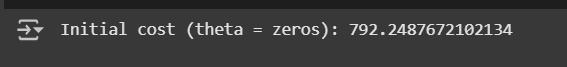
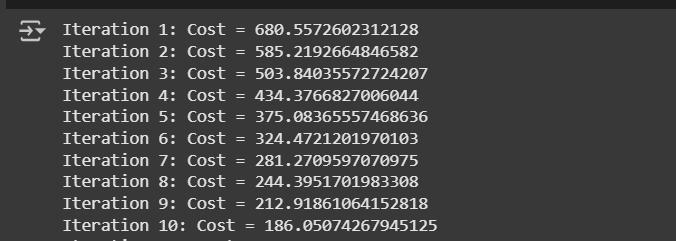


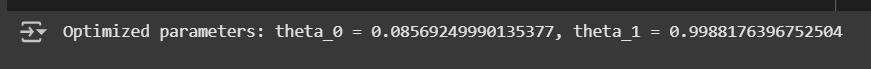
**FACULTY OF ENGINEERING AND**  **TECHNOLOGY** Department of Computer Engineering 01CE0524 – Machine Learning – Lab

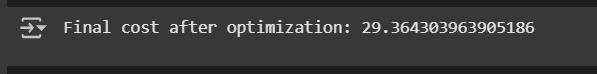
**Practical-6: Develop a Gradient descent of linear regression using sample dataset.**

**Output:**





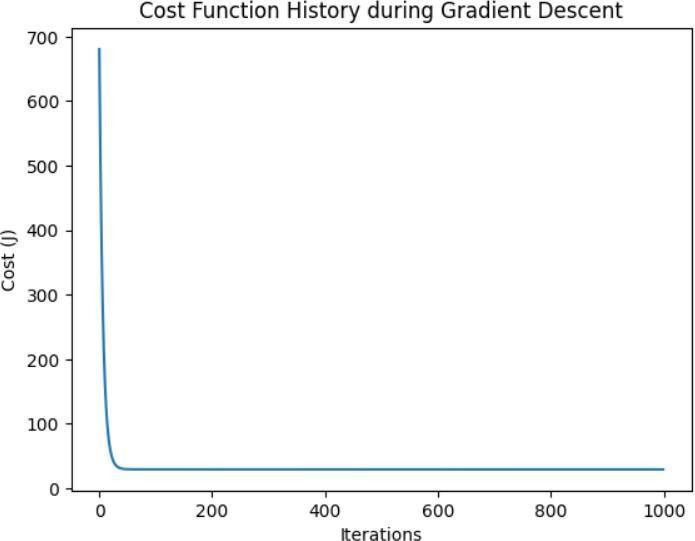


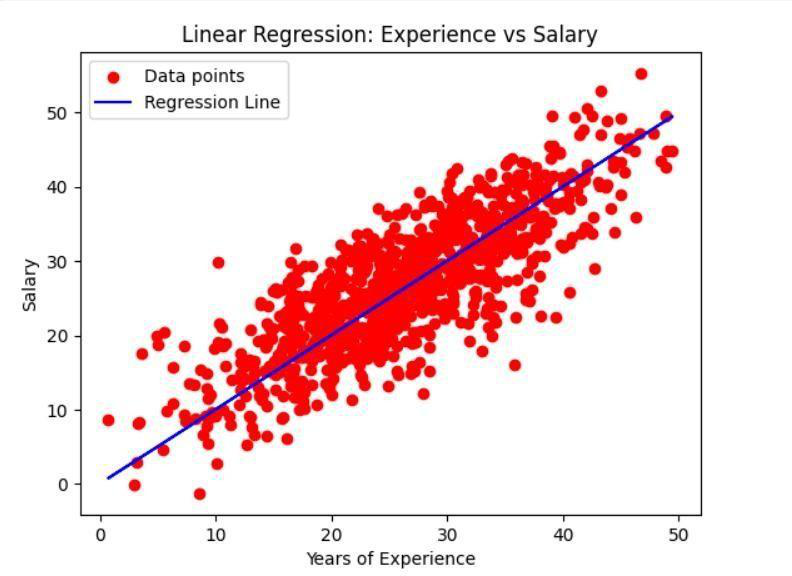


|  |  |
| --- | --- |
| Piyush Solanki (92301703038) (5EC4) | 21 |



**FACULTY OF ENGINEERING AND**  **TECHNOLOGY** Department of Computer Engineering 01CE0524 – Machine Learning – Lab





|  |  |
| --- | --- |
| Piyush Solanki (92301703038) (5EC4) | 22 |



**FACULTY OF ENGINEERING AND**  **TECHNOLOGY** Department of Computer Engineering 01CE0524 – Machine Learning – Lab

**Code:**   
import pandas as pd   
import numpy as np   
import matplotlib.pyplot as plt   
**# 1. Load dataset**   
df= pd.read\_csv("Experience-Salary.csv")   
**# Step 2: Extract input (x) and output (y)**   
**data** x = data['exp(in months)'].values   
y = data['salary(in thousands)'].values   
m = len(x) **# Number of data points**   
**# Step 3: Prepare the data (add column of ones for intercept**   
**term)**   
X = np.c\_[np.ones(m), x] **# Shape: (m, 2)**   
**# Step 4: Initialize parameters (theta\_0 and**   
**theta\_1)** theta = np.zeros(2) **# [theta\_0, theta\_1]**   
**# Step 5: Learning rate and number of iterations for gradient descent** learning\_rate = 0.0001   
iterations = 1000   
**# Step 6: Define the cost function (Mean Squared Error)**   
def cost\_function(X, y, theta):   
 predictions = X.dot(theta)   
 cost = (1 / m) \* np.sum((predictions - y) \*\* 2)   
 return cost   
**# Step 7: Gradient descent function to optimize theta**

|  |  |
| --- | --- |
| Piyush Solanki (92301703038) (5EC4) | 23 |



**FACULTY OF ENGINEERING AND**  **TECHNOLOGY** Department of Computer Engineering 01CE0524 – Machine Learning – Lab

def gradient\_descent(X, y, theta, learning\_rate, iterations): cost\_history = []   
for i in range(iterations):   
 predictions = X.dot(theta)   
 errors = predictions - y   
 gradient = (1 / m) \* X.T.dot(errors)   
 theta -= learning\_rate \* gradient   
 current\_cost = cost\_function(X, y, theta)   
 cost\_history.append(current\_cost)   
 **# Optional: print cost at each iteration**   
 print(f"Iteration {i+1}: Cost = {current\_cost}")   
 return theta, cost\_history   
**# Step 8: Print initial cost before training**   
initial\_cost = cost\_function(X, y, theta)   
print(f"Initial cost (theta = zeros): {initial\_cost}")

**# Step 9: Run gradient descent**   
theta\_optimized, cost\_history = gradient\_descent(X, y, theta, learning\_rate, iterations)   
**# Step 10: Output optimized parameters**   
print(f"Optimized parameters: theta\_0 = {theta\_optimized[0]}, theta\_1 = {theta\_optimized[1]}") **# Step 11: Print final cost**   
final\_cost = cost\_function(X, y, theta\_optimized)   
print(f"Final cost after optimization: {final\_cost}")

|  |  |
| --- | --- |
| Piyush Solanki (92301703038) (5EC4) | 24 |



**FACULTY OF ENGINEERING AND**  **TECHNOLOGY** Department of Computer Engineering 01CE0524 – Machine Learning – Lab

**# Step 12: Plot cost function history**   
plt.plot(range(iterations), cost\_history)   
plt.xlabel('Iterations')   
plt.ylabel('Cost (J)')   
plt.title('Cost Function History during Gradient Descent') plt.show() **# Step 13: Plot data points and regression line**   
plt.scatter(x, y, color='red', label='Data points')   
plt.plot(x, X.dot(theta\_optimized), label='Regression Line', color='blue') plt.xlabel('Years of Experience')   
plt.ylabel('Salary')   
plt.title('Linear Regression: Experience vs Salary')   
plt.legend()   
plt.show()

|  |  |
| --- | --- |
| Piyush Solanki (92301703038) (5EC4) | 25 |