House Price Prediction Mini Project

Aim

To predict house prices based on various features using a linear regression model.

Algorithm

Linear Regression

Description of Workflow

1. **Data Collection**
   Load the dataset containing features such as area, number of bedrooms, and other relevant attributes.

2. **Data Preprocessing**
   Clean the data by handling missing values and encoding categorical variables if necessary.
   Split the dataset into training and testing sets.

3. **Model Training**
   Implement the linear regression algorithm using the training data.
   Use gradient descent to optimize the model parameters (weights and bias).

4. **Model Evaluation**
   Evaluate the model's performance using metrics such as Mean Squared Error (MSE) or R-squared on the testing set.

5. **Prediction**
   Use the trained model to make predictions on new data.

Source Code (with Simulated Data and Fixes)

```
import numpy as np
import matplotlib.pyplot as plt

# Simulate a dataset
def load_data():
    np.random.seed(0)  # For reproducibility
    # Simulating 100 data points
    x_train = np.random.rand(100, 1) * 1000  # Feature:
Area in square feet
    y_train = 150 + 0.3 * x_train + (np.random.randn(100,
1) * 50)  # Price with some noise
    return x_train, y_train

# Compute cost function
def compute_cost(x, y, w, b):
    m = x.shape[0]
    total_cost = np.sum((x.dot(w) + b - y) ** 2) / (2 * m)
    return total_cost

# Compute gradient
def compute_gradient(x, y, w, b):
    m = x.shape[0]
    dj_dw = (1/m) * x.T.dot(x.dot(w) + b - y)  # Transpose
for correct shape
    dj_db = (1/m) * np.sum(x.dot(w) + b - y)
    return dj_dw, dj_db

# Gradient descent
def gradient_descent(x, y, w_in, b_in, alpha, num_iters):
    w = w_in
    b = b_in
    for i in range(num_iters):
        dj_dw, dj_db = compute_gradient(x, y, w, b)
        w -= alpha * dj_dw
        b -= alpha * dj_db
    return w, b

# Main execution
x_train, y_train = load_data()
initial_w = np.zeros((x_train.shape[1],1))  # Column vector
```

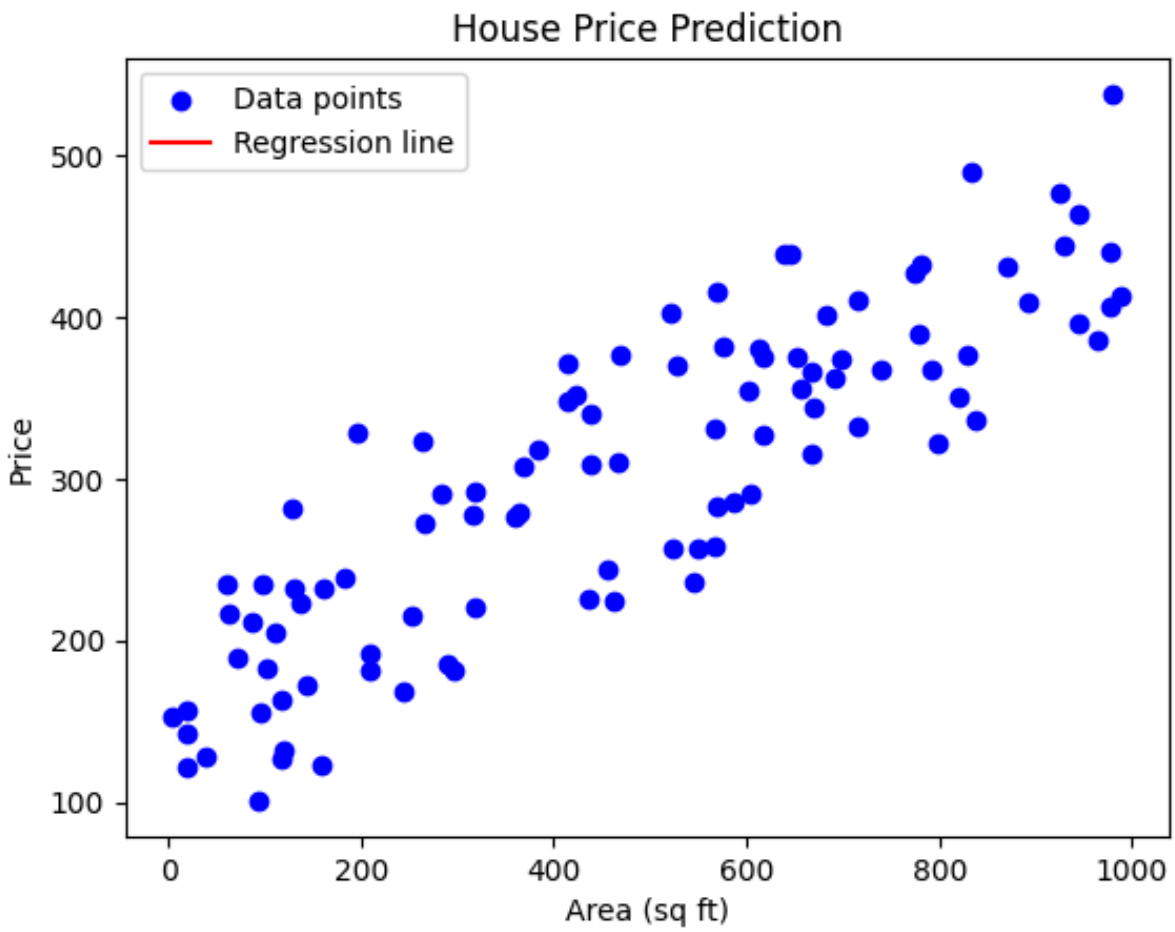
```
initial_b = 0
alpha = 0.01
iterations = 1500

w, b = gradient_descent(x_train, y_train, initial_w,
initial_b, alpha, iterations)
print("w, b found by gradient descent:", w.flatten(), b)

# Optional: Plotting the results
plt.scatter(x_train, y_train, color='blue', label='Data
points')
plt.plot(x_train, x_train.dot(w) + b, color='red',
label='Regression line')
plt.xlabel('Area (sq ft)')
plt.ylabel('Price')
plt.title('House Price Prediction')
plt.legend()
plt.show()
```

Output for the Code

w, b found by gradient descent: [1.16636235] -3.63029143940436



This output indicates the slope and intercept of the linear regression model, which can then be used to make predictions on house prices based on the input features.