NAAN MUDHALVAN PROJECT

PHASE 4 : Development Part 2

CUSTOMER CHURN PREDICTION

Team Members :

ANANYA K A – 2021115013

ANULATHA S K - 2021115014

ARADHYA M - 2021115015

SAKTHIVEL K – 2021115088

BALAJI J - 2021115323

INTRODUCTION

In this phase, we will continue building the project by performing the following:

- Exploratory data analysis (EDA)
- Statistical analysis
- Visualization
- Training Model

The above processes are performed on the already loaded and pre-processed dataset. During this stage, we will carry on with the project by conducting various analyses on the prepared dataset and finally visualization techniques are used to represent the same for better and clearer understanding.

DATA PRE-PROCESSING:

Once reliable data has been collected, it is time to clean and prepare the data for analysis. This process is Data pre-processing.

The above processes have already been performed and documented in the previous phase.

DATA COLLECTION:

Collecting data for Customer Churn Prediction is a vital aspect of understanding the effectiveness and impact of Customer views and their interests in order to perform proper analysis.

The source of the data for this project is,

Telco Customer Churn

EXPLORATORY DATA ANALYSIS (EDA):

Exploratory Data Analysis (EDA) is a critical phase in data analysis, typically performed early in the data exploration process. It involves a set of techniques and methods for summarizing, visualizing, and understanding the key characteristics and patterns within a dataset.

The primary goals of EDA are to:

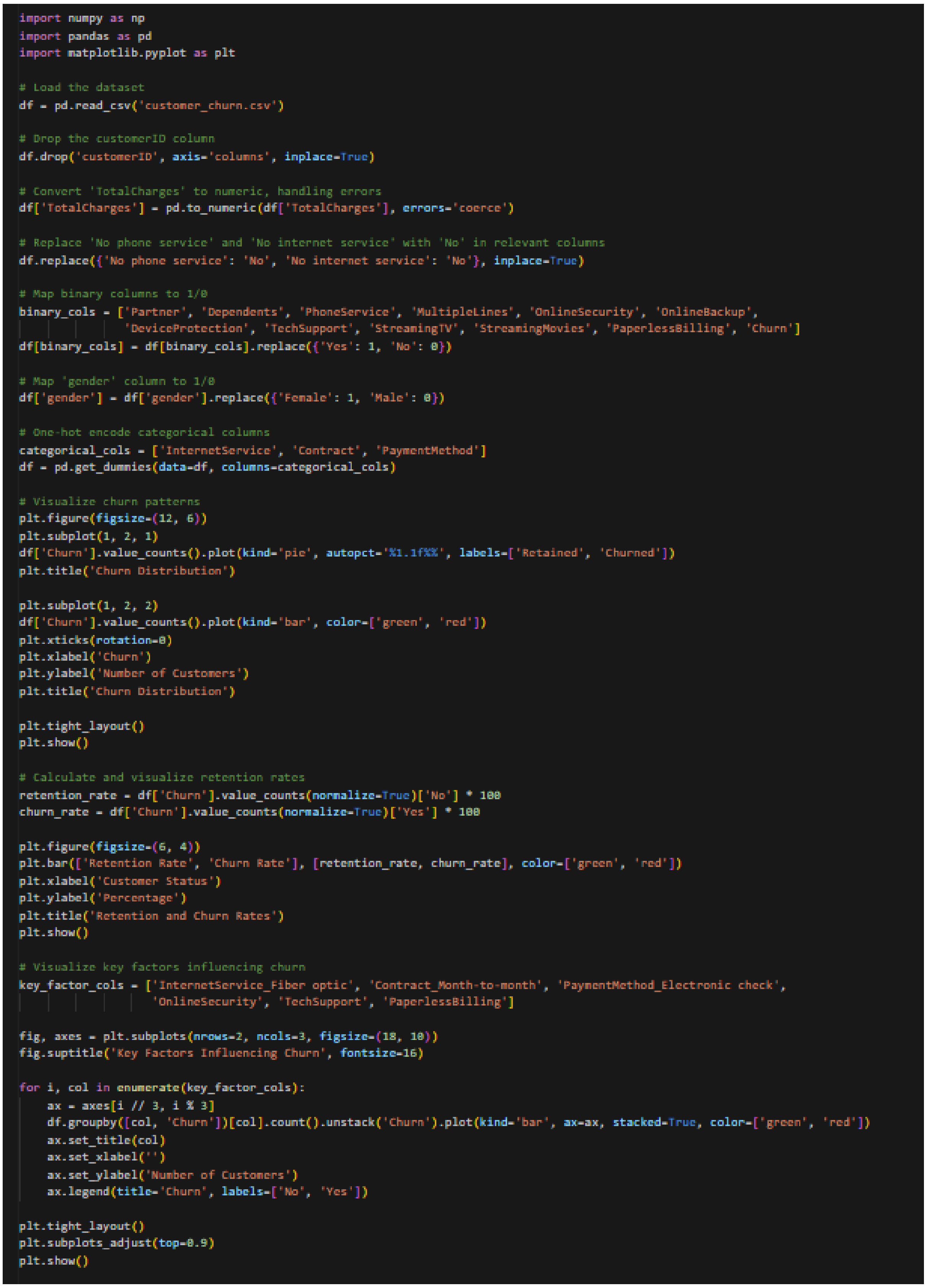
1. Gain an initial understanding of the data: EDA helps analysts and data scientists become familiar with the dataset, including its size, structure, and the types of variables it contains.

2. Identify data quality issues: EDA can reveal missing values, outliers, and potential errors in the data, allowing for data cleaning and preprocessing.

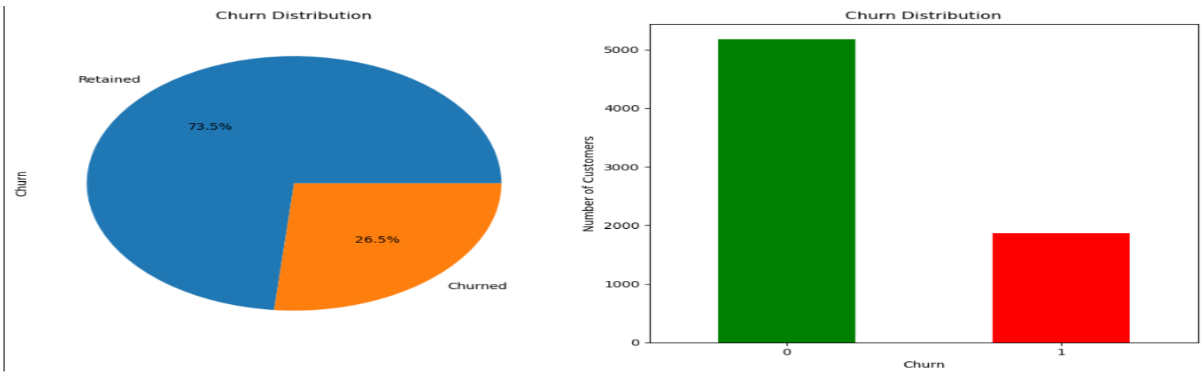
3. Detect patterns and relationships: EDA techniques, such as statistical summaries, plots, and charts, enable the discovery of trends, associations, and correlations within the data.

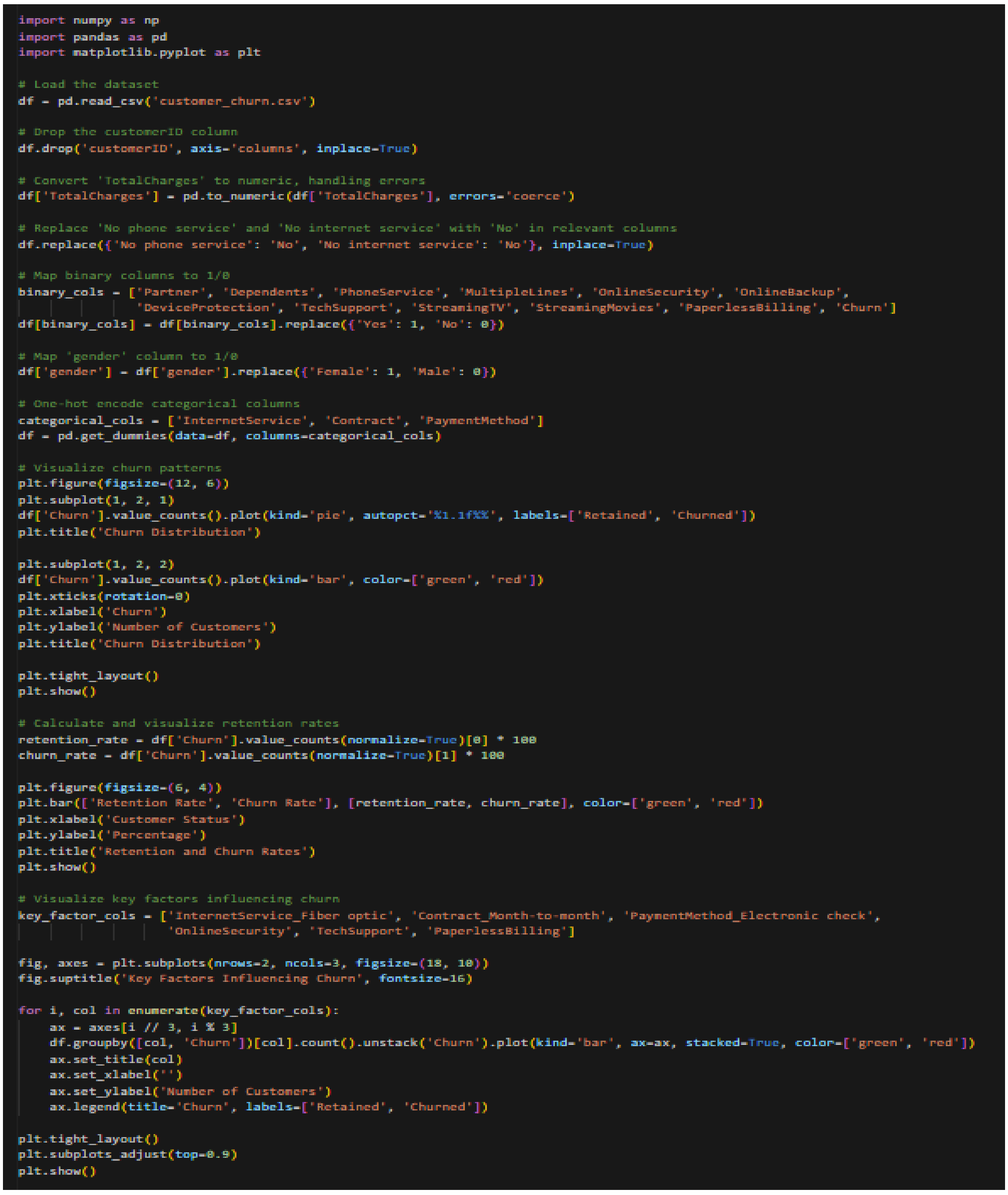
4. Formulate hypotheses: EDA often leads to the generation of hypotheses and research questions that can guide further analysis and experimentation.

Visualize churn patterns, retention rates, and key factors influencing churn

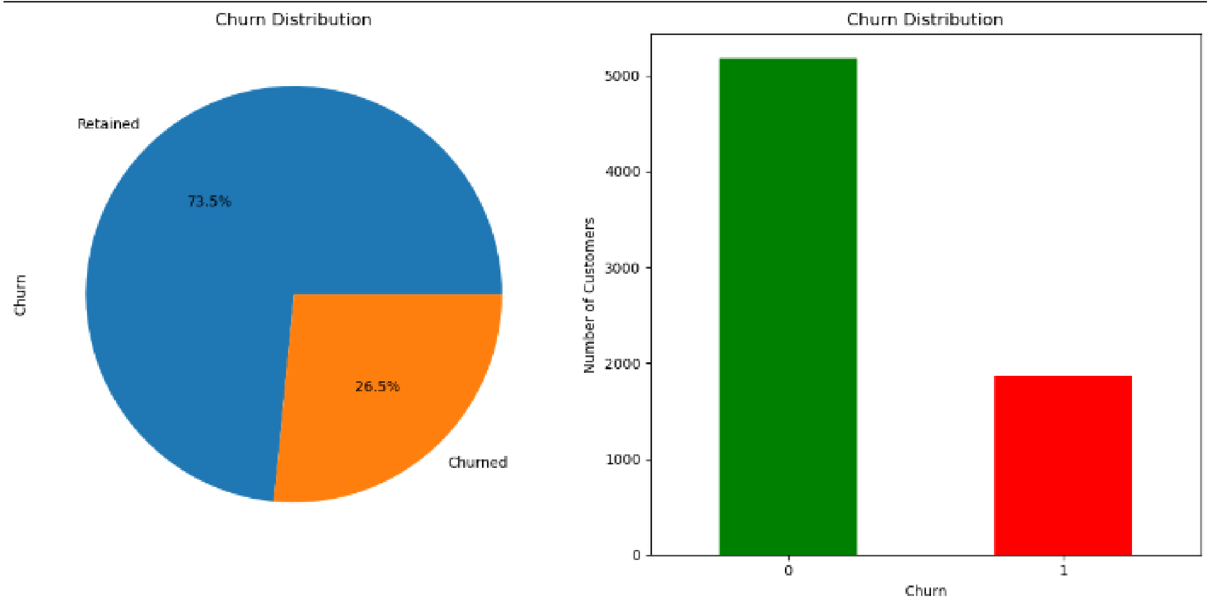


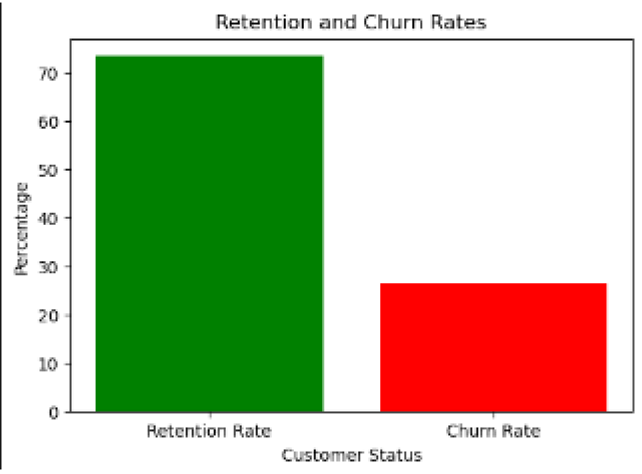
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('customer_churn.csv')

# Drop the customerID column
df.drop('customerID', axis='columns', inplace=True)

# Convert 'TotalCharges' to numeric, handling errors
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Replace 'No phone service' and 'No internet service' with 'No' in relevant columns
df.replace({'No phone service': 'No', 'No internet service': 'No'}, inplace=True)

# Map binary columns to 1/0
binary_cols = ['Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',
               'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'Churn']
df[binary_cols] = df[binary_cols].replace({'Yes': 1, 'No': 0})

# Map 'gender' column to 1/0
df['gender'] = df['gender'].replace({'Female': 1, 'Male': 0})

# One-hot encode categorical columns
categorical_cols = ['InternetService', 'Contract', 'PaymentMethod']
df = pd.get_dummies(data=df, columns=categorical_cols)

# Visualize churn patterns
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
df['Churn'].value_counts().plot(kind='pie', autopct='%1.1f%%', labels=['Retained', 'Churned'])
plt.title('Churn Distribution')

plt.subplot(1, 2, 2)
df['Churn'].value_counts().plot(kind='bar', color=['green', 'red'])
plt.xticks(rotation=0)
plt.xlabel('Churn')
plt.ylabel('Number of Customers')
plt.title('Churn Distribution')

plt.tight_layout()
plt.show()

# Calculate and visualize retention rates
retention_rate = df['Churn'].value_counts(normalize=True)['No'] * 100
churn_rate = df['Churn'].value_counts(normalize=True)['Yes'] * 100

plt.figure(figsize=(6, 4))
plt.bar(['Retention Rate', 'Churn Rate'], [retention_rate, churn_rate], color=['green', 'red'])
plt.xlabel('Customer Status')
plt.ylabel('Percentage')
plt.title('Retention and Churn Rates')
plt.show()

# Visualize key factors influencing churn
key_factor_cols = ['InternetService_Fiber optic', 'Contract_Month-to-month', 'PaymentMethod_Electronic check',
                   'OnlineSecurity', 'TechSupport', 'PaperlessBilling']

fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18, 10))
fig.suptitle('Key Factors Influencing Churn', fontsize=16)

for i, col in enumerate(key_factor_cols):
    ax = axes[i // 3, i % 3]
    df.groupby([col, 'Churn'])[col].count().unstack('Churn').plot(kind='bar', ax=ax, stacked=True, color=['green', 'red'])
    ax.set_title(col)
    ax.set_xlabel('')
    ax.set_ylabel('Number of Customers')
    ax.legend(title='Churn', labels=['No', 'Yes'])

plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.show()
```

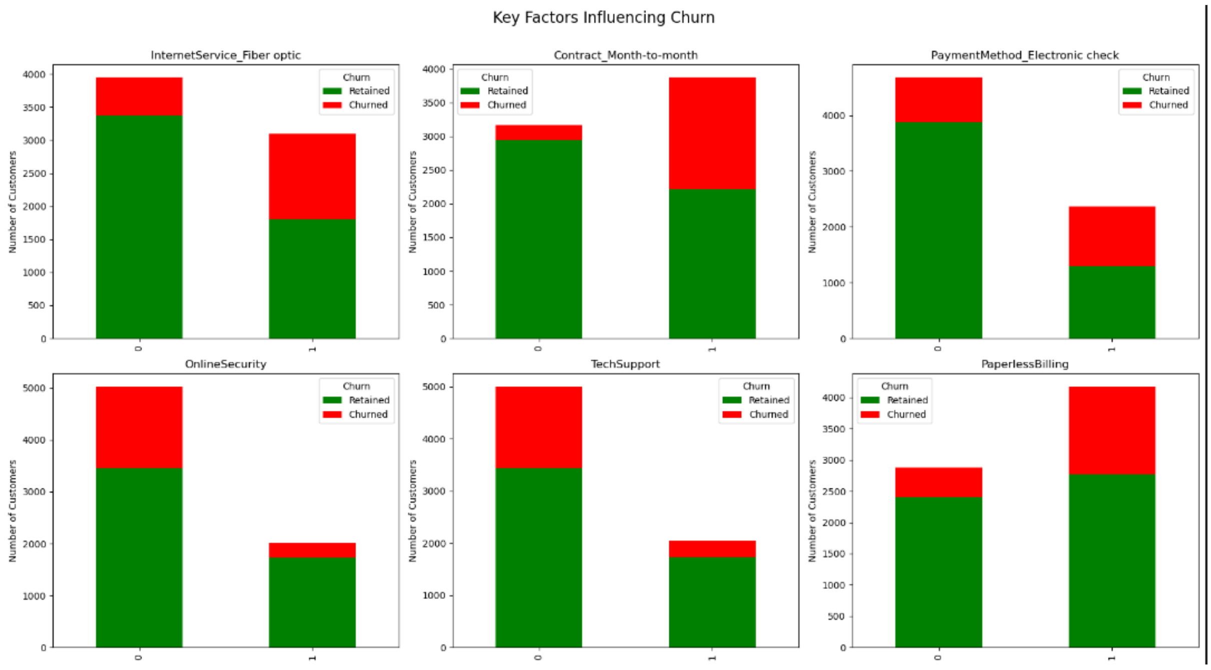




```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('customer_churn.csv')

# Drop the customerID column
df.drop('customerID', axis='columns', inplace=True)

# Convert 'TotalCharges' to numeric, handling errors
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Replace 'No phone service' and 'No internet service' with 'No' in relevant columns
df.replace({'No phone service': 'No', 'No internet service': 'No'}, inplace=True)

# Map binary columns to 1/0
binary_cols = ['Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',
               'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'Churn']
df[binary_cols] = df[binary_cols].replace({'Yes': 1, 'No': 0})

# Map 'gender' column to 1/0
df['gender'] = df['gender'].replace({'Female': 1, 'Male': 0})

# One-hot encode categorical columns
categorical_cols = ['InternetService', 'Contract', 'PaymentMethod']
df = pd.get_dummies(data=df, columns=categorical_cols)

# Visualize churn patterns
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
df['Churn'].value_counts().plot(kind='pie', autopct='%1.1f%%', labels=['Retained', 'Churned'])
plt.title('Churn Distribution')

plt.subplot(1, 2, 2)
df['Churn'].value_counts().plot(kind='bar', color=['green', 'red'])
plt.xticks(rotation=0)
plt.xlabel('Churn')
plt.ylabel('Number of Customers')
plt.title('Churn Distribution')

plt.tight_layout()
plt.show()

# Calculate and visualize retention rates
retention_rate = df['Churn'].value_counts(normalize=True)[0] * 100
churn_rate = df['Churn'].value_counts(normalize=True)[1] * 100

plt.figure(figsize=(6, 4))
plt.bar(['Retention Rate', 'Churn Rate'], [retention_rate, churn_rate], color=['green', 'red'])
plt.xlabel('Customer Status')
plt.ylabel('Percentage')
plt.title('Retention and Churn Rates')
plt.show()

# Visualize key factors influencing churn
key_factor_cols = ['InternetService_Fiber optic', 'Contract_Month-to-month', 'PaymentMethod_Electronic check',
                   'OnlineSecurity', 'TechSupport', 'PaperlessBilling']

fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18, 10))
fig.suptitle('Key Factors Influencing Churn', fontsize=16)

for i, col in enumerate(key_factor_cols):
    ax = axes[i // 3, i % 3]
    df.groupby([col, 'Churn'])[col].count().unstack('Churn').plot(kind='bar', ax=ax, stacked=True, color=['green', 'red'])
    ax.set_title(col)
    ax.set_xlabel('')
    ax.set_ylabel('Number of Customers')
    ax.legend(title='Churn', labels=['Retained', 'Churned'])

plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.show()
```


Churn Distribution
Retained
73.5%
26.5%
Churned
Churn
Churn Distribution
Number of Customers
0
1000
2000
3000
4000
5000
0
1
Churn


Retention and Churn Rates
Percentage
0
10
20
30
40
50
60
70
Retention Rate
Churn Rate
Customer Status


Key Factors Influencing Churn
InternetService_Fiber optic
Contract_Month-to-month
PaymentMethod_Electronic check
OnlineSecurity
TechSupport
PaperlessBilling
Churn
🟩 Retained
🟥 Churned
Number of Customers
0
1

Training Model:

1.Data Splitting:

Split your pre-processed dataset into training and testing sets. This allows you to train the model on one subset of the data and evaluate its performance on another subset. You can use libraries like scikit-learn to perform the data splitting.

```python
from sklearn.model_selection import train_test_split

# Split the data into features (X) and target (y)
X = df.drop('Churn', axis=1)
y = df['Churn']


# Split the data into training and testing sets (e.g., 80% for training, 20% for testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2.Model Selection:

Choose a machine learning model that is suitable for your classification task. Common models for binary classification (such as churn prediction) include Logistic Regression, Random Forest, Gradient Boosting, Support Vector Machines, and more. You can start with a simple model and then explore more complex models to compare their performance.

```python
from sklearn.ensemble import RandomForestClassifier

# Create and train the model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

3.Model Evaluation:

Evaluate the model's performance on the testing dataset. Common evaluation metrics for classification tasks include accuracy, precision, recall, F1-score, and the confusion matrix.

```python
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# Make predictions on the testing data
y_pred = model.predict(X_test)


# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)


# Generate a classification report
print(classification_report(y_test, y_pred))


# Generate a confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
```