**Project Title: Customer Churn Analysis and Prediction**

**Task1: Data Cleaning and Preprocessing**

import pandas as pd

# Load the dataset from the Excel file

df = pd.read\_excel('TelcoCustomerChurn.xlsx')

# Step 1: Check for missing values

print("Missing values per column before imputation:")

print(df.isnull().sum())

# Step 2: Handle missing values

# Identify numerical and categorical columns

num\_cols = df.select\_dtypes(include=['int64', 'float64']).columns

cat\_cols = df.select\_dtypes(include=['object']).columns

# For numerical columns, fill missing values with the median value

for col in num\_cols:

df[col].fillna(df[col].median(), inplace=True)

# For categorical columns, fill missing values with the mode (most frequent value)

for col in cat\_cols:

df[col].fillna(df[col].mode()[0], inplace=True)

# Verify that missing values have been handled

print("\nMissing values per column after imputation:")

print(df.isnull().sum())

# Step 3: Convert categorical variables to numerical using one-hot encoding

# The 'drop\_first=True' parameter helps to avoid the dummy variable trap (reducing multicollinearity)

df\_encoded = pd.get\_dummies(df, columns=cat\_cols, drop\_first=True)

# Display the first few rows of the processed DataFrame

print("\nData after one-hot encoding:")

print(df\_encoded.head())

**Tasks 2: Exploratory Data Analysis**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset (Assuming it's preprocessed as needed)

df = pd.read\_excel('TelcoCustomerChurn.xlsx')

# ----------------------------

# 1. Calculate and Visualize Overall Churn Rate

# ----------------------------

# Calculate churn counts and percentages

churn\_counts = df['Churn'].value\_counts()

print("Churn Counts:")

print(churn\_counts)

churn\_rate = churn\_counts / df.shape[0] \* 100

print("\nChurn Rate (%):")

print(churn\_rate)

# Pie Chart for overall churn distribution

plt.figure(figsize=(8,6))

plt.pie(churn\_counts, labels=churn\_counts.index, autopct='%1.1f%%', colors=sns.color\_palette('pastel'))

plt.title('Overall Churn Distribution')

plt.show()

# Bar Plot for churn counts

plt.figure(figsize=(8,6))

sns.countplot(data=df, x='Churn', palette='viridis')

plt.title('Count of Churned vs. Non-Churned Customers')

plt.show()

# ----------------------------

# 2. Explore Customer Distribution by Various Demographics

# ----------------------------

# Example demographics: Gender, SeniorCitizen, Partner, Dependents

demographic\_cols = ['gender', 'SeniorCitizen', 'Partner', 'Dependents']

for col in demographic\_cols:

plt.figure(figsize=(8,6))

sns.countplot(data=df, x=col, palette='Set2')

plt.title(f'Customer Distribution by {col}')

plt.xlabel(col)

plt.ylabel('Count')

plt.show()

# ----------------------------

# 3. Analyze Tenure Distribution

# ----------------------------

# Histogram with KDE for tenure distribution

plt.figure(figsize=(10,6))

sns.histplot(df['tenure'], kde=True, bins=30, color='skyblue')

plt.title('Distribution of Customer Tenure')

plt.xlabel('Tenure (Months)')

plt.ylabel('Count')

plt.show()

# Boxplot for tenure to visualize outliers and spread

plt.figure(figsize=(10,6))

sns.boxplot(x=df['tenure'], color='lightgreen')

plt.title('Boxplot of Customer Tenure')

plt.xlabel('Tenure (Months)')

plt.show()

# ----------------------------

# 4. Investigate Relationships Between Churn and Contract Types/Payment Methods

# ----------------------------

# Relationship between churn and contract types

plt.figure(figsize=(10,6))

sns.countplot(data=df, x='Contract', hue='Churn', palette='coolwarm')

plt.title('Churn Count by Contract Type')

plt.xlabel('Contract Type')

plt.ylabel('Count')

plt.show()

# Relationship between churn and payment methods

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

sns.countplot(data=df, x='PaymentMethod', hue='Churn', palette='magma')

plt.title('Churn Count by Payment Method')

plt.xlabel('Payment Method')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()

**Task 3: Customer Segmentation**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset (assuming preprocessing is complete)

df = pd.read\_excel('TelcoCustomerChurn.xlsx')

# ----------------------------

# 1. Create Customer Segments

# ----------------------------

# a. Segment based on Tenure

# Define tenure bins (e.g., 0-12, 13-24, 25-48, and 49-72 months)

tenure\_bins = [0, 12, 24, 48, 72]

tenure\_labels = ['0-12', '13-24', '25-48', '49-72']

df['tenure\_group'] = pd.cut(df['tenure'], bins=tenure\_bins, labels=tenure\_labels, right=True)

# b. Segment based on Monthly Charges

# Create bins using quantiles for a balanced segmentation

df['monthly\_charge\_group'] = pd.qcut(df['MonthlyCharges'], q=4, labels=['Low', 'Medium', 'High', 'Very High'])

# c. Contract Type is already categorical, so we will use it as is

# Ensure that the 'Contract' column is of type string (if needed)

df['Contract'] = df['Contract'].astype(str)

# ----------------------------

# 2. Analyze Churn Rates Across Segments

# ----------------------------

# Compute churn rate within each segment grouping.

# Assuming the 'Churn' column contains values 'Yes' and 'No'

churn\_rates = df.groupby(['tenure\_group', 'monthly\_charge\_group', 'Contract'])['Churn'] \

.apply(lambda x: (x == 'Yes').mean()) \

.reset\_index(name='ChurnRate')

print("Churn Rates across Customer Segments:")

print(churn\_rates)

# ----------------------------

# 3. Visualize the Segmentation Analysis

# ----------------------------

# Example: Create a heatmap for churn rates within a specific contract type.

# Here we filter for a contract type (e.g., 'Month-to-month')

contract\_type = 'Month-to-month'

subset = churn\_rates[churn\_rates['Contract'] == contract\_type]

# Pivot the data to prepare for heatmap visualization: tenure groups vs. monthly charge groups

pivot\_table = subset.pivot(index='tenure\_group', columns='monthly\_charge\_group', values='ChurnRate')

plt.figure(figsize=(10, 6))

sns.heatmap(pivot\_table, annot=True, cmap="YlGnBu", fmt=".2f")

plt.title(f'Churn Rate Heatmap for {contract\_type} Contract')

plt.xlabel('Monthly Charges Group')

plt.ylabel('Tenure Group')

plt.show()

**Task 4: Churn Prediction Model**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Import machine learning modules

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_auc\_score, roc\_curve

# ----------------------------

# 1. Load and Preprocess the Data

# ----------------------------

# Load the dataset (Assuming you've preprocessed missing values and applied one-hot encoding)

df = pd.read\_excel('TelcoCustomerChurn.xlsx')

# Convert target variable 'Churn' from categorical ('Yes'/'No') to numerical (1/0)

df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})

# (Optional) If your dataset hasn't been fully preprocessed, you may need to handle categorical variables:

# df = pd.get\_dummies(df, drop\_first=True)

# Define features and target variable

X = df.drop('Churn', axis=1)

y = df['Churn']

# ----------------------------

# 2. Split the Data into Training and Testing Sets

# ----------------------------

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# ----------------------------

# 3. Develop the Logistic Regression Model

# ----------------------------

# Initialize and train the logistic regression model

lr\_model = LogisticRegression(max\_iter=1000) # Increase max\_iter if needed

lr\_model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = lr\_model.predict(X\_test)

y\_pred\_prob = lr\_model.predict\_proba(X\_test)[:, 1]

# ----------------------------

# 4. Evaluate the Model's Performance

# ----------------------------

# Calculate evaluation metrics

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

class\_report = classification\_report(y\_test, y\_pred)

roc\_auc = roc\_auc\_score(y\_test, y\_pred\_prob)

print("Accuracy:", accuracy)

print("\nConfusion Matrix:\n", conf\_matrix)

print("\nClassification Report:\n", class\_report)

print("\nROC AUC Score:", roc\_auc)

# Plot the confusion matrix

plt.figure(figsize=(6, 4))

sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap='Blues')

plt.title('Confusion Matrix')

plt.ylabel('Actual Label')

plt.xlabel('Predicted Label')

plt.show()

# Plot ROC curve

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_prob)

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='gray')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve for Logistic Regression')

plt.legend(loc='best')

plt.show()