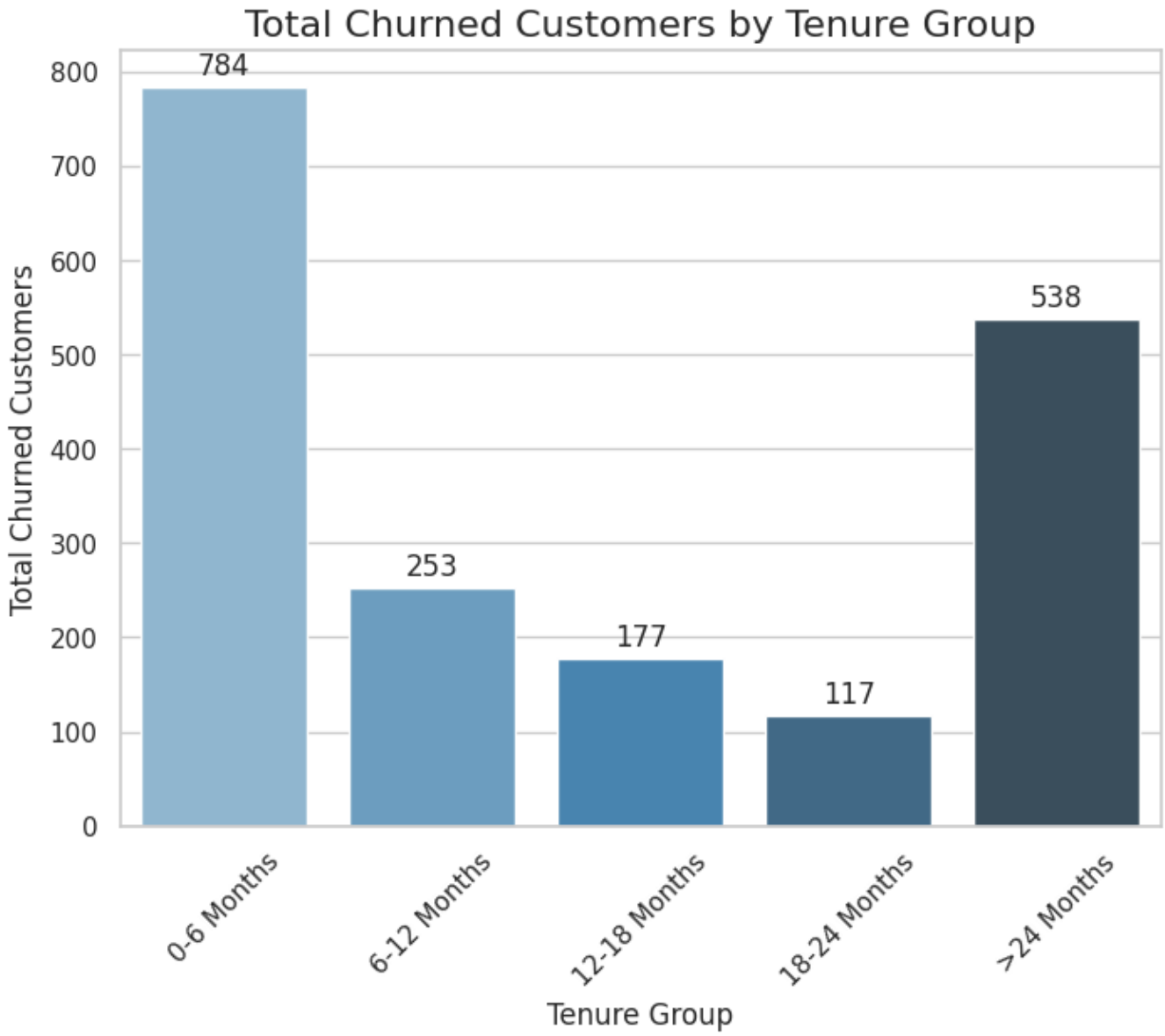
```
plt.figure(figsize=(8,6))
ax = sns.barplot(x=contract_count.index,y=contract_count.values,palette='Set2')
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9),
                textcoords='offset points')

plt.title('Chrun customer by Contract Type')
plt.xlabel('Contract Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```




```
churned_customers['tenure'].unique()
```

```
array([ 2,  8, 28, 49, 10,  1, 47, 17,  5, 34, 11, 15, 18,  9,  7, 12, 25,
       68, 55, 37,  3, 27, 20,  4, 58, 53, 13,  6, 19, 59, 16, 52, 24, 32,
       38, 54, 43, 63, 21, 69, 22, 61, 60, 48, 40, 23, 39, 35, 56, 65, 33,
       30, 45, 46, 62, 70, 50, 44, 71, 26, 14, 41, 66, 64, 29, 42, 67, 51,
       31, 57, 36, 72])
```

```
bins = [0, 6, 12, 18, 24, churned_customers['tenure'].max()]
labels = ['0-6 Months', '6-12 Months', '12-18 Months', '18-24 Months', '>24 Mor
churned_customers['tenure_group'] = pd.cut(churned_customers['tenure'], bins=bi



grouped = churned_customers.groupby('tenure_group').size().reset_index(name='tc



plt.figure(figsize=(8,6))
sns.barplot(x='tenure_group', y='total_churned_customers', data=grouped, palett

for p in plt.gca().patches:
    plt.gca().annotate(f'{int(p.get_height())}',
                       (p.get_x() + p.get_width() / 2., p.get_height()),
                       ha='center', va='center', xytext=(0, 9),
                       textcoords='offset points')

plt.title('Total Churned Customers by Tenure Group', fontsize=16)
plt.xlabel('Tenure Group', fontsize=12)
plt.ylabel('Total Churned Customers', fontsize=12)
plt.xticks(rotation=45)
plt.show()
```

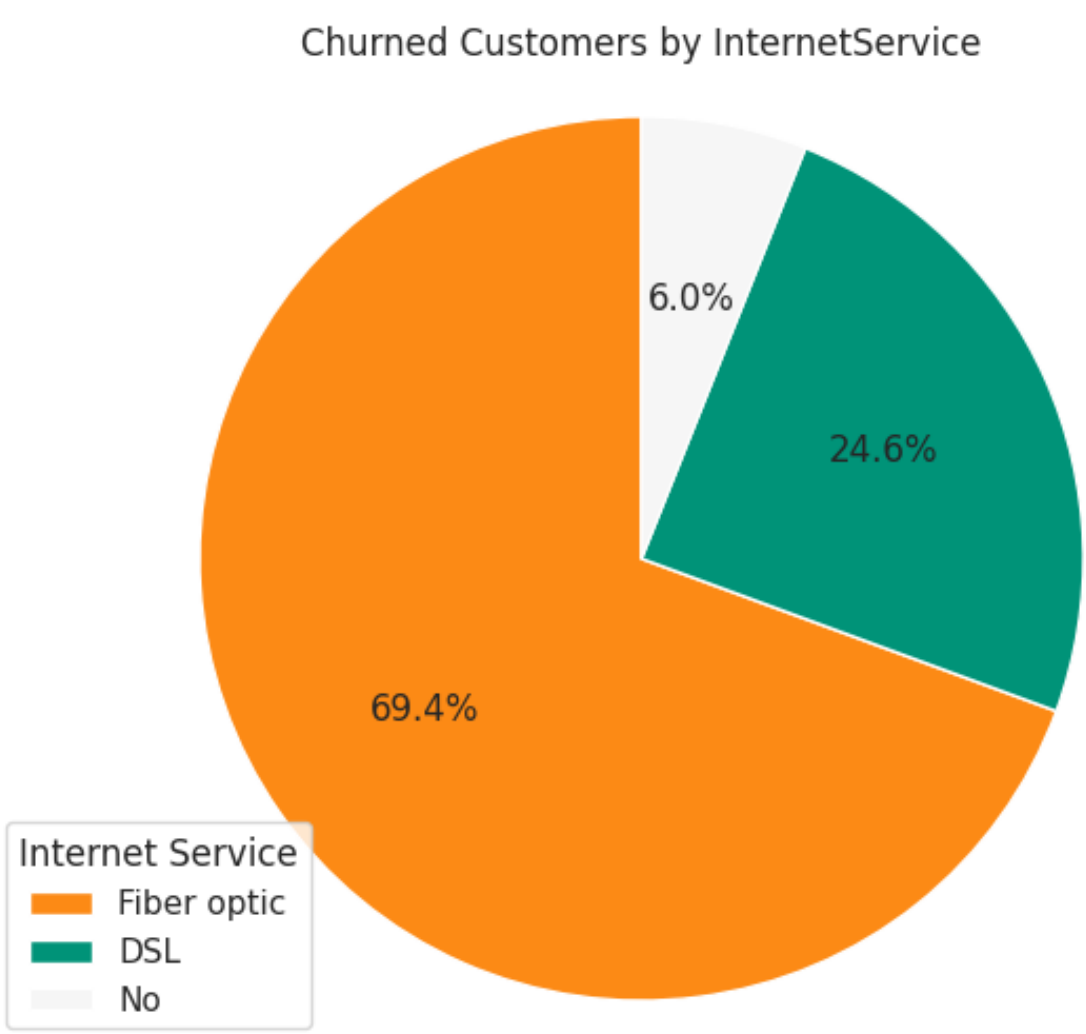

Total Churned Customers by Tenure Group
Total Churned Customers
800
700
600
500
400
300
200
100
0
784
253
177
117
538
0-6 Months
6-12 Months
12-18 Months
18-24 Months
>24 Months
Tenure Group

```
InternetService_count =  churned_customers['InternetService'].value_counts()
InternetService_count
```

InternetService	count
Fiber optic	1297
DSL	459
No	113

dtype: int64

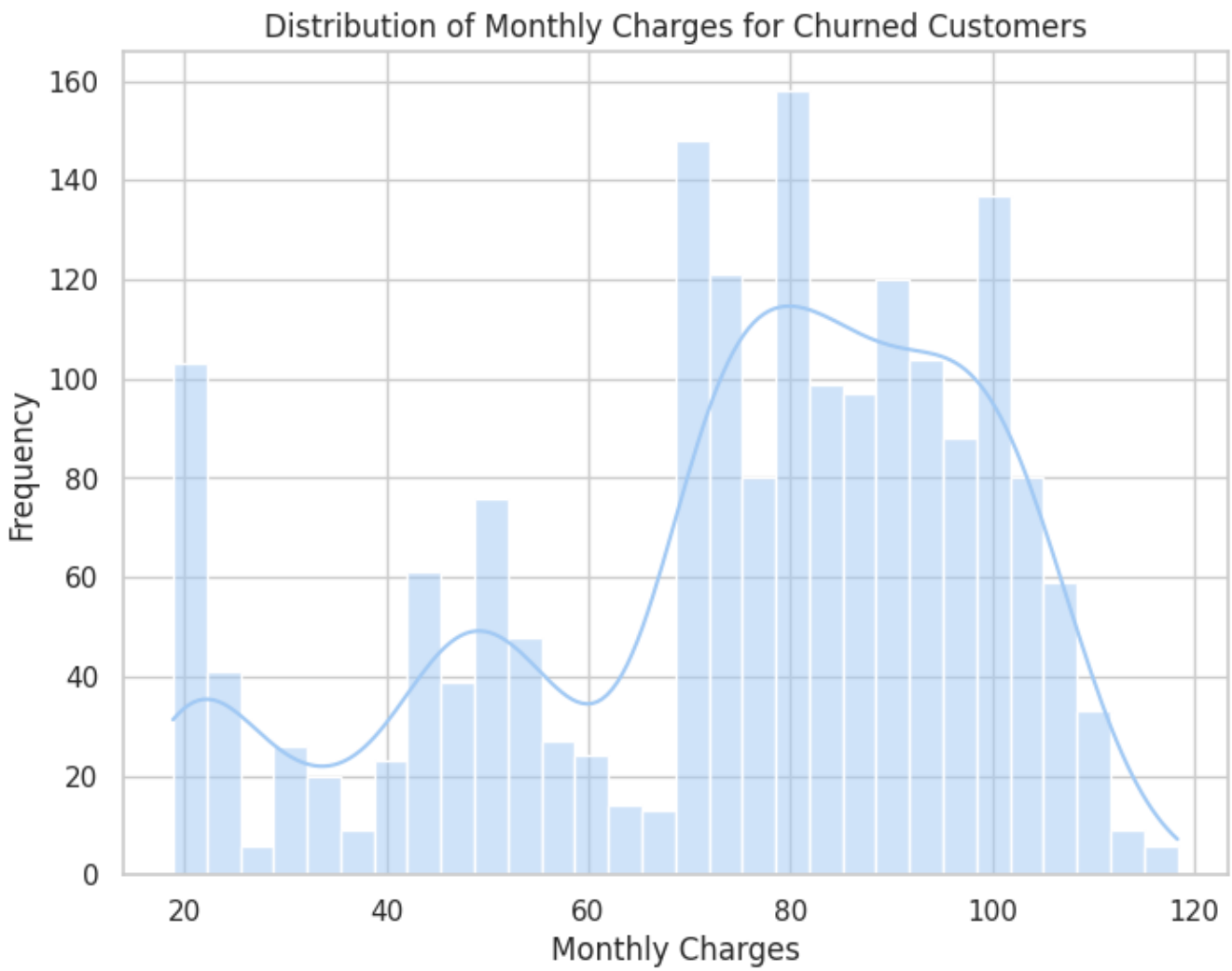
```
plt.figure(figsize=(8, 6))
plt.pie(InternetService_count, labels=None, autopct='%1.1f%%', colors=['#FC8A15
plt.legend(InternetService_count.index, title="Internet Service", loc="best")
plt.title('Churned Customers by InternetService')
plt.axis('equal')
plt.show()
```



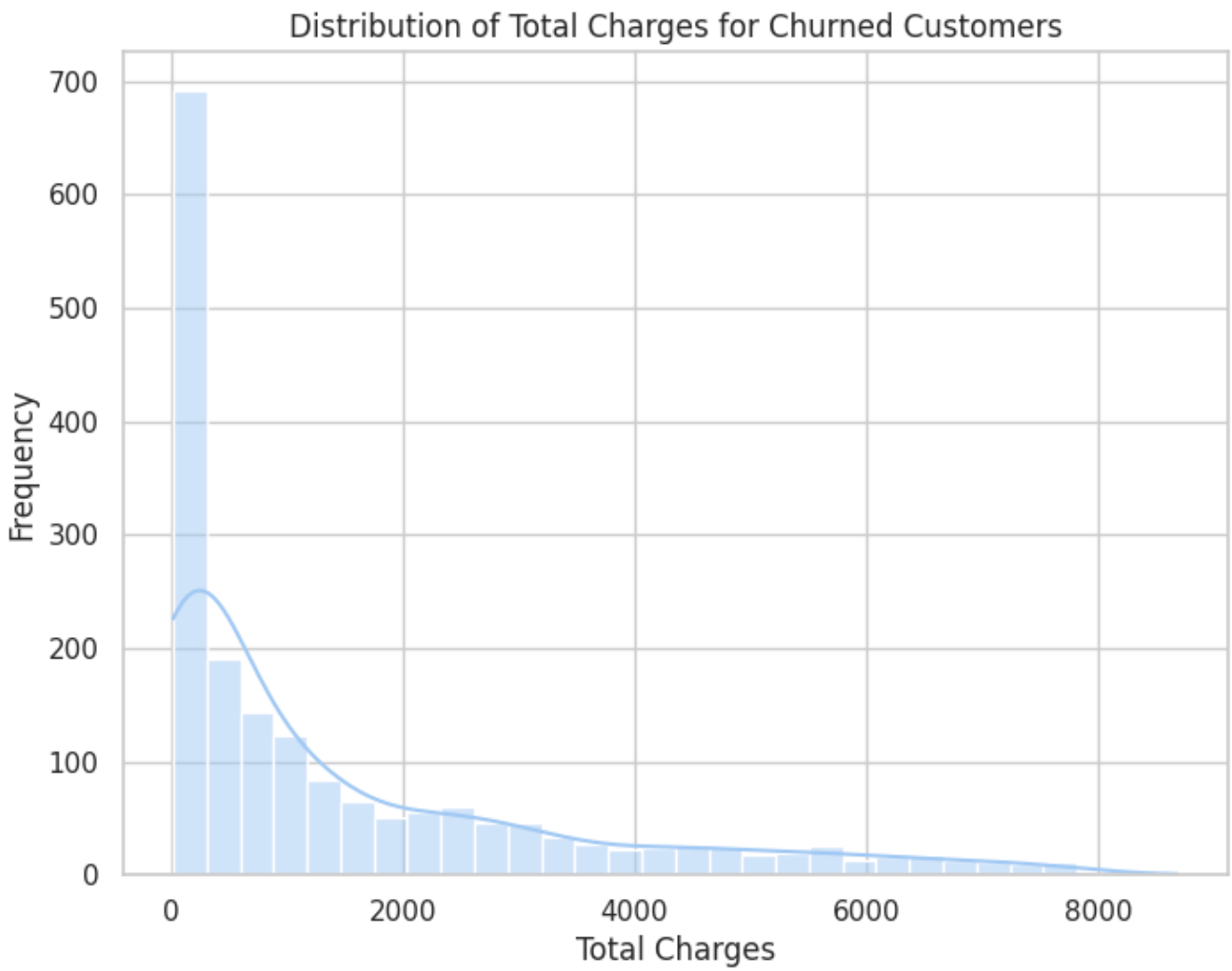

```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

```
churned_customers = df[df['Churn'] == 'Yes']
```

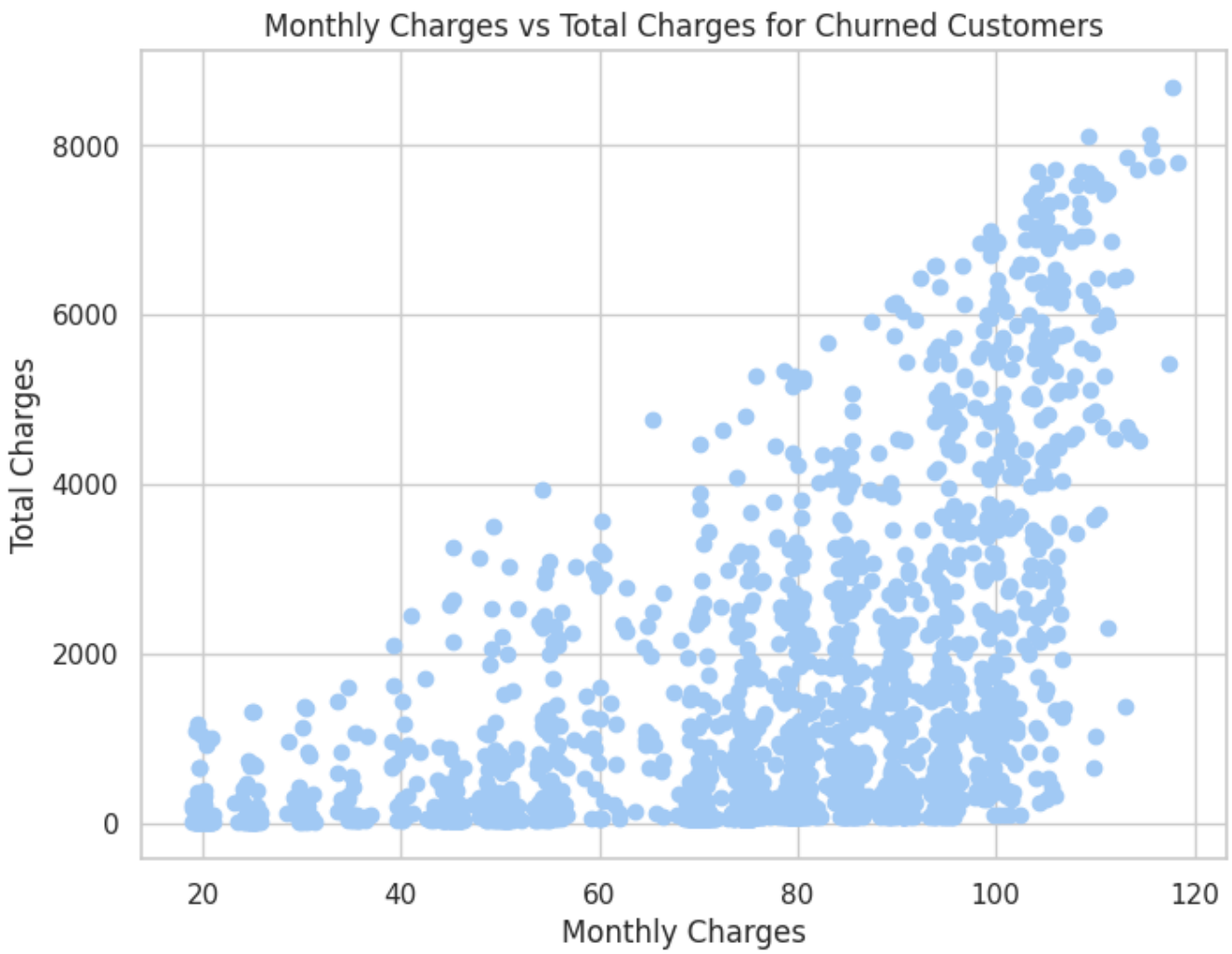
```
plt.figure(figsize=(8, 6))
sns.histplot(churned_customers['MonthlyCharges'], bins=30, kde=True)
plt.title('Distribution of Monthly Charges for Churned Customers')
plt.xlabel('Monthly Charges')
plt.ylabel('Frequency')
plt.show()
```



```
plt.figure(figsize=(8, 6))
sns.histplot(churned_customers['TotalCharges'], bins=30, kde=True)
plt.title('Distribution of Total Charges for Churned Customers')
plt.xlabel('Total Charges')
plt.ylabel('Frequency')
plt.show()
```



```
plt.figure(figsize=(8, 6))
plt.scatter(churned_customers['MonthlyCharges'], churned_customers['TotalCharge
plt.title('Monthly Charges vs Total Charges for Churned Customers')
plt.xlabel('Monthly Charges')
plt.ylabel('Total Charges')
plt.show()
```



Data Preprocessing

```
df['MultipleLines'].value_counts()
```

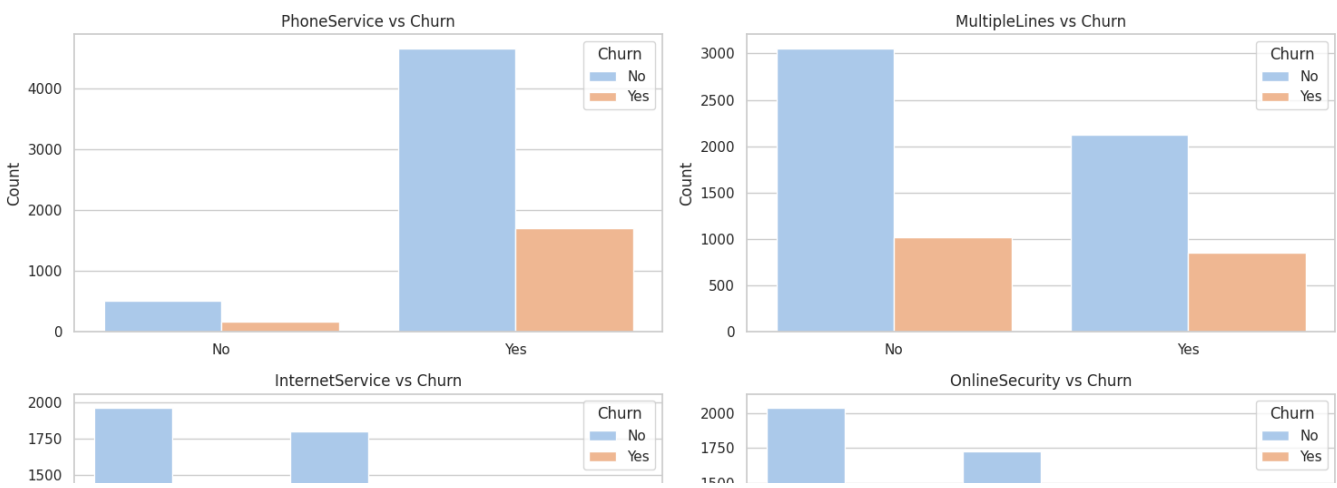
MultipleLines	count
No	3390
Yes	2971
No phone service	682

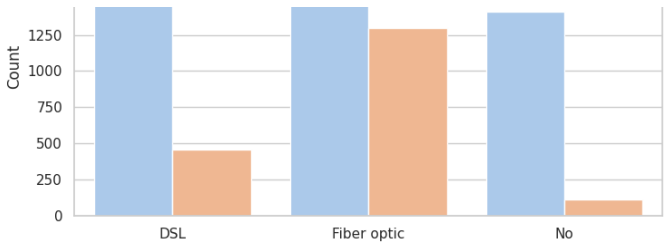
dtype: int64

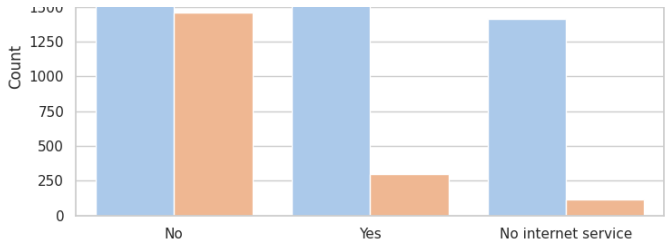
```
df['MultipleLines'] = df['MultipleLines'].replace('No phone service', 'No')
```

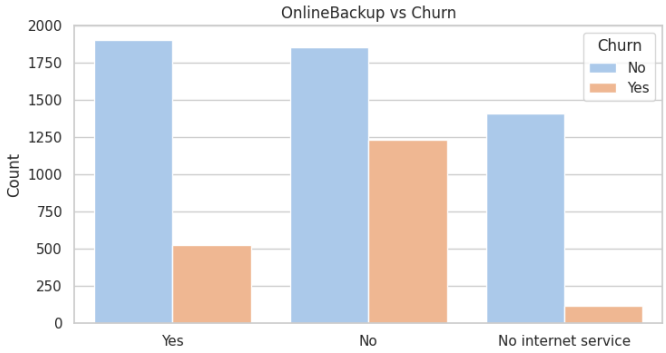
```
service_columns = [
    'PhoneService', 'MultipleLines', 'InternetService',
    'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
    'TechSupport', 'StreamingTV', 'StreamingMovies',
    'PaperlessBilling'
]
```

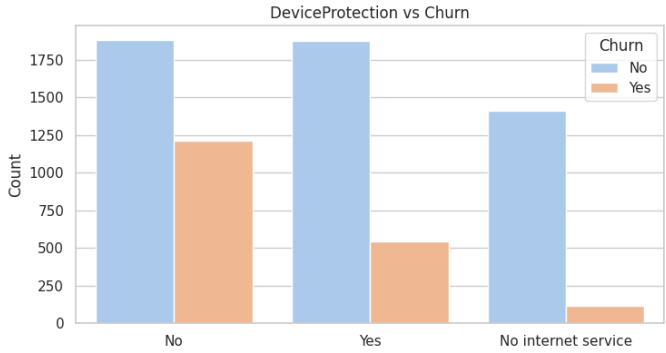
```
plt.figure(figsize=(15, 20))

for i, col in enumerate(service_columns, 1):
    plt.subplot(5, 2, i)
    sns.countplot(data=df, x=col, hue='Churn')
    plt.title(f'{col} vs Churn')
    plt.xlabel('')
    plt.ylabel('Count')

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

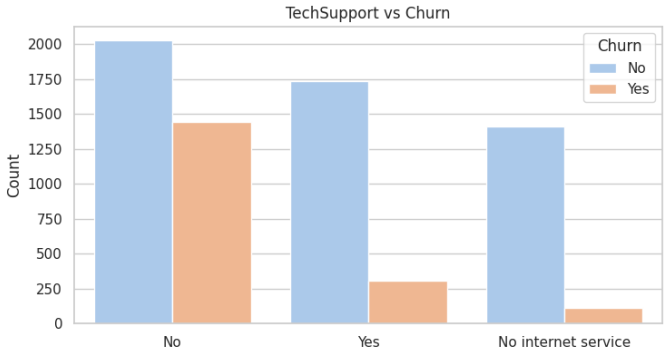


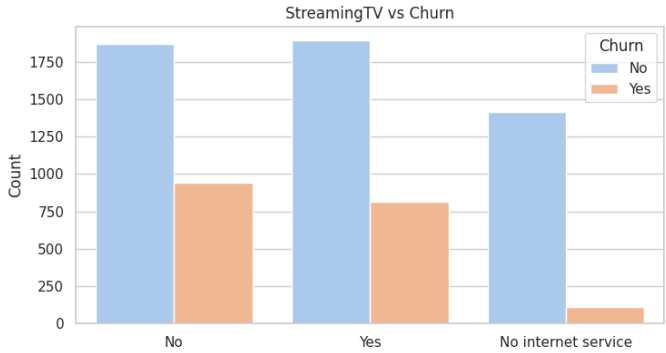

Count
DSL
Fiber optic
No

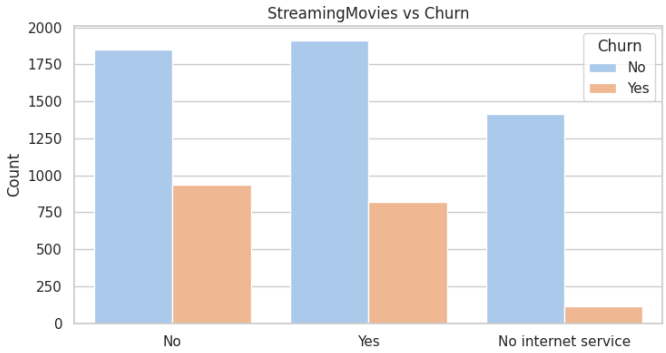

Count
No
Yes
No internet service

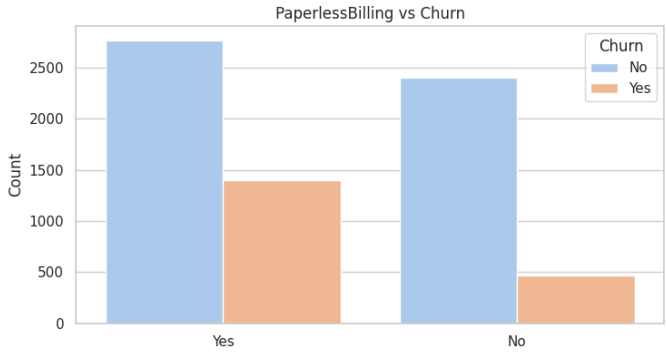

OnlineBackup vs Churn
Churn
No
Yes
Count
Yes
No
No internet service


DeviceProtection vs Churn
Churn
No
Yes
Count
No
Yes
No internet service


TechSupport vs Churn
Churn
No
Yes
Count
No
Yes
No internet service


StreamingTV vs Churn
Churn
No
Yes
Count
No
Yes
No internet service


StreamingMovies vs Churn
Churn
No
Yes
Count
No
Yes
No internet service


PaperlessBilling vs Churn
Churn
No
Yes
Count
Yes
No

```
df.replace('No internet service', 'No', inplace=True)

df.drop('customerID',axis=1,inplace=True)

df.drop('gender',axis=1,inplace=True)
```

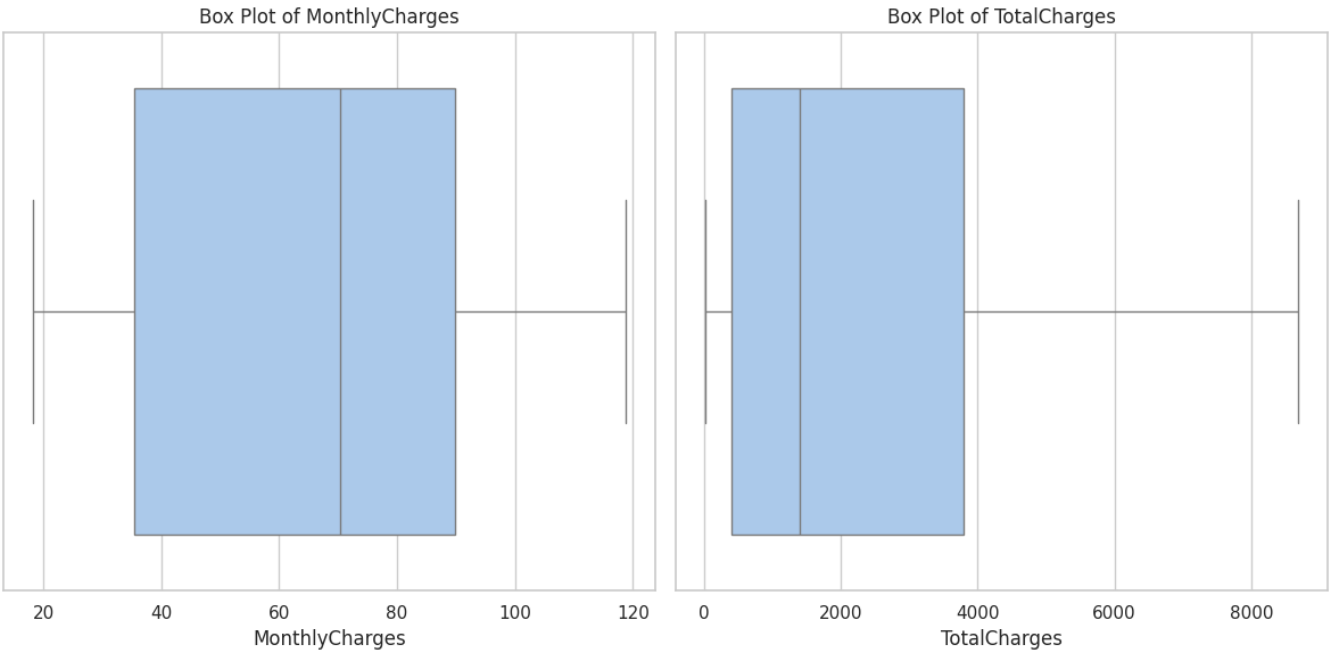
```
df.head()
```

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	I
0	0	Yes	No	1	No	No	
1	0	No	No	34	Yes	No	
2	0	No	No	2	Yes	No	
3	0	No	No	45	No	No	
4	0	No	No	2	Yes	No	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   SeniorCitizen     7043 non-null   int64
 1   Partner           7043 non-null   object
 2   Dependents        7043 non-null   object
 3   tenure            7043 non-null   int64
 4   PhoneService      7043 non-null   object
 5   MultipleLines     7043 non-null   object
 6   InternetService   7043 non-null   object
 7   OnlineSecurity    7043 non-null   object
 8   OnlineBackup      7043 non-null   object
 9   DeviceProtection  7043 non-null   object
 10  TechSupport       7043 non-null   object
 11  StreamingTV       7043 non-null   object
 12  StreamingMovies   7043 non-null   object
 13  Contract          7043 non-null   object
 14  PaperlessBilling  7043 non-null   object
 15  PaymentMethod     7043 non-null   object
 16  MonthlyCharges    7043 non-null   float64
 17  TotalCharges      7032 non-null   float64
 18  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(15)
memory usage: 1.0+ MB
```

```
numerical_features = ['MonthlyCharges', 'TotalCharges',]

plt.figure(figsize=(12, 6))

for i, feature in enumerate(numerical_features, 1):
    plt.subplot(1, len(numerical_features), i)
    sns.boxplot(data=df, x=feature)
    plt.title(f'Box Plot of {feature}')

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



```
def replace_yes_no(df, columns):

    for col in columns:
        df[col] = df[col].replace({'Yes': 1, 'No': 0})

columns_to_replace = [
    "Partner", "Dependents", "PhoneService", "OnlineSecurity", "OnlineBackup",
    "DeviceProtection", "TechSupport", "StreamingTV", "StreamingMovies",
    "PaperlessBilling", "Churn"
]

replace_yes_no(df, columns_to_replace)

df.head()
```

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	I
0	0	1	0	1	0	No	
1	0	0	0	34	1	No	
2	0	0	0	2	1	No	
3	0	0	0	45	0	No	
4	0	0	0	2	1	No	

```
from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()
columns = ["MultipleLines", "Contract", "PaymentMethod", "InternetService"]

for column in columns:
    df[column] = labelencoder.fit_transform(df[column])

df.head()
```

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	I
0	0	1	0	1	0	0	
1	0	0	0	34	1	0	
2	0	0	0	2	1	0	
3	0	0	0	45	0	0	
4	0	0	0	2	1	0	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   SeniorCitizen     7043 non-null   int64
 1   Partner           7043 non-null   int64
 2   Dependents        7043 non-null   int64
 3   tenure            7043 non-null   int64
 4   PhoneService      7043 non-null   int64
 5   MultipleLines     7043 non-null   int64
 6   InternetService   7043 non-null   int64
 7   OnlineSecurity    7043 non-null   int64
 8   OnlineBackup      7043 non-null   int64
 9   DeviceProtection  7043 non-null   int64
 10  TechSupport       7043 non-null   int64
 11  StreamingTV       7043 non-null   int64
 12  StreamingMovies   7043 non-null   int64
 13  Contract          7043 non-null   int64
 14  PaperlessBilling  7043 non-null   int64
 15  PaymentMethod     7043 non-null   int64
 16  MonthlyCharges    7043 non-null   float64
 17  TotalCharges      7032 non-null   float64
 18  Churn             7043 non-null   int64
dtypes: float64(2), int64(17)
memory usage: 1.0 MB
```

```
df.isna().sum()
```

	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

```
df['TotalCharges'].fillna(value=df['TotalCharges'],inplace=True)
```

```
df = df.dropna()
df.isna().sum()
```

	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

Build Models

```
!pip install catboost
```

```
Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/local/lib/python3
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/di
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDi
from sklearn.metrics import roc_curve
from imblearn.over_sampling import SMOTE
from catboost import CatBoostClassifier
from lightgbm import LGBMClassifier
```

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

```
smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, y)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.2
```

```
def evaluate_model(model, X_test, y_test, model_name):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    conf_matrix = confusion_matrix(y_test, y_pred)
    class_report = classification_report(y_test, y_pred)


    print(f'{model_name} Accuracy: {accuracy:.2f}')
    print(f'{model_name} Classification Report:\n{class_report}')


    disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix, display_labels=
    disp.plot(cmap='Blues')
    plt.title(f'{model_name} Confusion Matrix')
    plt.show()
```

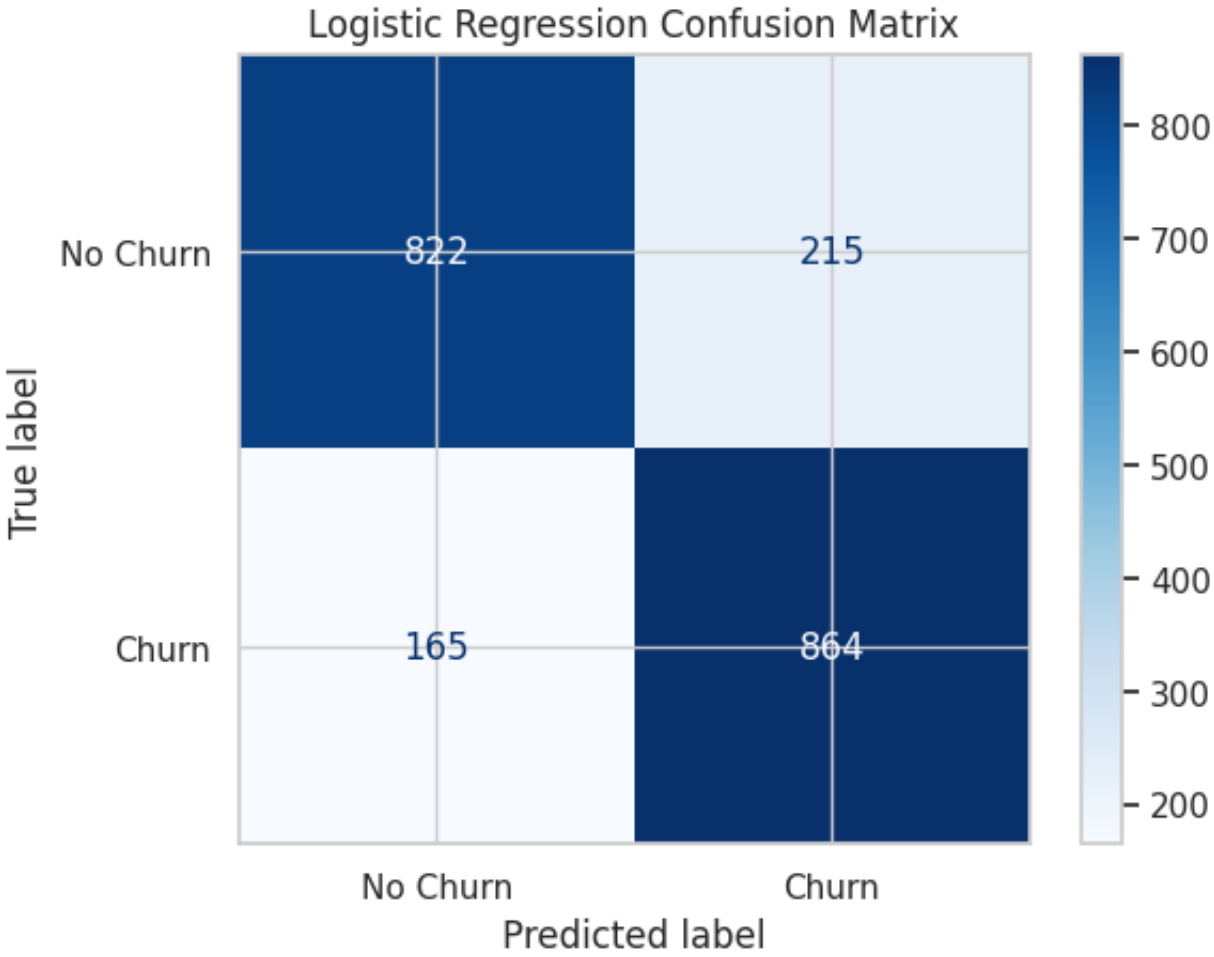
LogisticRegression

```
log_reg = LogisticRegression(max_iter=1000, random_state=42)
log_reg.fit(X_train ,y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

```
print("\nLogistic Regression Results:")
evaluate_model(log_reg, X_test, y_test, "Logistic Regression")
```

```
Logistic Regression Results:
Logistic Regression Accuracy: 0.82
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.79      0.81      1037
           1       0.80      0.84      0.82      1029

    accuracy                           0.82      2066
   macro avg       0.82      0.82      0.82      2066
weighted avg       0.82      0.82      0.82      2066
```



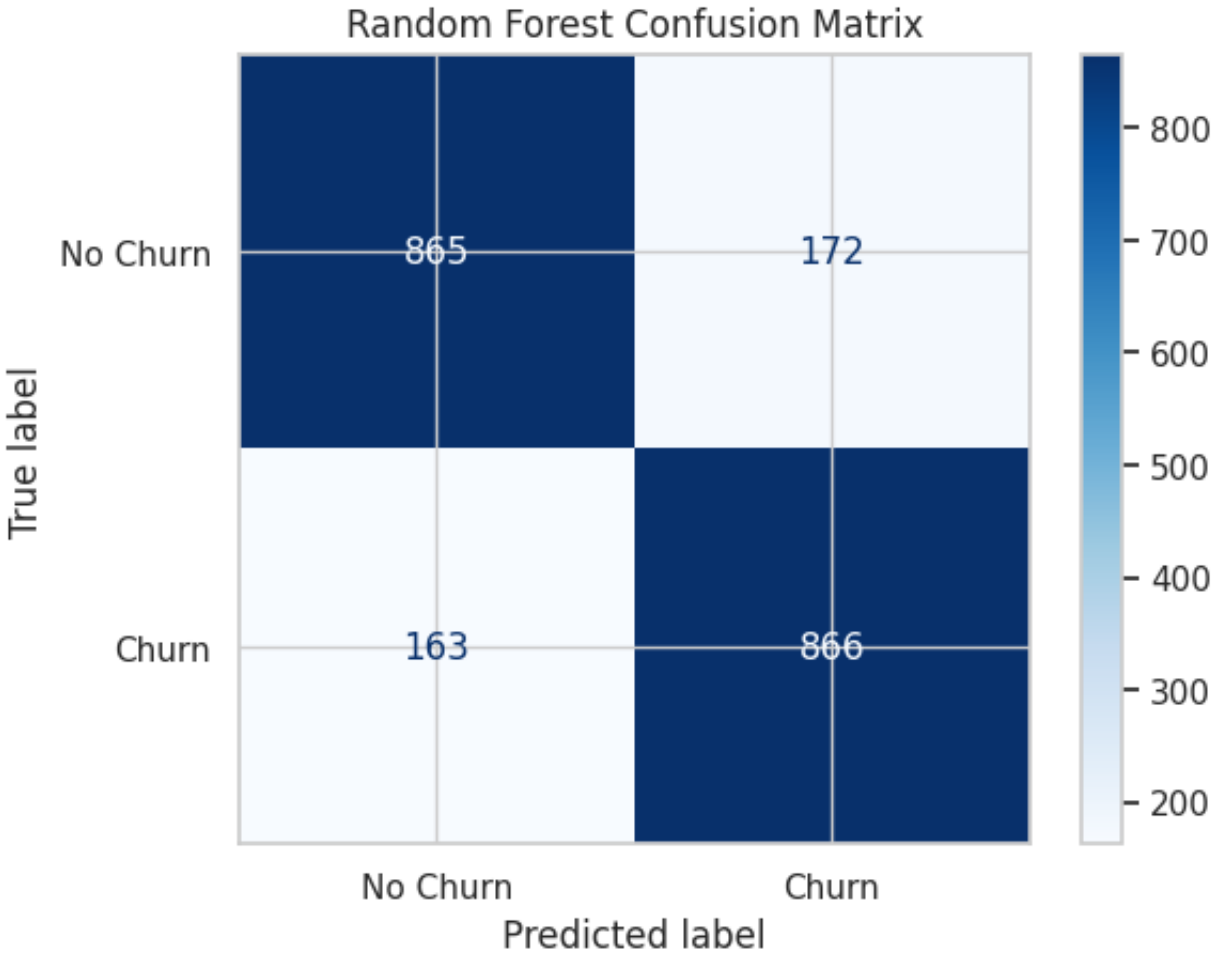
RandomForest

```
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
print("\nRandom Forest Results:")
evaluate_model(rf, X_test, y_test, "Random Forest")
```

```
Random Forest Results:
Random Forest Accuracy: 0.84
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.83      0.84      1037
           1       0.83      0.84      0.84      1029

    accuracy                           0.84      2066
   macro avg       0.84      0.84      0.84      2066
weighted avg       0.84      0.84      0.84      2066
```



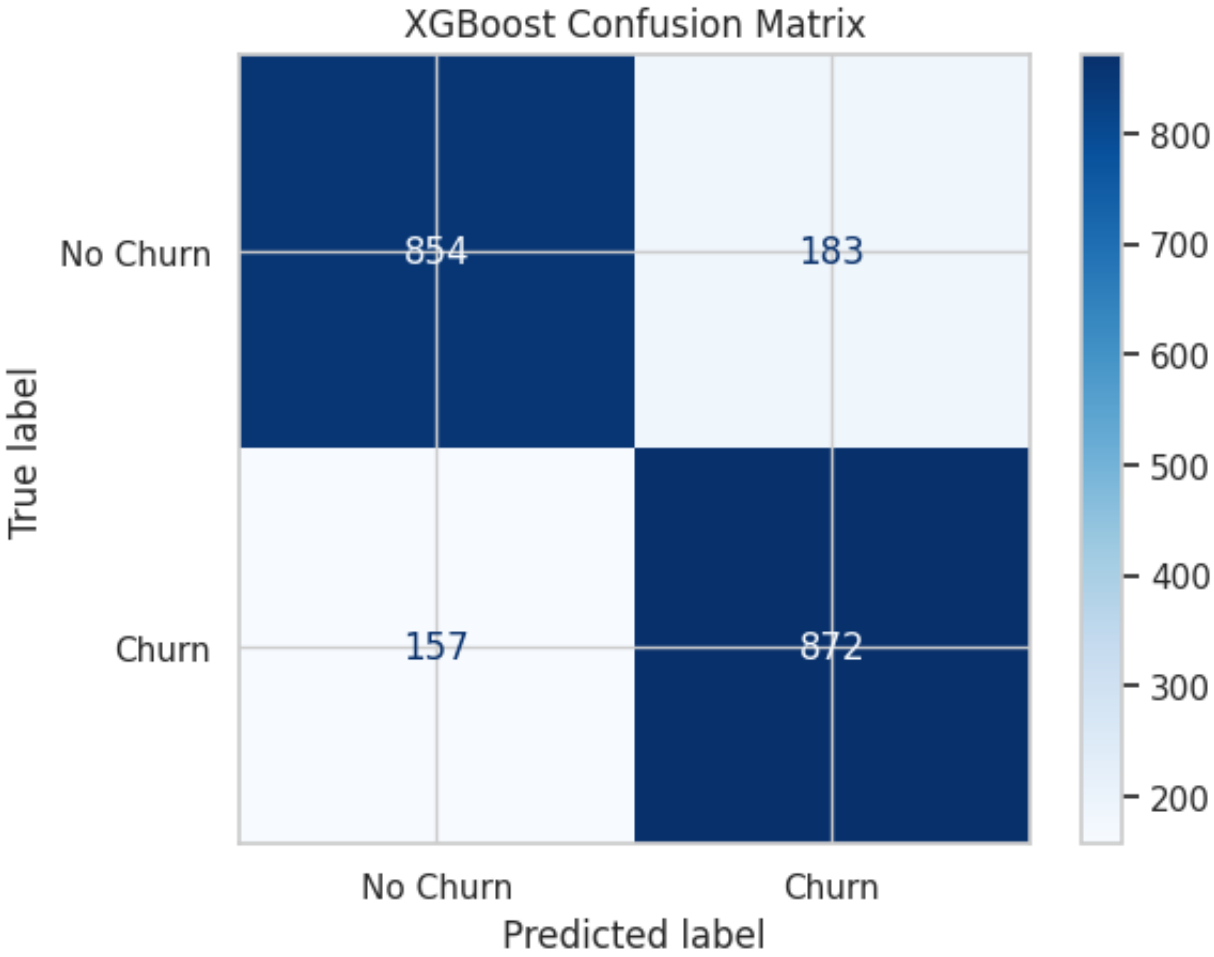
XGBoost

```
xgb = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', random_sta
xgb.fit(X_train, y_train)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=Nor
              enable_categorical=False, eval_metric='mlogloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=
              n_jobs=None, num_parallel_tree=None, random_state=42, ...)
```

```
print("\nXGBoost Results:")
evaluate_model(xgb, X_test, y_test, "XGBoost")
```

```
XGBoost Results:
XGBoost Accuracy: 0.84
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.82      0.83      1037
           1       0.83      0.85      0.84      1029

    accuracy                           0.84      2066
   macro avg       0.84      0.84      0.84      2066
weighted avg       0.84      0.84      0.84      2066
```



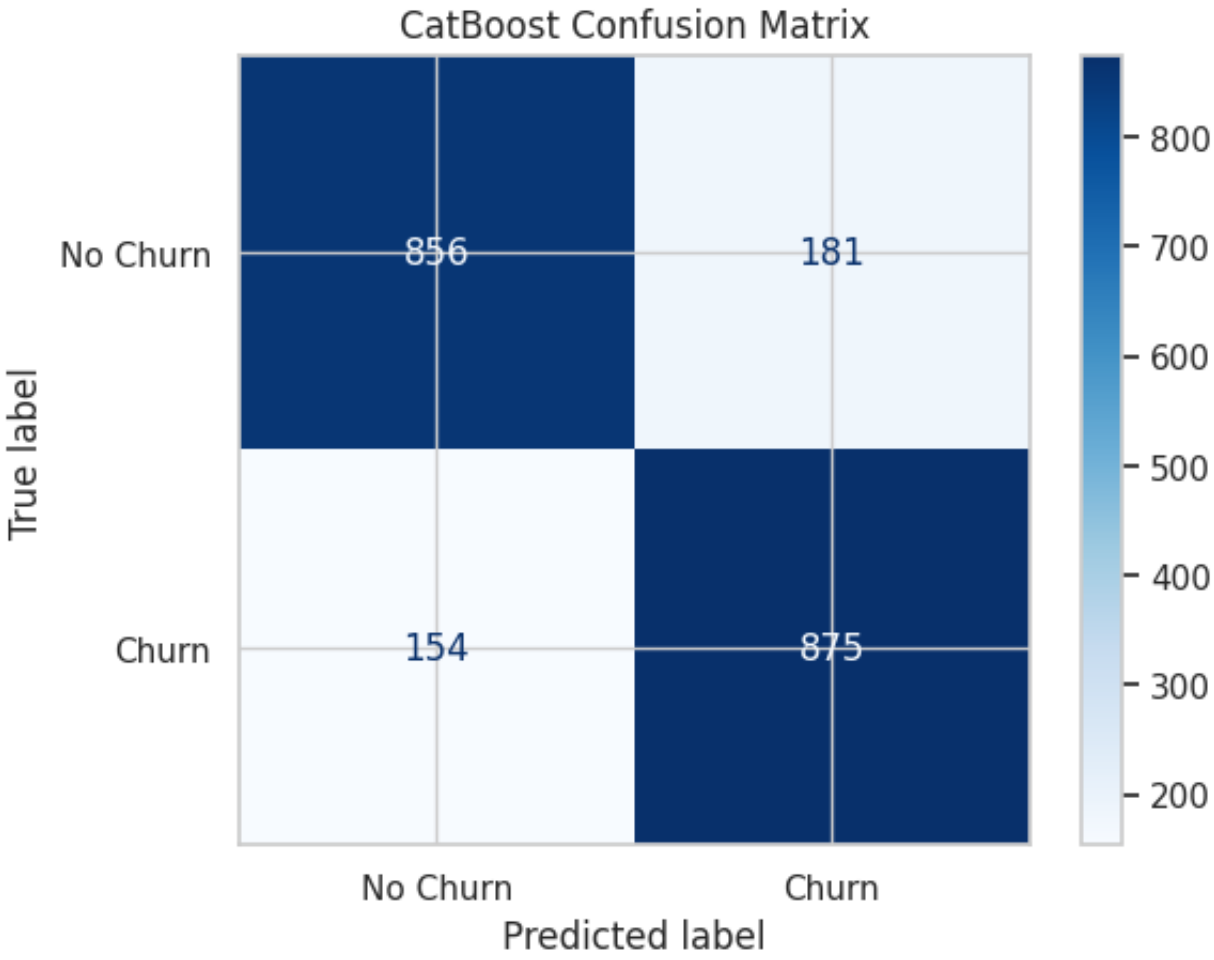
CatBoost

```
catboost = CatBoostClassifier(verbose=0, random_state=42)
catboost.fit(X_train, y_train)
```

```
<catboost.core.CatBoostClassifier at 0x7f05887b29b0>
```

```
print("\nCatBoost Results:")
evaluate_model(catboost, X_test, y_test, "CatBoost")
```

```
CatBoost Results:
CatBoost Accuracy: 0.84
CatBoost Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.83      0.84      1037
           1       0.83      0.85      0.84      1029

    accuracy                           0.84      2066
   macro avg       0.84      0.84      0.84      2066
weighted avg       0.84      0.84      0.84      2066
```



LightGBM

```
lgbm = LGBMClassifier(random_state=42)
lgbm.fit(X_train, y_train)
```

```
[LightGBM] [Info] Number of positive: 4134, number of negative: 4126
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of t
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 617
[LightGBM] [Info] Number of data points in the train set: 8260, number of u
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500484 -> initscore=0.001
[LightGBM] [Info] Start training from score 0.001937
```

```
▾ LGBMClassifier
LGBMClassifier(random_state=42)
```

```
print("\nLightGBM Results:")
evaluate_model(lgbm, X_test, y_test, "LightGBM")
```

```
LightGBM Results:
LightGBM Accuracy: 0.83
LightGBM Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.82      0.83      1037
           1       0.82      0.84      0.83      1029

    accuracy                           0.83      2066
   macro avg       0.83      0.83      0.83      2066
weighted avg       0.83      0.83      0.83      2066
```