TAS Rename notebook)42112

```
import pandas as pd
import random
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import numpy as np


seed_value = 210042112
random.seed(seed_value)

names = [f"STD_{i}" for i in range(1, 1001)]
cgpa = [round(random.uniform(2.4, 4.0), 2) for _ in range(1000)]
num_projects = [random.randint(0, 20) for _ in range(1000)]
num_internships = [random.randint(0, 20) for _ in range(1000)]
salary = [round(30000 + (cgpa[i] - 2.4) * 25000 / 1.6 + num_projects[i] * 2000 +
                num_internships[i] * 1000 +random.uniform(-5000, 5000), 2)for i in range(1000)]


data = {
    "Name": names,
    "CGPA": cgpa,
    "Number of Projects": num_projects,
    "Number of Internship": num_internships,
    "Salary (In BDT)": salary,
}
df = pd.DataFrame(data)

file_path = 'graduates_data.csv'
df.to_csv(file_path, index=False)

print(f"Dataset saved to {file_path}")
```

```
Dataset saved to graduates_data.csv
```

```
print(df.head())
```

```
     Name  CGPA  Number of Projects  Number of Internship  Salary (In BDT)
0  STD_1  3.99                   6                    13         84500.23
1  STD_2  2.68                   0                     7         38717.90
2  STD_3  3.58                   4                     0         53594.48
3  STD_4  3.51                  13                     1         79220.61
4  STD_5  2.73                   5                    19         65741.64
```

```
X = df[["CGPA", "Number of Projects", "Number of Internship"]].values
y = df["Salary (In BDT)"].values

X = np.hstack((np.ones((X.shape[0], 1)), X))


theta = np.zeros(X.shape[1])

def h(x, theta):
    return np.dot(x, theta)


def cost_function(x, y, theta):
    m = len(y)
    predictions = h(x, theta)
    cost = (1 / (2 * m)) * np.sum((predictions - y) ** 2)
    return cost


def gradient_descent(x, y, theta, learning_rate=0.1, num_epochs=10):
    m = len(y)
    cost_history = []

    for epoch in range(num_epochs):
        predictions = h(x, theta)
        gradient = (1 / m) * np.dot(x.T, (predictions - y))
        theta = theta - learning_rate * gradient
```

```
            cost = cost_function(x, y, theta)
            cost_history.append(cost)
            # Rename notebook {epoch+1}/{num_epochs}, Cost: {cost:.2f}, Theta: {theta}')

        return theta, cost_history


final_theta, cost_history = gradient_descent(X, y, theta, learning_rate=0.001, num_epochs=100)


test_input = np.array([1, 3.92, 6, 1])
predicted_salary = h(test_input, final_theta)

print(f"Predicted Salary (In BDT): {predicted_salary:.2f}")
```

```
Predicted Salary (In BDT): 35042.70
```