Linear Regression Optimization

Overview:

This report details the implementation of a linear regression optimization algorithm using gradient descent. The primary objective is to minimize the Mean Squared Error (MSE) cost function by updating the model parameters iteratively.

Code Implementation:

Importing Libraries:

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

We use NumPy for numerical operations and Matplotlib for data visualization.

Given Data:

```
x = np.array([5, 10, 20, 30])
y = np.array([10, 15, 25, 35])
theta = np.array([1, 1])  # Initial parameters
alpha = 0.002  # Learning rate
```

The dataset consists of input (`x`) and target output (`y`) arrays, and initial parameters (`theta`). The learning rate (`alpha`) determines the step size during optimization.

Gradient Descent Loop:

```
num_iterations = 100
m = len(y)
cost_values = []
iterations_list = []
```

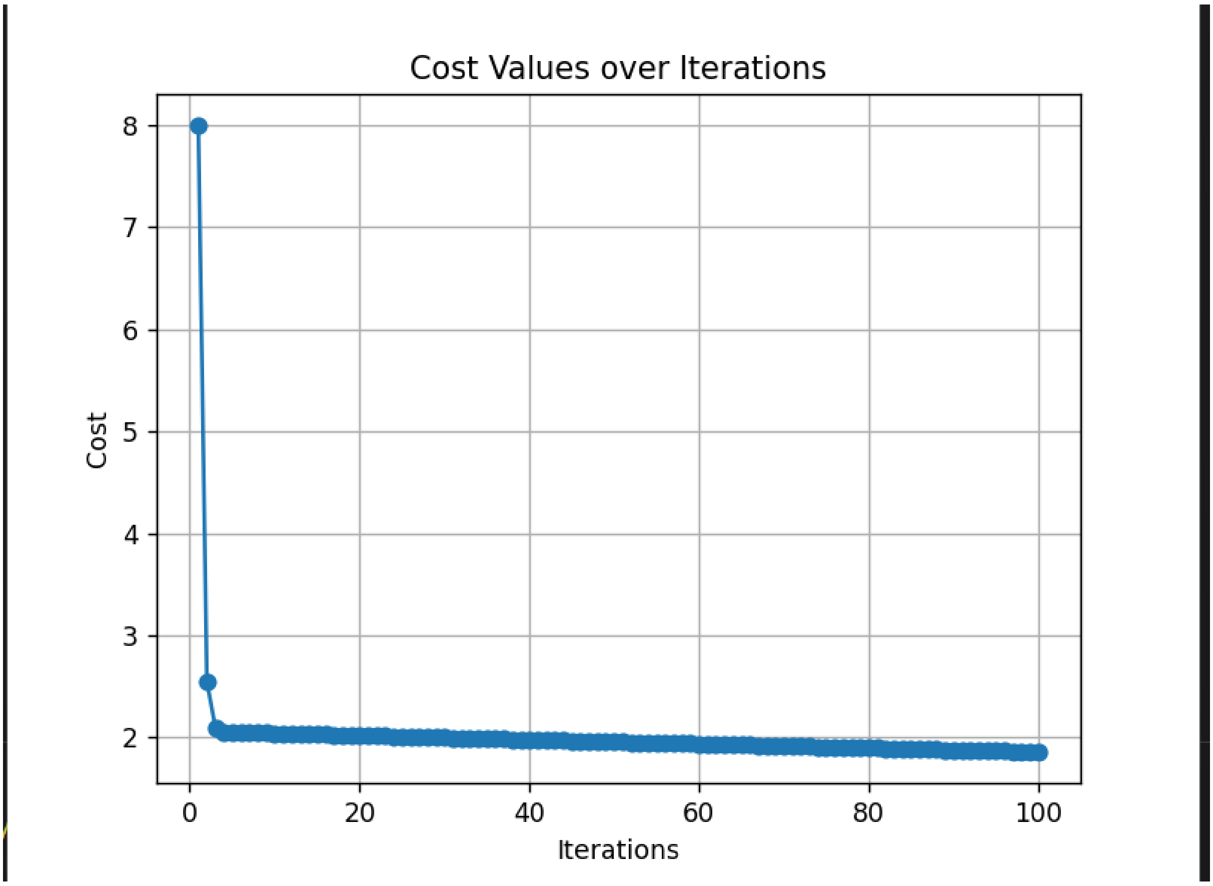
```
for iteration in range(num_iterations):
    # ... [Gradient descent calculations]
    cost_values.append(cost_sum / (2 * m))
    iterations_list.append(iteration + 1)
```

The gradient descent loop iterates through 100 cycles, updating parameters to minimize the cost function. Cost values and iteration numbers are stored for later visualization.

Plotting Results:

```
plt.plot(iterations_list, cost_values, marker='o')
plt.title('Cost Values over Iterations')
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.grid(True)
plt.show()
```

The resulting cost values over iterations are plotted for analysis.



Interpretation:

The plotted graph visually represents the convergence of the optimization algorithm. The x-axis denotes iterations, while the y-axis shows the cost function's values. A decreasing trend indicates successful optimization, moving towards the minimum cost and improving the accuracy of the linear regression model.