ASSIGNMENT 1 : LINEAR REGRESSION

Link for dataset:
Name:_____
Read the whole assignment before starting, Part 2 has the code template for doing Part 1.

Part 1: The Assignment Tasks

1. **Define Loss Functions:** Create two Python functions that calculate the total error for a given slope (m) and intercept (c).
2. **Optimization:** Use scipy.optimize.minimize to find the best-fit parameters for Datasets A and B (using MSE).
3. **The Outlier Challenge:** For **Dataset C**, find two sets of parameters:
    - One that minimizes the **MSE**.
    - One that minimizes the **MAE**.
4. **Visualization:** Plot Dataset C with both the MSE line and the MAE line. Highlight the outliers in a different color.
5. **Statistical Analysis:** For both lines in Dataset C, calculate:
    - The number of points **above** vs **below** the line.
    - The **Mean Distance** of points on either side.

Part 3: Implementation Template

Required Tools

- **Library:** from scipy.optimize import minimize
- **Core Function:** minimize(fun, x0, args=(x, y))
    - fun: The loss function you want to minimize.
    - x0: Your initial guess for [m, c], e.g., [0, 0].
    - args: The actual data (x, y) passed to your loss function
    - This function changes the values of x0 such that it minimizes the function fun for the given args.

Code Skeleton

Python

```
#import matplotlib.pyplot numpy, pandas and the above library
# --- STEP 1: DEFINE LOSS FUNCTIONS —
# save the given csv file as df
x=df['x'].values
data_c=df['y'].values
def mse_loss(params, x, y):
    m, c = params
```

```
    y_hat = m * x + c
    # Calculate Mean Squared Error
    # YOUR CODE HERE: return np.mean(...)

def mae_loss(params, x, y):
    m, c = params
    y_hat = m * x + c
    # Calculate Mean Absolute Error
    # YOUR CODE HERE: return np.mean(np.abs(...))

# --- STEP 2: OPTIMIZATION ---
# Example for Dataset C using MSE
initial_guess = [0, 0]
result_mse = minimize(mse_loss, initial_guess, args=(x, data_c))
m_mse, c_mse = result_mse.x

# YOUR CODE HERE: Repeat for MAE on Dataset C
# result_mae = minimize(...)

# --- STEP 3: RESIDUAL ANALYSIS ---
def analyze_line(x, y, m, c):
    y_pred = m * x + c
    residuals = y - y_pred


    # YOUR CODE HERE:
    # 1. Count points where residuals > 0 (Above) and residuals < 0 (Below)
    # 2. Calculate mean of residuals for Above group
    # 3. Calculate mean of ABSOLUTE residuals for Below group

    print(f"Above: {count_above} (Avg Dist: {dist_above:.2f})")
    print(f"Below: {count_below} (Avg Dist: {dist_below:.2f})"

# CODE FOR PLOTTING

import matplotlib.pyplot as plt
#assuming that your ideal m and c values are stored as m_opt and c_opt
# 1. Generate predictions using the optimized parameters
# We use the original x_data to create the y-values for the line
y_pred = m_opt * x_data + c_opt

# 2. Create the plot
plt.figure(figsize=(10, 6))

```

```
# Plot original data points as a scatter plot
plt.scatter(x_data, y_data, color='blue', alpha=0.5, label='Original Data')

# Plot the regression line
plt.plot(x_data, y_pred, color='red', linewidth=2, label=f'Fit: y = {m_opt:.2f}x + {c_opt:.2f}')

# 3. Add labels and styling
plt.title('Linear Regression using scipy.optimize.minimize', fontsize=14)
plt.xlabel('x', fontsize=12)
plt.ylabel('y', fontsize=12)
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```

Part 4: Discussion Questions for Student

- **Balance:** Which line (MSE or MAE) split the 2,000 points more evenly (1000 above / 1000 below)? Why? Write the count of split of points above and below the line for dataset C.
- **Comment on the sensitivity of the two loss functions in dataset C and write the observations and reasoning for the observations**