**Question 1**

Implement linear regression to predict student score using the following dataset.

* Use student\_scores dataset
* Use packages: NumPy, pandas, scikit-learn.
* Preprocess the dataset.
* Train and evaluate a linear regression model.
* Visualize the results.

**Source Code**

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler

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

import matplotlib.pyplot as plt

dataset = pd.read\_csv('Week 2\student\_scores.csv', header=0)

print(dataset.head())

scaler = StandardScaler()

dataset[['Hours', 'Scores']] = scaler.fit\_transform(dataset[['Hours', 'Scores']])

X = dataset[['Hours']]

y = dataset[['Scores']]

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

model = LinearRegression()

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

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

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print('RMSE Score: ', rmse)

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

print("R2 Score: ", r2)

X\_test\_orig = scaler.inverse\_transform(np.hstack([X\_test, np.zeros\_like(X\_test)]))[:, 0]

y\_test\_orig = scaler.inverse\_transform(np.hstack([np.zeros\_like(y\_test), y\_test]))[:, 1]

y\_pred\_orig = scaler.inverse\_transform(np.hstack([np.zeros\_like(y\_pred), y\_pred]))[:, 1]

plt.scatter(X\_test\_orig, y\_test\_orig, color='blue', label='Actual')

plt.plot(X\_test\_orig, y\_pred\_orig, color='red', label='Predicted')

plt.xlabel('Study Hours')

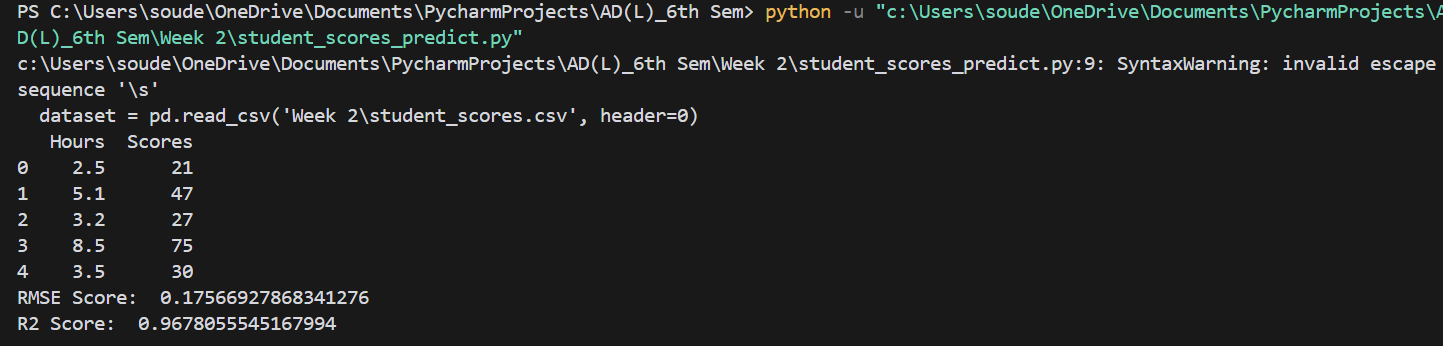
plt.ylabel('Scores')

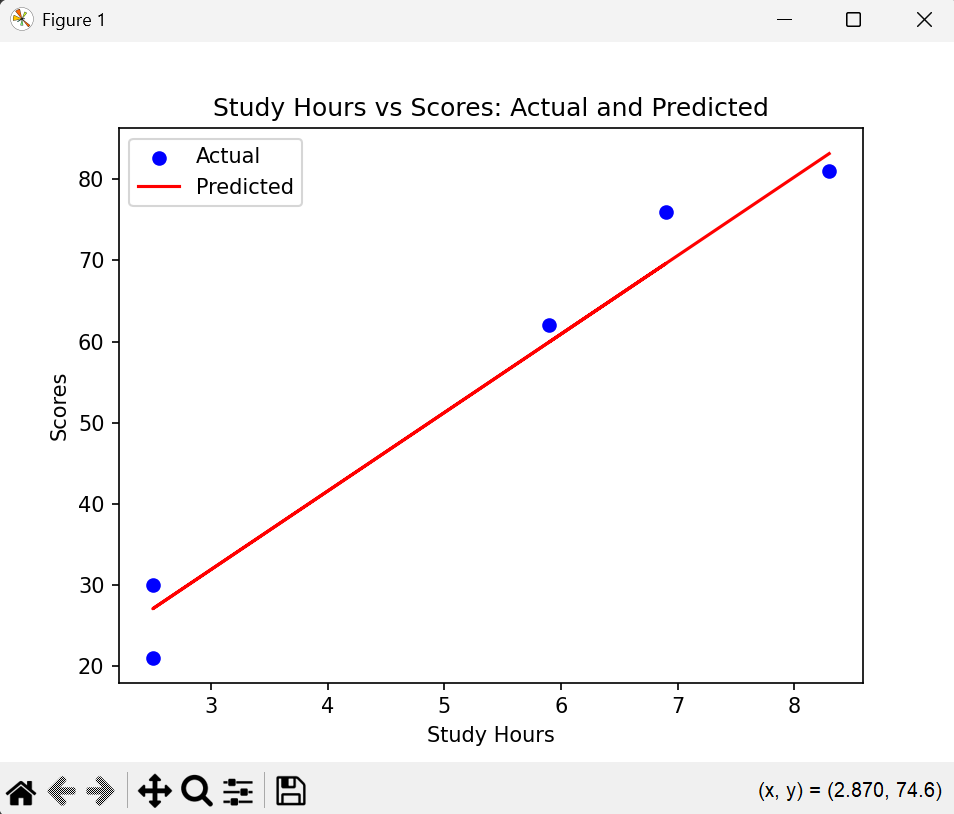
plt.title('Study Hours vs Scores: Actual and Predicted')

plt.legend()

plt.show()

**Output**

*Terminal*

*Study Hours vs Scores Regression Plot*

**Question 2**

Implement linear regression to predict weight using the following dataset.

* Use height-weight dataset
* Use packages: NumPy, pandas, scikit-learn.
* Preprocess the dataset.
* Train and evaluate a linear regression model.
* Visualize the results.

**Source Code**

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler

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

import matplotlib.pyplot as plt

import seaborn as sns

dataset = pd.read\_csv('Week 2\height-weight.csv', header=0)

print(dataset.head(5))

dataset\_enc = pd.get\_dummies(dataset, columns=['Gender'], drop\_first=True)

scaler = StandardScaler()

dataset\_enc[['Height', 'Weight']] = scaler.fit\_transform(dataset\_enc[['Height', 'Weight']])

print(dataset\_enc)

X = dataset\_enc[['Height', 'Gender\_Male']]

y = dataset\_enc[['Weight']]

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

model = LinearRegression()

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

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

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print('RMSE Score: ', rmse)

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

print("R2 Score: ", r2)

plt.figure(figsize=(6, 5))

plt.scatter(y\_test, y\_pred, alpha=0.5)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--', lw=2)

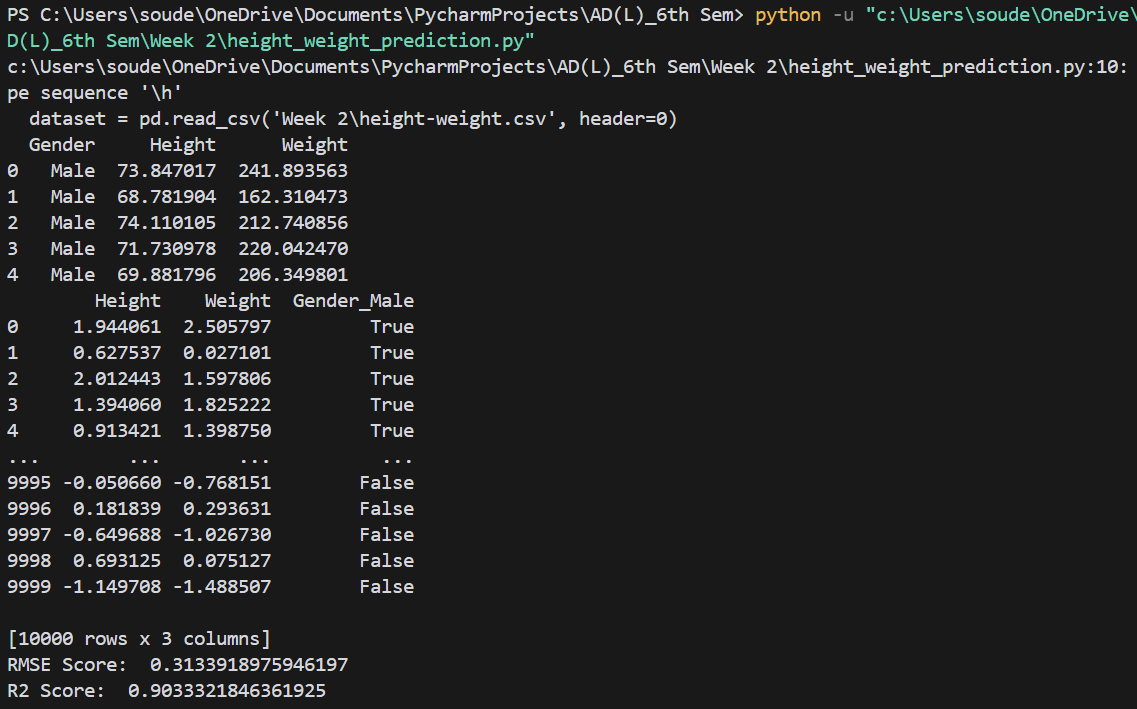
plt.xlabel('Actual Weight (Standardized)')

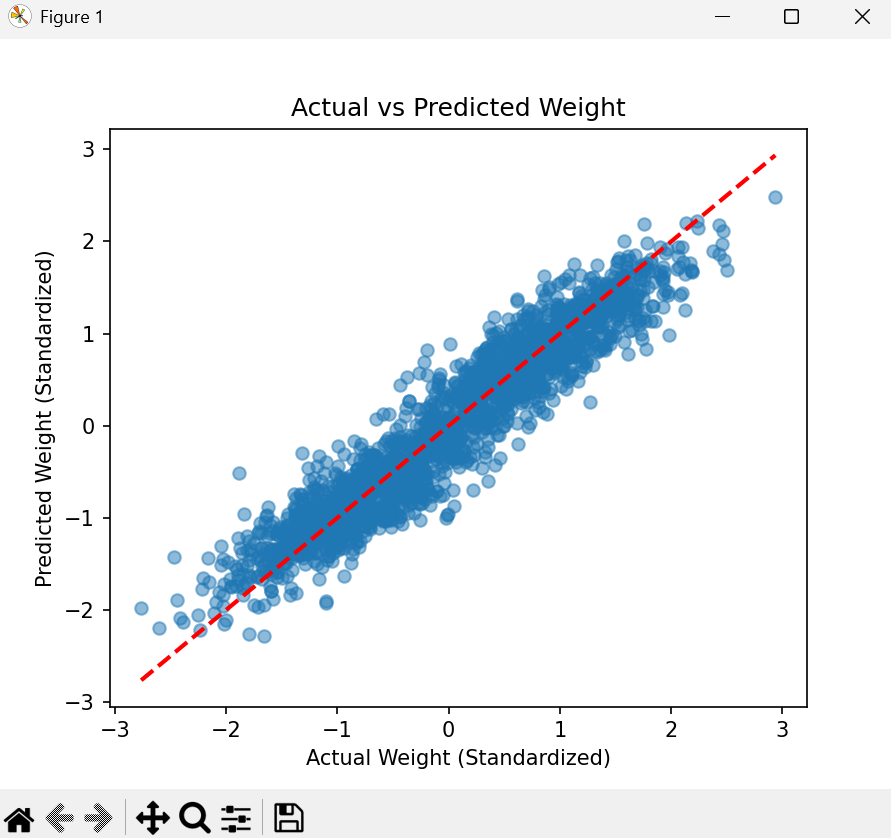
plt.ylabel('Predicted Weight (Standardized)')

plt.title('Actual vs Predicted Weight')

plt.show()

**Output**

*Terminal*

*Actual vs Predicted Weight Regression Line*

**Question 3**

Implement linear regression to predict house price using the following dataset.

* Use housing dataset
* Use packages: NumPy, pandas, scikit-learn.
* Preprocess the dataset.
* Train and evaluate a linear regression model.
* Visualize the results.

**Source Code**

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler

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

import matplotlib.pyplot as plt

import seaborn as sns

dataset = pd.read\_csv('Week 2\Housing.csv', header=0)

print(dataset.head())

categorical\_cols = ['mainroad', 'guestroom', 'basement', 'hotwaterheating',

'airconditioning', 'prefarea', 'furnishingstatus']

dataset\_encoded = pd.get\_dummies(dataset, columns=categorical\_cols, drop\_first=True)

print(dataset\_encoded.head())

scaler = StandardScaler()

dataset\_encoded[['area']] = scaler.fit\_transform(dataset\_encoded[['area']])

X = dataset\_encoded.drop('price', axis=1)

y = dataset\_encoded['price']

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

model = LinearRegression()

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

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

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

min\_price = y.min()

max\_price = y.max()

normalized\_rmse = abs((rmse - min\_price) / (max\_price - min\_price))

print('RMSE Score: ', normalized\_rmse)

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

print("R2 Score: ", r2)

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

plt.scatter(y\_test, y\_pred, alpha=0.5)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--', lw=2)

plt.xlabel('Actual Prices')

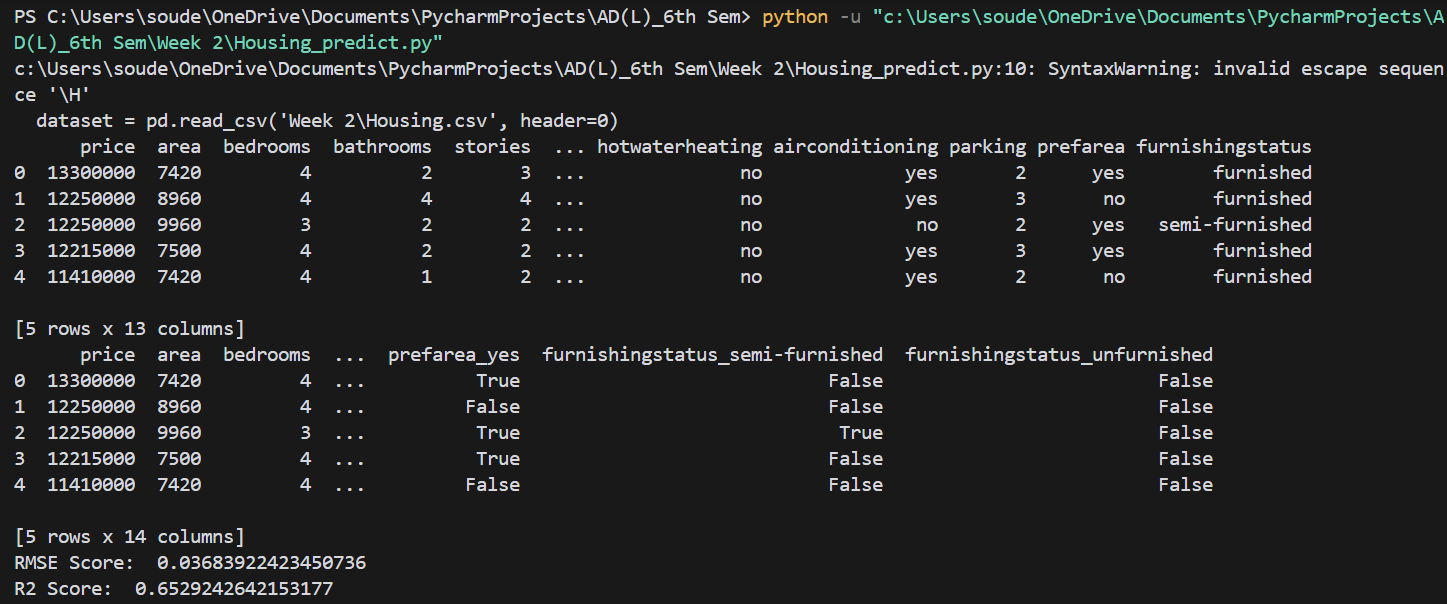
plt.ylabel('Predicted Prices')

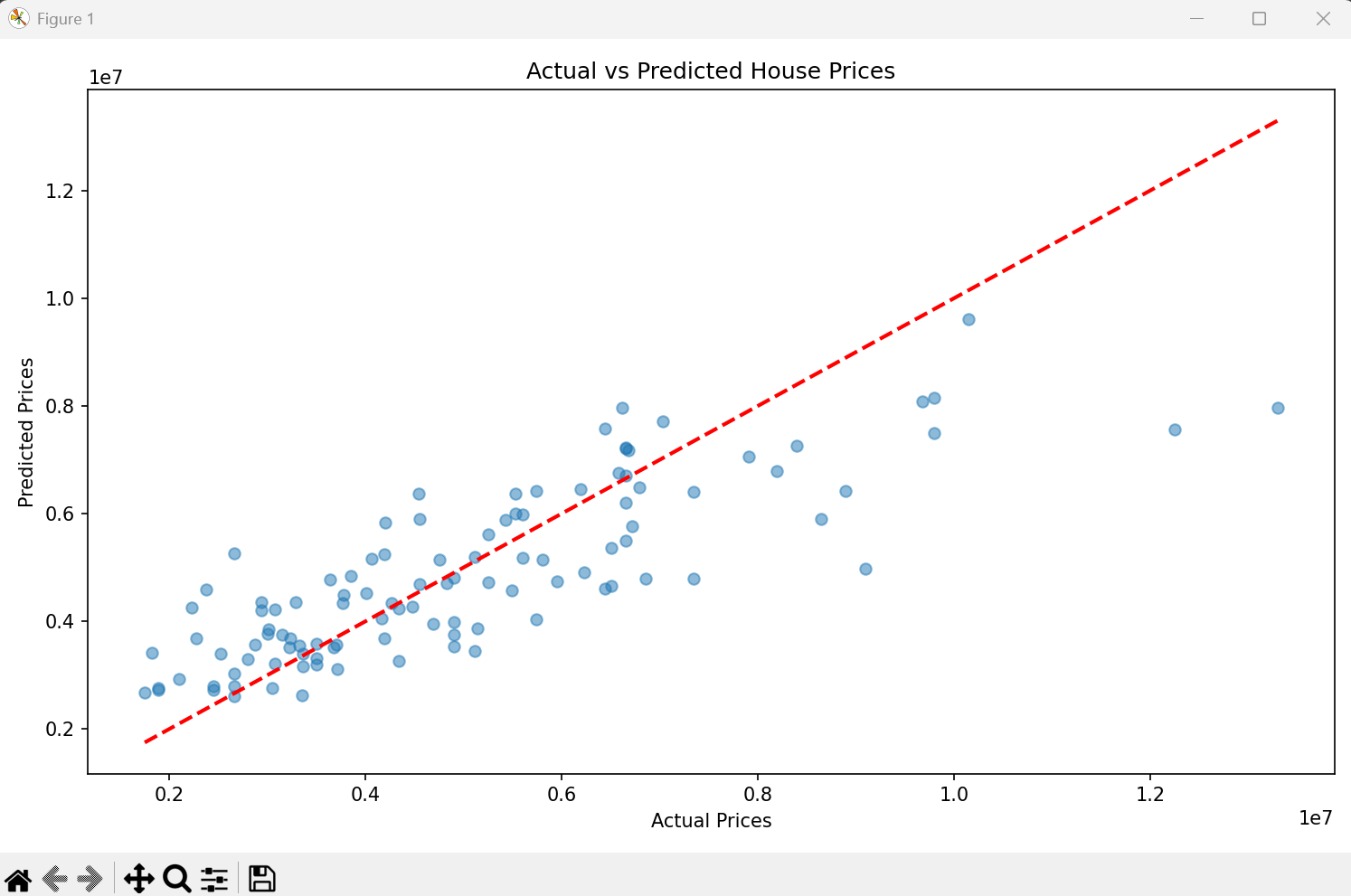
plt.title('Actual vs Predicted House Prices')

plt.tight\_layout()

plt.show()

**Output**

*Terminal*

**

*Actual vs Predicted House Prices Regression Line*