linReg

September 16, 2024

Daniel Jolin

Student ID: 801282735

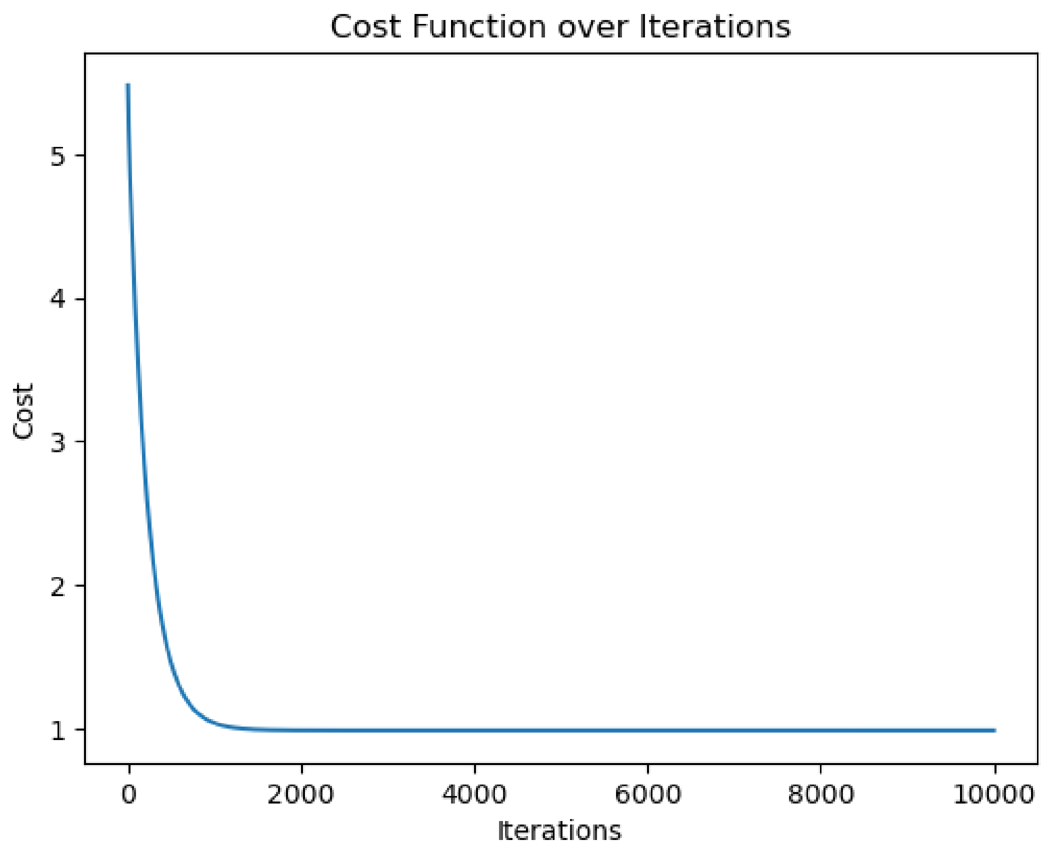
Homework 1

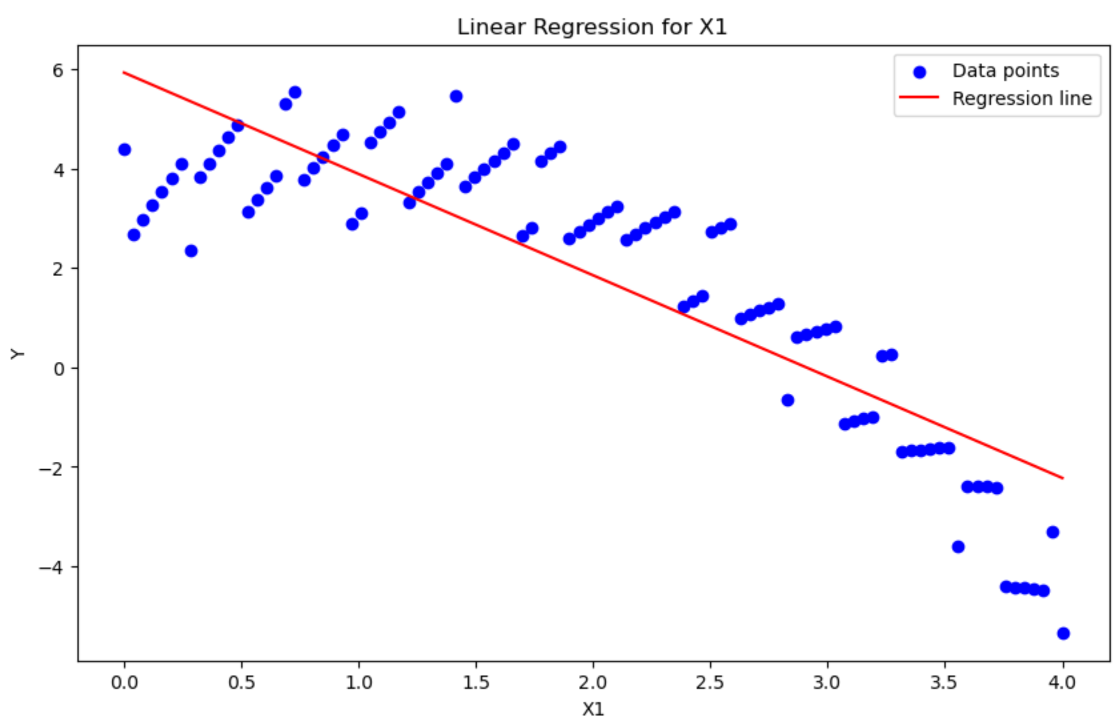
[19]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


def predict(X, theta):
    return np.dot(X, theta)

def compute_cost(X, y, theta):
    m = len(y)
    predictions = predict(X, theta)
    cost = (1 / (2 * m)) * np.sum((predictions - y) ** 2)
    return cost

def gradient_descent(X, y, theta, learning_rate, iterations):
    m = len(y)
    cost_history = []

    for _ in range(iterations):
        predictions = predict(X, theta)
         # Compute gradients
        gradients = (1/m) * np.dot(X.T, (predictions - y))
        # Update parameters
        theta -= learning_rate * gradients
        cost = compute_cost(X, y, theta)
        cost_history.append(cost)

    return theta, cost_history

# Load the data
data = pd.read_csv("assets/D3.csv")
X = data[['X1', 'X2', 'X3']].values
y = data['Y'].values
```

[20]:
```
# Initialize parameters
learning_rate = 0.01
iterations = 10000
```

[21]:
```
# Function to plot single feature regression
def plot_single_feature(X, y, theta, feature_name):
    plt.figure(figsize=(10, 6))
    plt.scatter(X, y, color='blue', label='Data points')
    plt.plot(X, np.dot(np.c_[np.ones((len(X), 1)), X], theta), color='red',␣
 ↪label='Regression line')
    plt.xlabel(feature_name)
    plt.ylabel('Y')
    plt.legend()
    plt.title(f'Linear Regression for {feature_name}')
    plt.show()
```

[22]:
```
# Train on X1
X1 = X[:, 0].reshape(-1, 1)
X1_b = np.c_[np.ones((len(X1), 1)), X1]
theta1 = np.zeros(2)
theta1, cost_history = gradient_descent(X1_b, y, theta1, learning_rate,␣
 ↪iterations)

plt.plot(range(iterations), cost_history)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Cost Function over Iterations')
plt.show()


# Plot for X1
plot_single_feature(X1, y, theta1, 'X1')
print(f'theta is:{theta1}')
print(f'y = {theta1[1]}*X1 + {theta1[0]}')
print(f'final cost is: {cost_history[iterations -1]}')
```

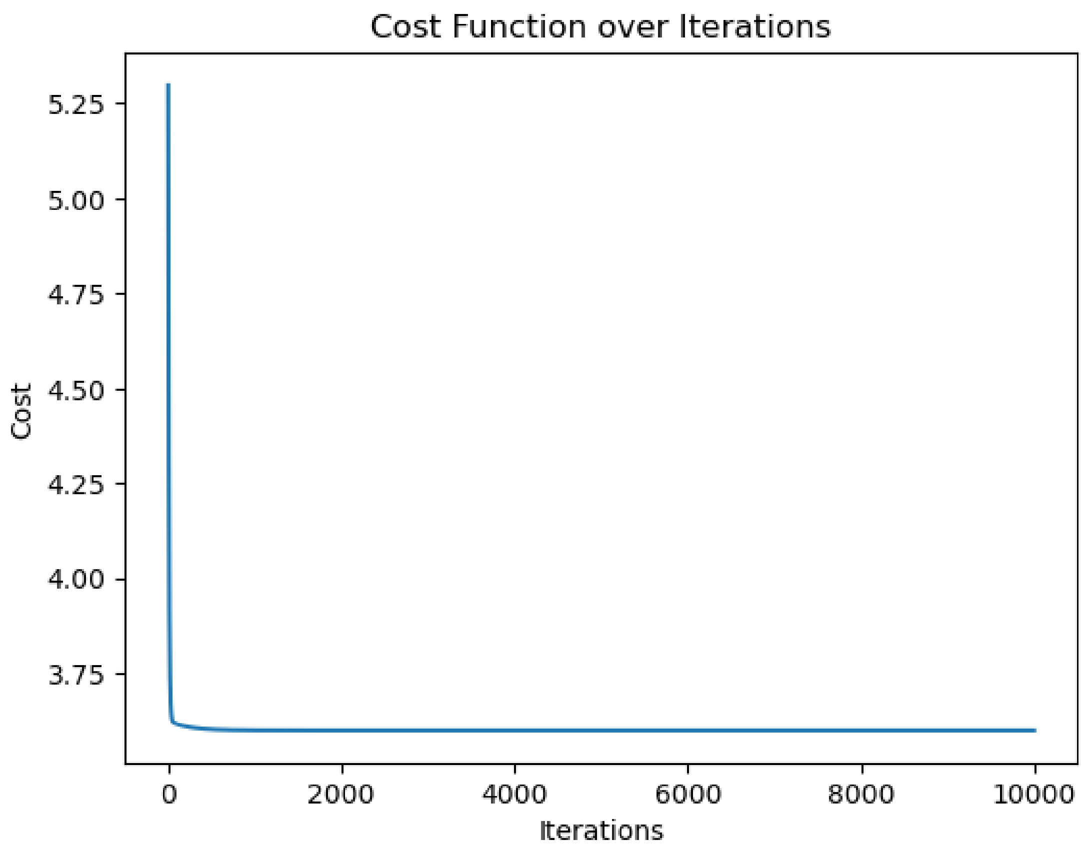

Cost Function over Iterations
Cost
Iterations
5
4
3
2
1
0
2000
4000
6000
8000
10000

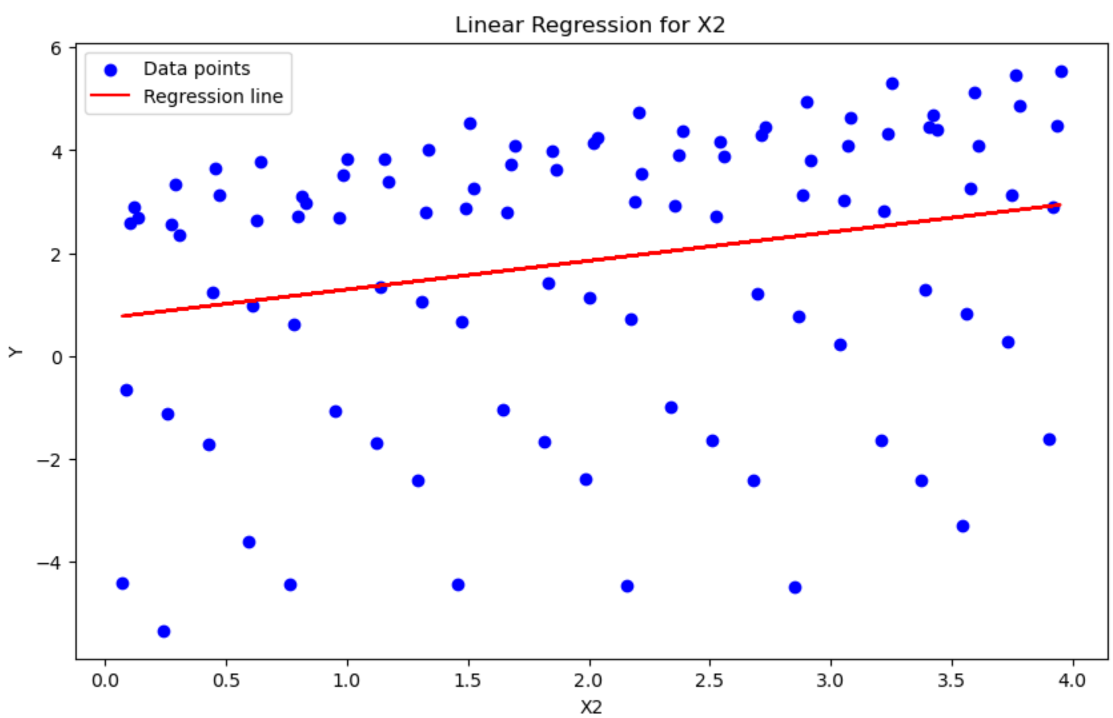

Linear Regression for X1
Data points
Regression line
Y
X1
6
4
2
0
−2
−4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

```
theta is:[ 5.92794892 -2.03833663]
y = -2.0383366331233708*X1 + 5.927948916706452
final cost is: 0.9849930825405946
```

[23]:

```
# Train on X2
X2 = X[:, 1].reshape(-1, 1)
X2_b = np.c_[np.ones((len(X2), 1)), X2]
theta2 = np.zeros(2)
theta2, cost_history = gradient_descent(X2_b, y, theta2, learning_rate,␣
 ↪iterations)

plt.plot(range(iterations), cost_history)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Cost Function over Iterations')
plt.show()

# Plot for X2
plot_single_feature(X2, y, theta2, 'X2')
print(f'theta is:{theta2}')
print(f'y = {theta2[1]}*X2 + {theta2[0]}')
print(f'final cost is: {cost_history[iterations -1]}')
```

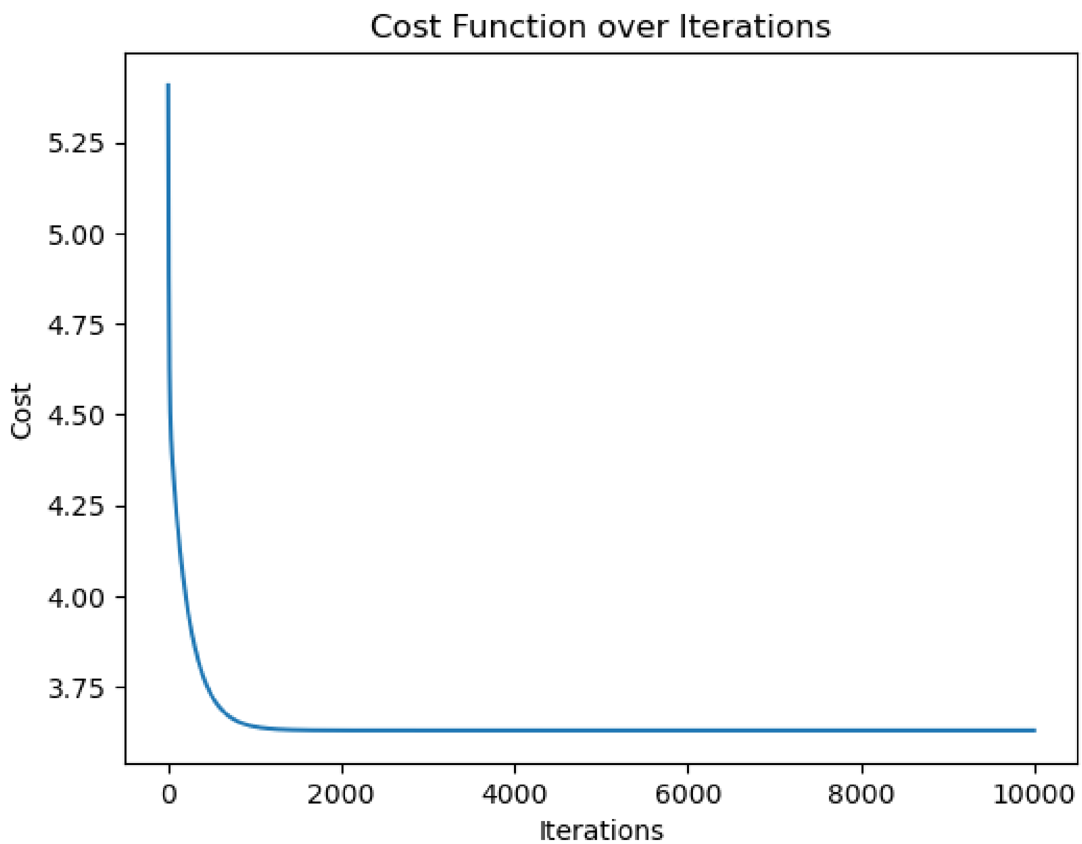

Cost Function over Iterations
Cost
Iterations
5.25
5.00
4.75
4.50
4.25
4.00
3.75
0
2000
4000
6000
8000
10000

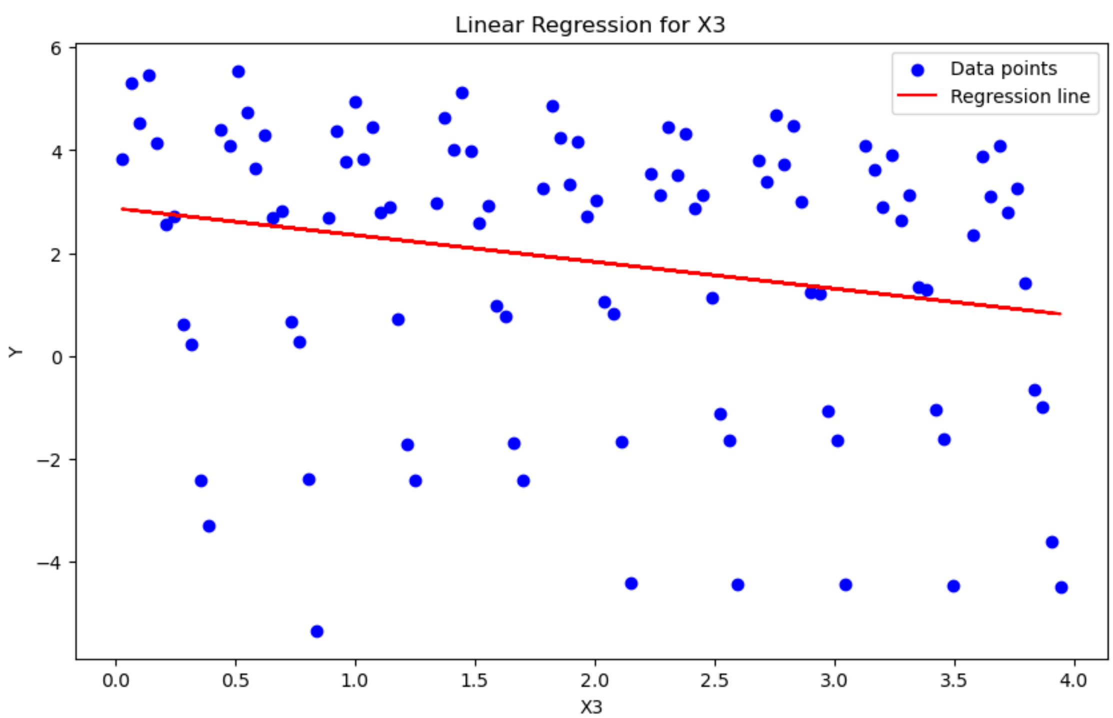

Linear Regression for X2
Data points
Regression line
Y
X2
6
4
2
0
−2
−4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

```
theta is:[0.73606043 0.55760761]
y = 0.557607610373368*X2 + 0.736060429990056
final cost is: 3.5993660181680416
```

[24]:

```python
# Train on X3
X3 = X[:, 2].reshape(-1, 1)
X3_b = np.c_[np.ones((len(X3), 1)), X3]
theta3 = np.zeros(2)
theta3, cost_history = gradient_descent(X3_b, y, theta3, learning_rate,
 ↪iterations)

plt.plot(range(iterations), cost_history)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Cost Function over Iterations')
plt.show()

# Plot for X3
plot_single_feature(X3, y, theta3, 'X3')
print(f'theta is:{theta3}')
print(f'y = {theta3[1]}*X3 + {theta3[0]}')
print(f'final cost is: {cost_history[iterations -1]}')
```

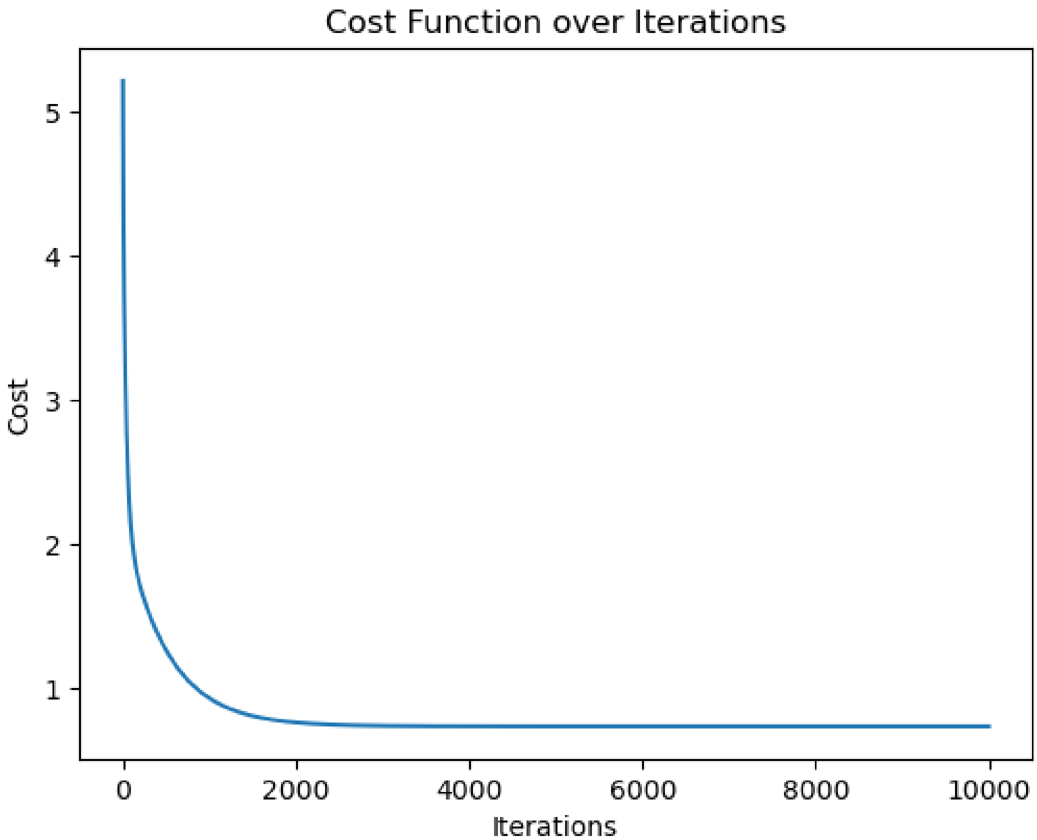

Cost Function over Iterations
Cost
5.25
5.00
4.75
4.50
4.25
4.00
3.75
0
2000
4000
6000
8000
10000
Iterations


Linear Regression for X3
Data points
Regression line
Y
6
4
2
0
−2
−4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
X3

```
theta is:[ 2.8714221  -0.52048288]
y = -0.520482884123542*X3 + 2.871422103541768
final cost is: 3.6294511246079164
```

[25]:
```python
# Add a column of ones to X for the bias term
X_b = np.c_[np.ones((X.shape[0], 1)), X]

theta = np.zeros(X_b.shape[1])  # Initialize theta with zeros (4 parameters:
 ↪bias + 3 features)

# Run gradient descent
theta, cost_history = gradient_descent(X_b, y, theta, learning_rate, iterations)

# Print the optimized parameters
print("theta:", theta)



# predictions
new_X = np.array([[1, 1, 1], [2, 0, 4], [3, 2, 1]])
new_X_b = np.c_[np.ones((new_X.shape[0], 1)), new_X]  # Add bias terme
predictions = predict(new_X_b, theta)
print("Predictions:", predictions)

# Plot the cost function
import matplotlib.pyplot as plt

plt.plot(range(iterations), cost_history)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Cost Function over Iterations')
plt.show()
```

```
theta: [ 5.31392511 -2.00368507  0.53260334 -0.26556638]
Predictions: [3.57727699 0.24428943 0.10251018]
```


Cost Function over Iterations
Cost
5
4
3
2
1
0
2000
4000
6000
8000
10000
Iterations