import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

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

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

data = pd.read\_csv('data\_mobile\_price\_range.csv')

# Print the first few rows of the dataset

print("Full Dataset:")

print(data.head())

# Split the data into features (X) and target (y)

X = data.drop('price\_range', axis=1)

y = data['price\_range']

# 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)

# Print the training and testing datasets

print("\nTraining Set Features:")

print(X\_train.head())

print("\nTraining Set Labels:")

print(y\_train.head())

print("\nTesting Set Features:")

print(X\_test.head())

print("\nTesting Set Labels:")

print(y\_test.head())

# Predictions on the training set

y\_train\_pred = rf\_classifier.predict(X\_train)

# Predictions on the test set

y\_test\_pred = rf\_classifier.predict(X\_test)

# Plot regression graph for training set

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

sns.regplot(x=y\_train, y=y\_train\_pred, scatter\_kws={'alpha':0.5})

plt.xlabel('Actual Price Range')

plt.ylabel('Predicted Price Range')

plt.title('Regression Plot - Training Set')

plt.grid(True)

plt.show()

# Plot regression graph for test set

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

sns.regplot(x=y\_test, y=y\_test\_pred, scatter\_kws={'alpha':0.5})

plt.xlabel('Actual Price Range')

plt.ylabel('Predicted Price Range')

plt.title('Regression Plot - Test Set')

plt.grid(True)

plt.show()

# Initialize the Random Forest Classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the model

rf\_classifier.fit(X\_train, y\_train)

# Predictions on the test set

y\_pred = rf\_classifier.predict(X\_test)

# Evaluate the model

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

print('\nAccuracy:', accuracy)

# Print classification report

print(classification\_report(y\_test, y\_pred))

# Compute confusion matrix

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

print('Confusion Matrix:')

print(conf\_matrix)

# Function to get user input

def get\_user\_input():

    features = {}

    for column in X.columns:

        value = input(f"Enter {column}: ")

        features[column] = [float(value)]  # assuming all values are numeric

    return pd.DataFrame(features)

# Get user input

user\_input = get\_user\_input()

# Make prediction using the trained model

predicted\_price\_range = rf\_classifier.predict(user\_input)

print("\nPredicted Price Range:", predicted\_price\_range)