TASK NO 4: Predicting Customer Churn in a Telecom Company

Data loading Subtask:

Load the dataset "somuprodata1.csv" into a pandas DataFrame.

- CODE:
- import pandas as pd

df = pd.read_csv('somuprodata1.csv') display(df.head())

Data Preprocessing: Clean the data, handle missing values, and encodecategorical features. **Data exploration** Subtask:
Explore the loaded dataset df.
Reasoning: Explore the dataset by finding its shape, data types, descriptive statistics, target variable distribution, missing values, and correlation with the target variable.
CODE:

```
# 1. Shape of the data
print("Shape of the data:", df.shape)

# 2. Data types of each column
print("\nData types of each column:\n", df.dtypes)

# 3. Descriptive statistics for numerical features
print("\nDescriptive statistics for numerical features:\n", df.describe())

# 4. Distribution of the target variable (churn)
print("\nDistribution of the target variable (churn):\n", df['Churn'].value_counts())

# 5. Missing values in each column
```

print("\nMissing values in each column:\n", df.isnull().sum())

Data cleaning Subtask:

Clean the data in the DataFrame df.

Reasoning: Identify and handle missing values, convert 'TotalCharges' to numeric, and handle duplicates.

CODE:

```
# Handle missing values
df.dropna(inplace=True)

# Convert 'TotalCharges' to numeric
df['TotalCharges']=pd.to_numeric(df['TotalCharges'], errors='coerce')
df.dropna(subset=['TotalCharges'], inplace=True)

# Handle duplicate rows
df.drop_duplicates(inplace=True)
```

Data wrangling Subtask:

Encode categorical features in the DataFrame df.

Reasoning: Identify categorical features and encode them using one-hot encoding or label encoding based on the number of unique values and the presence of a natural order.

CODE:

```
from sklearn.preprocessing import LabelEncoder

# Identify categorical features
categorical_features =
df.select_dtypes(include=['object']).columns.tolist()

# Encode categorical features for
feature in categorical_features:   if
df[feature].nunique() <= 10:  #
Consider one-hot encoding for
features with fewer unique values
```

```
    df = pd.get_dummies(df, columns=[feature], prefix=[feature])   else:
# Consider label encoding for features with many unique values
    le = LabelEncoder()
    df[feature] = le.fit_transform(df[feature])
```

EDA: Visualize relationships between features and churn using barcharts,histograms,and correlation matrices. **Data visualization**
Subtask:
Visualize the relationships between features and churn using histograms, bar charts, and correlation matrices.
Reasoning: Generate histograms for numerical features, comparing churned and non-churned customers.
CODE:

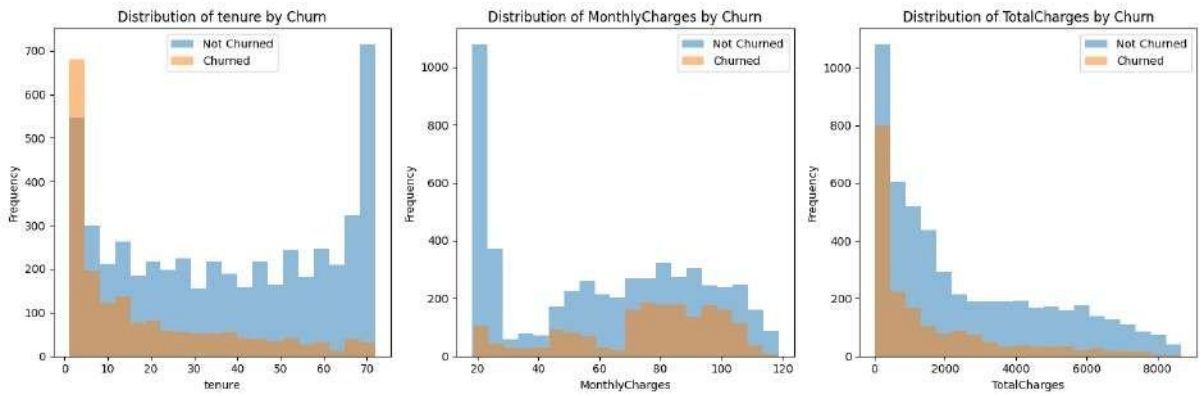
```
import matplotlib.pyplot as plt

# Histograms for numerical features, comparing churned and
nonchurned customers
numerical_features = ['tenure', 'MonthlyCharges', 'TotalCharges']

plt.figure(figsize=(15, 5))

for i, feature in enumerate(numerical_features):   plt.subplot(1,
3, i + 1)
        plt.hist(df[df['Churn'] ==    0][feature],    bins=20,    alpha=0.5,
label='Not Churned')
        plt.hist(df[df['Churn'] ==    1][feature],    bins=20,    alpha=0.5,
label='Churned')
plt.xlabel(feature)
  plt.ylabel('Frequency')
  plt.title(f'Distribution of {feature} by Churn')   plt.legend()

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



Reasoning: Generate bar charts for categorical features to understand their relationship with churn.

CODE:

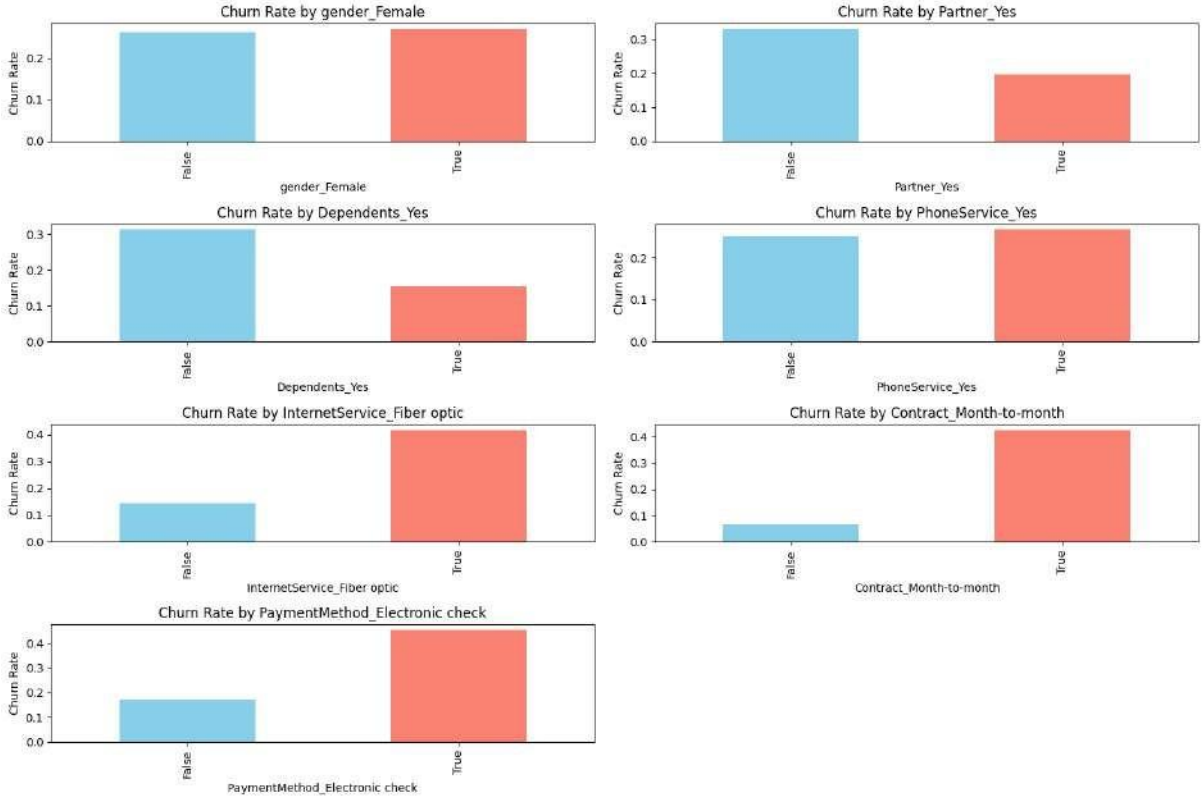
```
import matplotlib.pyplot as plt

categorical_features = ['gender_Female', 'Partner_Yes', 'Dependents_Yes', 'PhoneService_Yes', 'InternetService_Fiber optic', 'Contract_Month-to-month', 'PaymentMethod_Electronic check']

plt.figure(figsize=(15, 10))

for i, feature in enumerate(categorical_features):   plt.subplot(4, 2, i + 1)
  churn_rates = df.groupby(feature)['Churn'].mean()
churn_rates.plot(kind='bar', color=['skyblue', 'salmon'])
plt.xlabel(feature)   plt.ylabel('Churn Rate')
  plt.title(f'Churn Rate by {feature}')
```

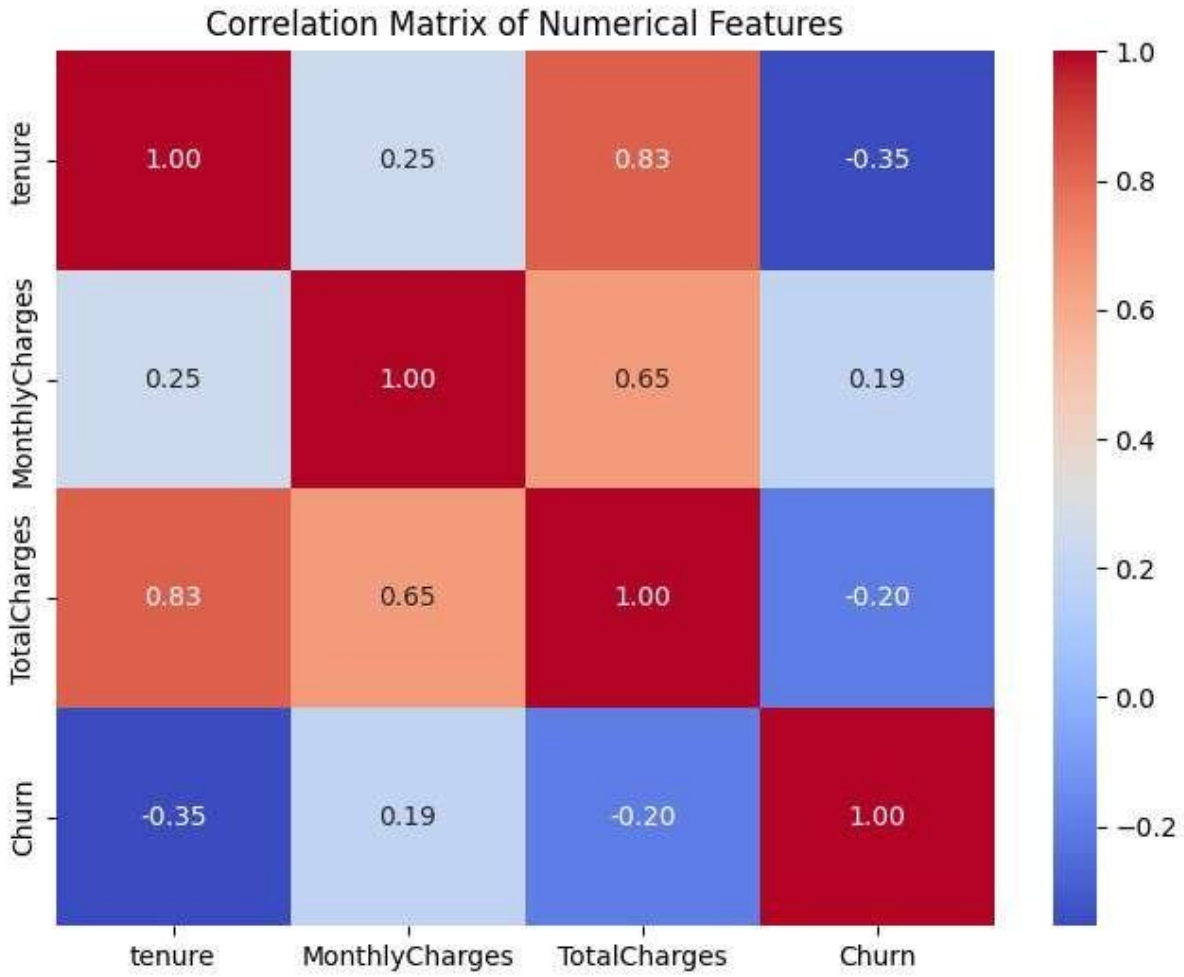
plt.tight_layout() plt.show()



Reasoning: Generate a correlation matrix for numerical features and visualize it using a heatmap.

CODE:

```
import seaborn as sns import
matplotlib.pyplot as plt
# Correlation matrix for numerical features
numerical_features = ['tenure', 'MonthlyCharges', 'TotalCharges', '
Churn']
correlation_matrix = df[numerical_features].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', f
mt=".2f")
plt.title('Correlation Matrix of Numerical Features') plt.show()
```



Model Building: Use Decision Trees for classification tasks.

Data splitting Subtask:

Split the data into training and testing sets.

Reasoning: Split the data into training and testing sets based on the instructions.

CODE:

```
from sklearn.model_selection import train_test_split

# Define the target variable (y) and feature set (X) y =
df['Churn']
X = df.drop('Churn', axis=1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)
```

Model training Subtask:
Train a Decision Tree Classifier model on the training data. **Reasoning**: Train a Decision Tree Classifier model on the training data using X_train and y_train.
CODE:

```
from sklearn.tree import DecisionTreeClassifier

# Instantiate a DecisionTreeClassifier object
model = DecisionTreeClassifier(random_state=42)

# Fit the model to the training data model.fit(X_train,
y_train)
```

Reasoning: Explore the trained model by printing its attributes like feature_importances_.
CODE:

```
# Print feature importances print("Feature importances:",
model.feature_importances_)
```

Evaluation: Use accuracy, confusion matrix, precision, recall, and F1 score. **Model evaluation** Subtask:
Evaluate the trained Decision Tree model using accuracy, confusion matrix, precision, recall, and F1 score.
Reasoning: Evaluate the trained Decision Tree model using accuracy, confusion matrix, precision, recall, and F1 score CODE:

```
from sklearn.metrics import accuracy_score, confusion_matrix,
precision_score, recall_score, f1_score

# Predict the target variable for the test data y_pred =
model.predict(X_test)

# Calculate and print the accuracy accuracy =
accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
# Generate and print the confusion matrix cm =
confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

# Calculate and print precision, recall, and F1 score
precision = precision_score(y_test, y_pred) recall =
recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Precision:", precision)
print("Recall:", recall) print("F1
Score:", f1)
```

Deployment: Build a customer churn prediction system to identify at-risk customers

1. Create a Flask app

First, you need to create a Flask application that will serve your model. Here's a basic structure for a Flask app called app.py: from flask import Flask, request, jsonify import pandas as pd
import pickle

```
app = Flask(__name__)

# Load the trained model with
open('  somuprodata1.csv’
, 'rb') as f:
    model = pickle.load(f)

@app.route('/predict', methods=['POST'])
def predict():     data = request.get_json()
df = pd.DataFrame([data])
```