CSBB311: MACHINE LEARNING LAB

ASSIGNMENT 4 :- Linear Regression

Submitted By:

Name: Kartik Mittal

Roll No: 221210056

Branch: CSE

Semester: 5th Sem

Group : 2

Submitted To: Dr. Preeti Mehta

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY DELHI

2024

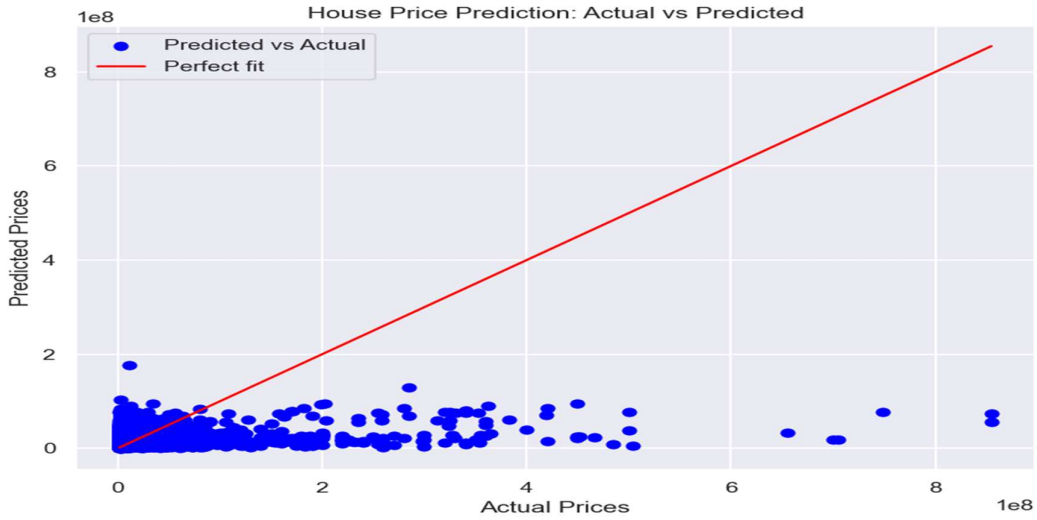
Code (Using Inbuilt Library):-

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load data from your house prediction CSV
data = pd.read_csv('multiple.csv')  # Replace with your actual dataset path

# Print the first few rows to understand the dataset structure
print(data.head())

# One-hot encode the 'City' categorical column
data_encoded = pd.get_dummies(data, columns=['City'], drop_first=True)

# Assuming your dataset has features like 'Area', 'Bedrooms', and one-hot encoded 'City'
X = data_encoded[['Area', 'Bedrooms'] + [col for col in data_encoded.columns if 'City_' in col]].valu
y = data['Price'].values  # Dependent variable (target - house prices)

# Create and fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# Make predictions
predictions = model.predict(X)
```

```python
mse = mean_squared_error(y, predictions)
r2 = r2_score(y, predictions)

# Print the coefficients (slopes) and intercept
print("Coefficients (slopes):", model.coef_)
print("Intercept:", model.intercept_)

# Print evaluation metrics
print(f"Mean Squared Error (MSE): {mse:.3f}")
print(f"R-squared (R2 Score): {r2:.3f}")

# Display accuracy in percentage
accuracy = r2 * 100
print(f"Accuracy (R2 Score as percentage): {accuracy:.2f}%")

# Visualize the relationship between actual and predicted values
plt.figure(figsize=(8, 6))
sns.set(style="darkgrid")

# Scatter plot for actual vs predicted
plt.scatter(y, predictions, color='blue', label='Predicted vs Actual')

# Line representing perfect prediction
plt.plot([min(y), max(y)], [min(y), max(y)], color='red', label='Perfect fit')

# Adding labels and title
plt.title('House Price Prediction: Actual vs Predicted')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
```

Output :-

```
   Sample       City     Price  Area                          Location  Bedrooms
0       0  Bangalore  30000000  3340                   JP Nagar Phase 1         4
1       1  Bangalore   7888000  1045         Dasarahalli on Tumkur Road         2
2       2  Bangalore   4866000  1179  Kannur on Thanisandra Main Road          2
3       3  Bangalore   8358000  1675                        Doddanekundi         3
4       4  Bangalore   6845000  1670                             Kengeri         3
Coefficients (slopes): [   10620.2823423  -1544523.55545918   872187.83118101  9487752.36061403
 -2000843.71414889  -418916.08595631  9130004.92201851]
Intercept: -1543250.1390244085
Mean Squared Error (MSE): 506582389491399.875
R-squared (R2 Score): 0.111
Accuracy (R2 Score as percentage): 11.14%
PS C:\Users\HP\Desktop\college\semester 5\Machine Learning\lab4 regression>
```



Code (Without Using Inbuilt Library):-

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error, r2_score

# Linear Regression class
class MultipleLinearRegression:
    def __init__(self):
        self.coefficients = None

    # Method to fit the model
    def fit(self, X, y):
        # Add a column of ones to X to account for the intercept (b0)
        ones = np.ones((X.shape[0], 1))
        X_b = np.hstack((ones, X))

        # Calculate coefficients using the normal equation: (X^T X)^-1 X^T y
        self.coefficients = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)

    # Method to make predictions
    def predict(self, X):
        # Add a column of ones to X for the intercept
        ones = np.ones((X.shape[0], 1))
        X_b = np.hstack((ones, X))

        # Predict: y = X_b * coefficients
        return X_b.dot(self.coefficients)
```

```
    # Get coefficients (slopes and intercept)
    def get_coefficients(self):
        return self.coefficients

# Load data from CSV using pandas
data = pd.read_csv('multiple.csv')  # Replace with your actual dataset path

# Assuming your CSV has columns 'Area', 'Bedrooms', 'Price'
X = data[['Area', 'Bedrooms']].values  # Independent variables (features)
y = data['Price'].values  # Dependent variable (target)

# Create MultipleLinearRegression object and fit the data
model = MultipleLinearRegression()
model.fit(X, y)

# Get predictions
predictions = model.predict(X)

# Print the coefficients and predictions
print("Coefficients (intercept and slopes):", model.get_coefficients())
print("Predictions:", predictions)

# Compute evaluation metrics
mse = mean_squared_error(y, predictions)
r2 = r2_score(y, predictions)

print(f"Mean Squared Error (MSE): {mse:.3f}")
print(f"R-squared (R2 Score): {r2:.3f}")
```

```
# Display accuracy in percentage
accuracy_percentage = r2 * 100
print(f"Accuracy (R2 Score as percentage): {accuracy_percentage:.2f}%")

# Plotting the data and regression predictions using matplotlib and seaborn
plt.figure(figsize=(8, 6))
sns.set(style="darkgrid")

# Scatter plot for actual data
plt.scatter(y, predictions, color='blue', label='Predicted vs Actual')

# Line representing perfect prediction
plt.plot([min(y), max(y)], [min(y), max(y)], color='red', label='Perfect fit')

# Adding labels and title
plt.title('Multiple Linear Regression: Actual vs Predicted')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.legend()

# Show the plot
plt.show()
```

Output :-

```
Coefficients (intercept and slopes): [ 4438656.79238709    10148.76645301 -2353426.79914625]
Predictions: [28921829.54885615 10337264.13749025 11697198.84219361 ...
  9189366.51034798  9829825.81483974 10083544.97616499]
Mean Squared Error (MSE): 526976266021907.125
R-squared (R2 Score): 0.076
Accuracy (R2 Score as percentage): 7.56%
```