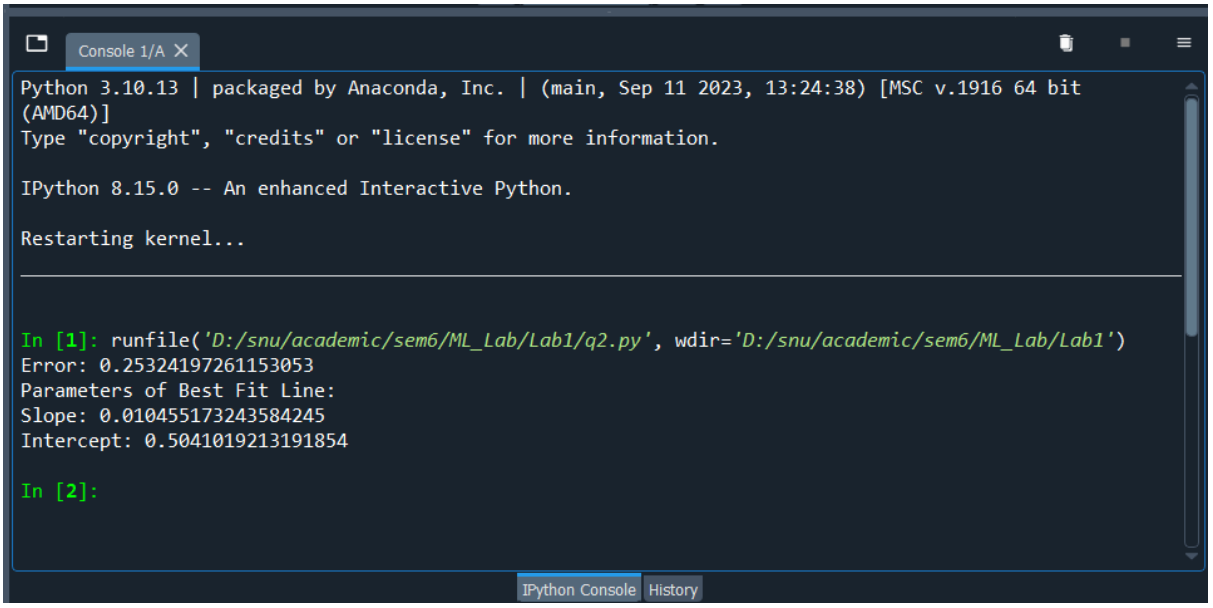
Ex 1	Linear Regression
27.12.2023	

Q1 : Create a random 2-D numpy array with 1500 values. Simulate different lines of fit using 1000 values from the array and find the errors for each of these lines. Find the line with the least error among these lines and store it as the line of best fit. Using this line of best fit, predict the target variable for the other 500 values.

CODE :

```
import numpy as np
from sklearn.linear_model import LinearRegression

np.random.seed(42)
random_array = np.random.rand(1500, 2)

x_values = random_array[:, 0]
y_values = random_array[:, 1]

def calculate_error(y_true, y_pred):
    return np.mean(np.abs(y_true - y_pred))


best_fit_line = None
min_error = float('inf')


for i in range(1000):

    ind = np.random.choice(1000, 900, replace=False)

    x_train = x_values[ind]
```

```
    y_train = y_values[ind]

    x_train = x_train.reshape(-1, 1)

    model = LinearRegression()
    model.fit(x_train, y_train)


    y_pred = model.predict(x_train.reshape(-1, 1))

    error = calculate_error(y_train, y_pred)


    if error < min_error:
        min_error = error
        best_fit_line = model

x_test = x_values[1000:]
y_test = y_values[1000:]
x_test = x_test.reshape(-1, 1)
predicted_values = best_fit_line.predict(x_test)


print("Error:", calculate_error(y_test, predicted_values))
print("Parameters of Best Fit Line:")
print("Slope:", best_fit_line.coef_[0])
print("Intercept:", best_fit_line.intercept_)
```



Q2 : Use the data1.csv to build a simple linear regression from scratch without using sklearn libraries and print the RMSE and mean absolute error values. Use both the equations available in the slides (in theory page) to build the model and compare the intercept and coefficient values.

CODE :

```
import csv
import math
import numpy as np
data = []
with open('data1.csv', 'r') as file:
    csv_reader = csv.reader(file)
    next(csv_reader)
    for row in csv_reader:
        data.append([float(row[0]), float(row[1])])


def calculate_coefficients_partial_derivative(data):
    x = [row[0] for row in data]
    y = [row[1] for row in data]
```

```
    x_mean = sum(x) / len(x)
    y_mean = sum(y) / len(y)


    numerator = sum((x[i] * y[i] - y_mean * x[i]) for i in
range(len(data)))
    denominator = sum((x[i] ** 2 - x_mean * x[i]) for i in
range(len(data)))


    b1 = numerator / denominator
    b0 = y_mean - b1 * x_mean
    return b0, b1


def calculate_metrics(data, b0, b1):
    mse = 0
    mae = 0
    n = len(data)


    for i in range(n):
        x, y = data[i]
        y_pred = b0 + b1 * x
        mse += (y - y_pred) ** 2
        mae += abs(y - y_pred)


    mse /= n
    rmse = math.sqrt(mse)
    mae /= n
    return rmse, mae


def calculate_coefficients_correlation(data):
    x = [row[0] for row in data]
```

```
    y = [row[1] for row in data]
    x_mean, y_mean = np.mean(x), np.mean(y)

    numerator = sum((x[i] - x_mean) * (y[i] - y_mean) for i in
range(len(data)))
    denominator = math.sqrt(sum((x[i] - x_mean)**2 for i in
range(len(data))) * sum((y[i] - y_mean)**2 for i in
range(len(data))))

    r = numerator / denominator if denominator != 0 else 0

    Sx = np.std(x)
    Sy = np.std(y)
    b1 = r * Sy / Sx
    b0 = y_mean - b1 * x_mean
    return b0, b1


intercept_partial, coefficient_partial =
calculate_coefficients_partial_derivative(data)


intercept_correlation, coefficient_correlation =
calculate_coefficients_correlation(data)


rmse_partial, mae_partial = calculate_metrics(data,
intercept_partial, coefficient_partial)


rmse_correlation, mae_correlation = calculate_metrics(data,
intercept_correlation, coefficient_correlation)


print("Using Partial Derivative Equations:")
print(f"Intercept (b0): {intercept_partial}")
```

```python
print(f"Coefficient (b1): {coefficient_partial}")
print(f"Root Mean Squared Error (RMSE): {rmse_partial}")
print(f"Mean Absolute Error (MAE): {mae_partial}")
print("\nUsing Correlation Coefficient and Standard Deviation:")
print(f"Intercept (b0): {intercept_correlation}")
print(f"Coefficient (b1): {coefficient_correlation}")
print(f"Root Mean Squared Error (RMSE): {rmse_correlation}")
print(f"Mean Absolute Error (MAE): {mae_correlation}")
```