**3.Write a Python script that trains a simple linear regression model using scikit-learn. Use a dataset of your choice, split it into training and testing sets, and evaluate the model's performance.**

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.datasets import fetch\_california\_housing

# Fetch the California housing dataset

housing = fetch\_california\_housing()

# Convert to pandas DataFrame

data = pd.DataFrame(housing.data, columns=housing.feature\_names)

data['PRICE'] = housing.target

# Display the first few rows of the dataset

print("First few rows of the California housing dataset:")

print(data.head())

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

X = data.drop(columns='PRICE')

y = data['PRICE']

# Split the dataset into training and testing sets (80% train, 20% test)

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

# Create a Linear Regression model

model = LinearRegression()

# Train the model using the training data

model.fit(X\_train, y\_train)

# Make predictions using the testing data

y\_pred = model.predict(X\_test)

# Evaluate the model's performance

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

# Display

print("\nModel Performance:")

print(f"Mean Squared Error: {mse:.2f}")

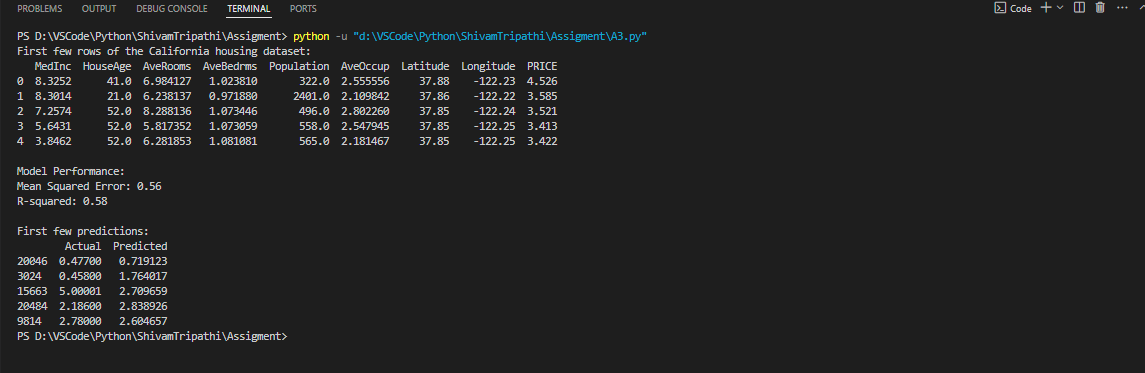
print(f"R-squared: {r2:.2f}")

results = pd.DataFrame({'Actual': y\_test, 'Predicted': y\_pred})

print("\nFirst few predictions:")

print(results.head())

**OUTPUT:**

****