



TANGLATEC SCHOOLSTRAAT 8 9883,
BELLEM, BELGIUM +32486257049
info@tanglatec.com www.tanglatec.com
VAT: BE0644.780.972

DETAILED WEEKLY REPORT MACHINE LEARNING AND DATA SCIENCE WEEK 4

Submitted by Nyuysemo Calglain

Under the supervision of
TANGLATECH

23/08/2024

Task

Given the python and c++ code worked on earlier

- 1. Run all the files in your C++ and python code**
- 2. Identify the maximum voltage and the corresponding x-value (direct detection and fitted peaks)**
- 3. plot x versus maximum voltage for all the data files (so, ~1300 peaks versus x values)**

Contents

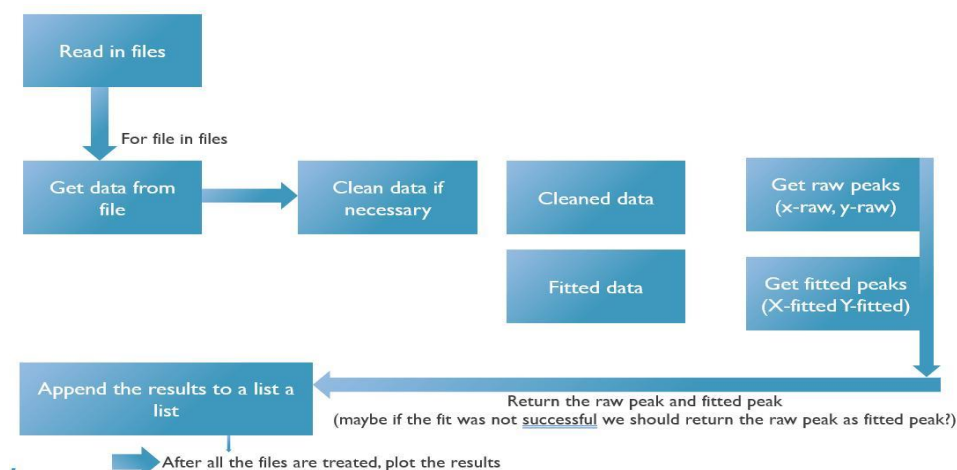
1	Introduction	3
2	Python Code implementation	3
2.1	What was already there.....	3
2.2	Improvements.....	4
2.2.1	Read in files and get data from files	4
2.2.2	Clean and analyze data	5
2.2.3	Append the results to a list	6
2.3	Plotting x versus maximum voltage for all the data files.....	7
2.4	Final results	8
2.5	Problems faced	11
2.6	Solution	11
2.7	Solution	11
3	C++ code implementation.....	12
3.1	Summary of previous Code Functionality	12
3.2	Improvements.....	12
3.2.1	Additions	12
3.3	Issues faced	15
4	General Conclusion	16
5	References	16

1 Introduction

In this report we will discuss how the tasks above were targeted in both c++ and python that is the method, achievements, failures, problems faced and proposed solutions. Screenshots for various aspects or sections of the code will be provided and a guide on how to execute the code. The actual code has been attached to this report and pushed to GitHub at <https://github.com/Carlglain/TanglaTec-internship>.

This report contains two sections, 1 for the python implementation and the other for the c++ implementation.

Generally, for both the python and the c++ implementation of this data analysis problem I tried as much as possible to stick to the algorithm shown on the image below.



2 Python Code implementation

2.1 What was already there

After library importation and with the previous code that was able to import a single dataset in .txt, .csv, .tsv and excel format and process the data. The previous code could also find the peak voltage from the provided data with its corresponding x value as well as the maximum value gotten from the gaussian fitting method and its x location.

2.2 Improvements

2.2.1 Read in files and get data from files

To follow the algorithm depicted by the image above I had to import another library import “os”

Which permits directory and file operations like;

“os.listdir(path)” which lists files and directories in a directory.

I then created a variable called directory_path which holds the path to the folder where our datasets are stored. I also created another variable called index so that when ill be operating a dataset it'll help indicate the position of the particular dataset am working with.

I then created a loop that goes through the directory path and retrieve each dataset at a time and then perform analysis on the dataset. After each iteration a single dataset is passed to the analyze_dataset() function I created and the data it returns is collected and appended to two arrays, one that stores the direct max results and the other stores the gaussian max results. This data passed to the analyze_dataset function comes from taking the file path returned from the os operation and passed to the readData() function from our previous implementation.

This implementation can be seen on the screenshot below.

Screenshot:

```
#declare 2 arrays that will hold the data from the anaylse_dataset function
dir_results = []
gauss_results = []
directory_path = "test_data"
index = 1 #to help identify the index of the file we are currently processing

#loop through the data set folder and perform all the necessary operations(read the data, process the

for file_name in os.listdir(directory_path): #get a single dataset at a time (identified by file_name)
    file_path = os.path.join(directory_path, file_name)
    data = ReadData(file_path)
    if data is not None:
        direct, gauss = analyze_dataset(data,file_name,index) #analyse dataset and return a tuple cont

        #add the results to an array containing the results from other datasets
        dir_results.append(direct)
        gauss_results.append(gauss)
        index +=1
```

2.2.2 Clean and analyze data

To analyze a data a function `analyze_dataset(x,y,z)` was created that takes 3 parameters as input and returns two dictionaries one with the direct analysis results and the other with the gaussian analysis results.

In this function most functions from our previous week implementation like the `DirectMaxSearch()` and the `fit_gaussian()` functions .

Before analyzing the data set to look for the different maxima and their positions data cleaning measures are first taken.

Another improvement to the code was the application of data cleaning measures.

The logic of the cleaning measure I took is that if there are missing values in a dataset the missing value is filled with the previous value. Outliers were also handled and duplicate data files dropped. The implementation is as below.

Screenshot:

```
#function to perform data cleaning
def clean_data(data):
    # Handle missing values
    data.ffill() # Replace missing values with previous value

    # Handle outliers
    Q1 = data['Y'].quantile(0.25)
    Q3 = data['Y'].quantile(0.75)
    IQR = Q3 - Q1
    data = data[~((data['Y'] < (Q1 - 1.5 * IQR)) | (data['Y'] > (Q3 + 1.5 * IQR)))]

    # Handle duplicates
    data.drop_duplicates(inplace=True)

    return data
```

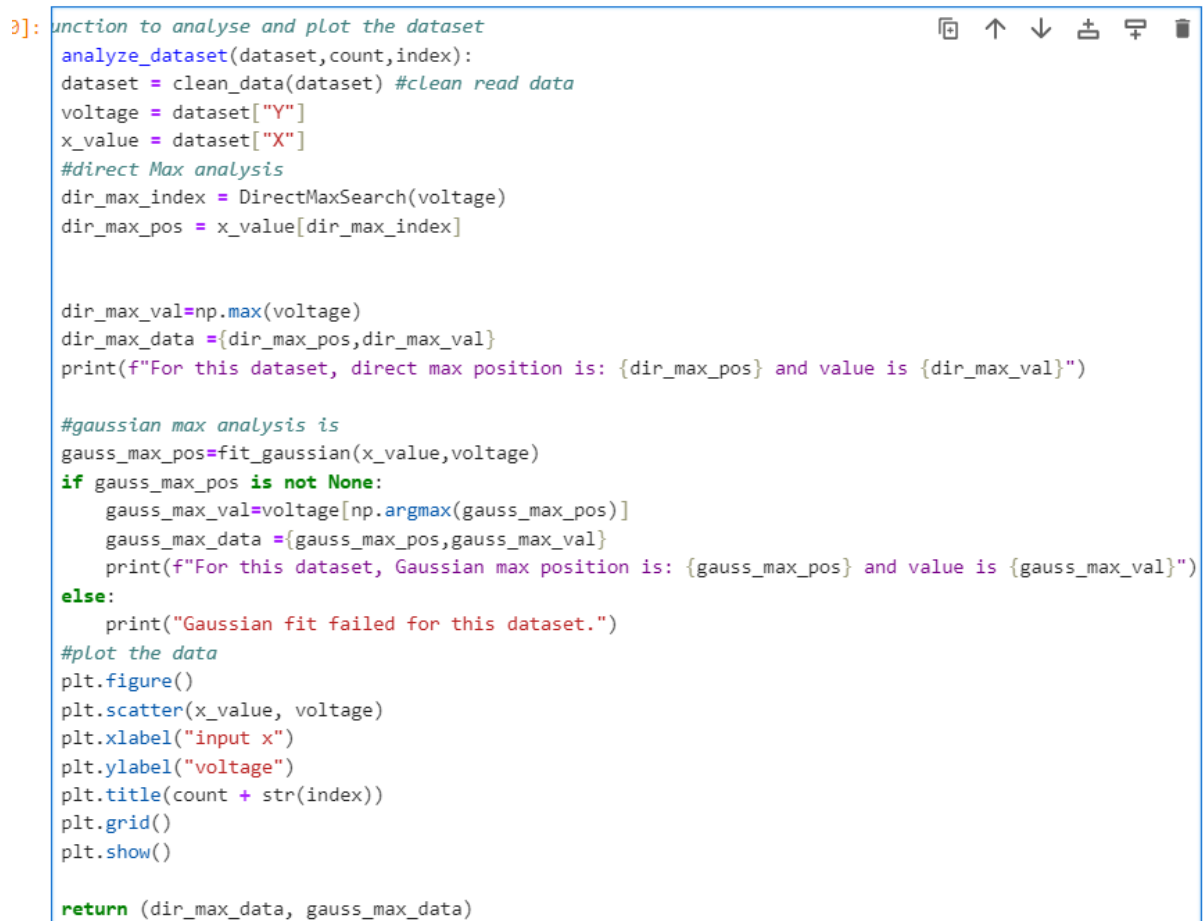
Proceeding with the analysis. After the data has been cleaned, we call our `directMaxSearch()` function and store it's return value in a variable and then store the actual maximum value in another variable then store these two variables in a dictionary which will later be returned.

The `fit_gaussian()` function is also called and the gaussian maximum value and it's position stored in a dictionary.

The two dictionaries are then returned as a tuple from the `analyze_dataset()` function.

The implementation of this function is shown on the screenshot below.

Screenshot:



```
def analyze_dataset(dataset, count, index):
    dataset = clean_data(dataset) #clean read data
    voltage = dataset["Y"]
    x_value = dataset["X"]
    #direct Max analysis
    dir_max_index = DirectMaxSearch(voltage)
    dir_max_pos = x_value[dir_max_index]

    dir_max_val=np.max(voltage)
    dir_max_data = {dir_max_pos,dir_max_val}
    print(f"For this dataset, direct max position is: {dir_max_pos} and value is {dir_max_val}")

    #gaussian max analysis is
    gauss_max_pos=fit_gaussian(x_value,voltage)
    if gauss_max_pos is not None:
        gauss_max_val=voltage[np.argmax(gauss_max_pos)]
        gauss_max_data = {gauss_max_pos,gauss_max_val}
        print(f"For this dataset, Gaussian max position is: {gauss_max_pos} and value is {gauss_max_val}")
    else:
        print("Gaussian fit failed for this dataset.")
    #plot the data
    plt.figure()
    plt.scatter(x_value, voltage)
    plt.xlabel("input x")
    plt.ylabel("voltage")
    plt.title(count + str(index))
    plt.grid()
    plt.show()

    return (dir_max_data, gauss_max_data)
```

2.2.3 Append the results to a list

Calling the `analyze_datasets()` function and it's result appended to separate lists as we discussed earlier, at the end of the execution we export these two list as two separate csv files one for the direct max analysis results and the other for the gaussian analysis results.

As shown by the screenshot below.

Screenshot:

```
[12]: #storing the data gotten from reading the datasets in seperate files
df_gauss=pd.DataFrame(gauss_results)
df_gauss.to_csv("gaussianData.csv", index =False) #store guassian max data in gaussianData.csv file

df_dir=pd.DataFrame(dir_results)
df_dir.to_csv("directData.csv", index =False) #store the direct max data in directData.csv
```

2.3 Plotting x versus maximum voltage for all the data files

Here I created a function `read_and_process_final_datasets(fileName)` that takes a file name probably one of the csv files we exported and plots the data and also displays the best fit line as shown below

Screenshot:

```

#Plotting the final maximum data and their positions
def read_and_process_final_datasets(fileName):
    df = pd.DataFrame(pd.read_csv(fileName, delimiter = ",", "")) #read the data from the file where we store
    df.columns = ["x", "y"]
    volt = df["y"]
    x_val = df["x"]
    #plot the data
    plt.figure()
    plt.scatter(x_val, volt)
    plt.xlabel("max x")
    plt.ylabel("max voltage")
    plt.title("Maximum x_values against Maximum voltage")
    # Set the font size and style
    plt.rcParams.update({'font.size': 12})
    # plt.style.use('seaborn-whitegrid')

    plt.grid()
    plt.show()

    #displaying the best fitline graph
    slope, intercept, r_value, p_value, std_err = stats.linregress(x_val, volt)

    # Create the best fit line function
    def best_fit_line(x):
        return slope * x + intercept
    # Generate x values for the best fit line
    x_fit = np.linspace(min(x_val), max(x_val), 100) # Adjust the range of x_fit as needed
    # Plot the best fit line
    plt.plot(x_fit, best_fit_line(x_fit), color='red')
    plt.title("Scatter Plot of Maximum x-values against Maximum Voltage with Best Fit Line ")
    plt.show()

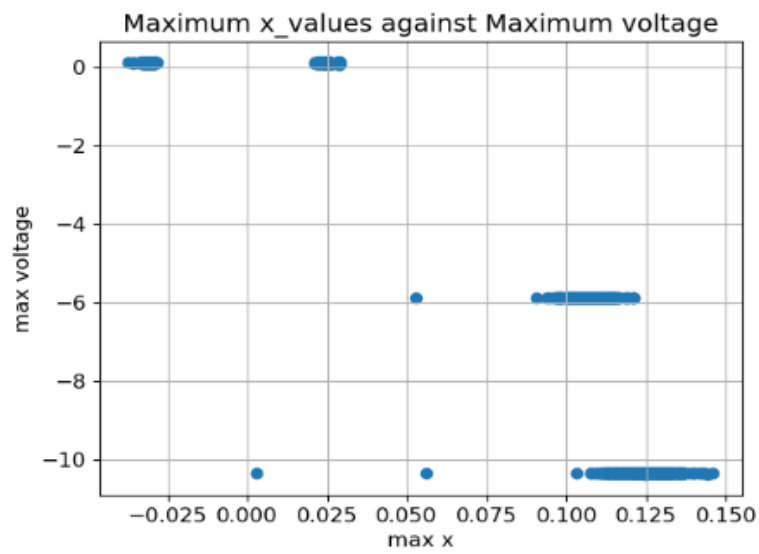
```

2.4 Final results

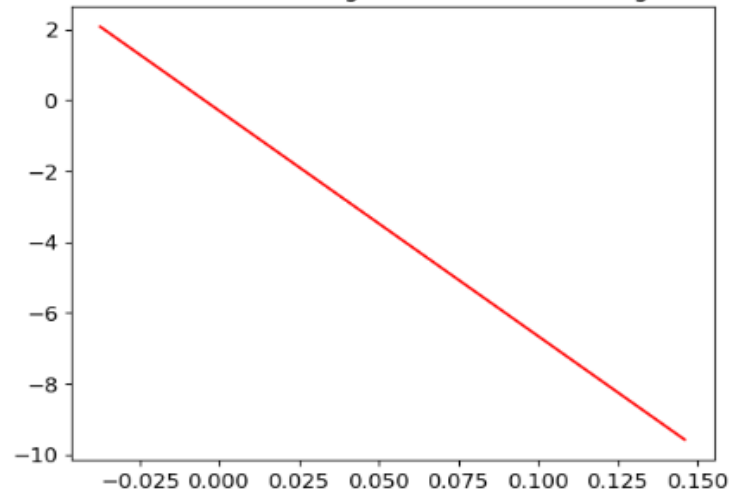
After the execution of all the listed functions above the final output graphs are as follows

a) Direct max analysis


```
[17]: #direct final maximum data and position plot  
read_and_process_final_datasets("directData.csv")
```

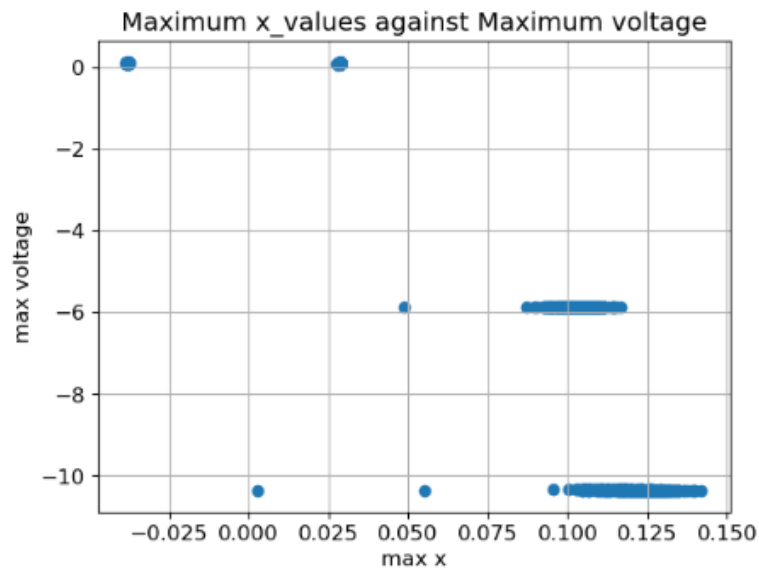


Scatter Plot of Maximum x-values against Maximum Voltage with Best Fit Line

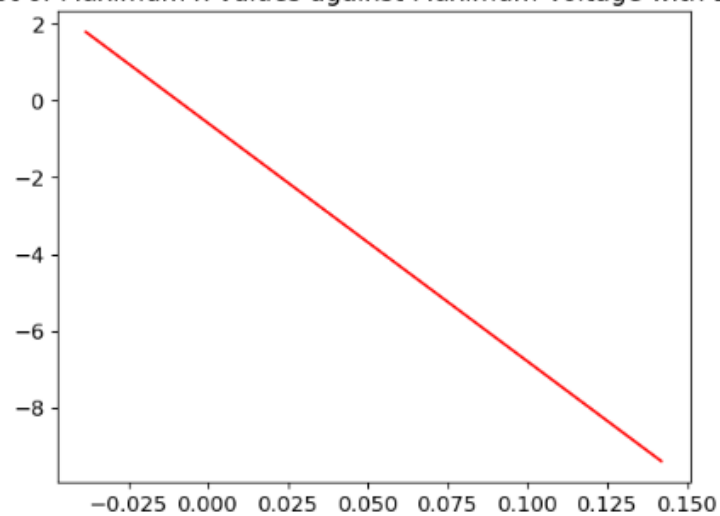


b) Gaussian max analysis

```
[16]: #gaussian final maximum data and position plot  
read_and_process_final_datasets("gaussianData.csv")
```



Scatter Plot of Maximum x-values against Maximum Voltage with Best Fit Line



2.5 Problems faced

- 1) I was able to execute 1207 datasets and 67 failed even after applying the data cleaning techniques mentioned above

2.6 Solution

For now, due to limited time I haven't addressed this issue the major thing I did was to identify and extract this error files in a folder called errorDatasets which I'll attach to the code folder and send.

- 2) The code takes long to execute given the lengthy datasets and the 67 data sets that could not run gave me a hard time.

2.7 Solution

I had to run the code identify the datasets that are working reduce them and put in a separate folder while putting the failed datasets in another folder. I repeated this process for more than a hundred times which is inefficient so it's this morning I thought of implementing a function which could identify and skip such datasets dough I haven't implanted this function yet.

3 C++ code implementation

3.1 Summary of previous Code Functionality

The previous code could

- ✓ Initializes an ETL (Extract, Transform, Load) object with the provided command-line arguments.
- ✓ Reads the CSV data set using the readCSV() method of the ETL object.
- ✓ Extracts the maximum voltage and corresponding current from the data set.
- ✓ Converts the CSV data to an Eigen matrix and calculates the standard deviation and mean.
- ✓ Extracts the values from the second column of the data set and calculates the mean and standard deviation.
- ✓ Calculate the Gaussian peak value and position using the calculated mean and standard deviation.
- ✓ Outputs the Gaussian fit results to the console.

3.2 Improvements

3.2.1 Additions

I added two global data storage variables for Gaussian and direct method results as seen below to store multiple dataset maximum values and their positions.

```
--  
13 // Global data storage for Gaussian and direct method results  
14 std::vector<std::vector<std::pair<double, double>>> all_gauss_data; // Store multiple dataset maximum values and their positions  
15 std::vector<std::vector<std::pair<double, double>>> all_direct_data;
```

I added some functions to the existing ones. These added functions are

1. **save_data()**: Saves data to a specified file. Its purpose is to save a vector of `std::pair<double, double>` to a file, appending to the file if it already exists. The code stores the results of both the Gaussian fitting and direct method calculations in global vectors

(all_gauss_data and all_direct_data). This allows for the analysis of multiple data sets and the potential visualization of the results.

Its implementation is as seen below

Screenshot:

```
17 void save_data(const std::string& fileLocation, const std::vector<std::pair<double, double>>& dir) {
18     std::ofstream out_file(fileLocation, std::ios::app);
19     if (!out_file.is_open()) {
20         std::cerr << "Error opening output file: " << fileLocation << std::endl;
21         return;
22     }
23     for (const auto& pair : dir) {
24         out_file << pair.first << "," << pair.second << std::endl;
25     }
26     out_file.close();
27 }
```

2. appendDataToAllDataSets(): Appends a pair of values (like a peak value and its position) to a specified global vector like the all_guass_data or all_direct_data variables above

its implementation is as seen below

Screenshot:

```
28 void appendDataToAllDataSets(double max_value, double peak_pos, std::vector<std::vector<std::pair<double, double>>> &library){
29     std::vector<std::pair<double, double>> current_data;
30     current_data.push_back(std::make_pair(max_value, peak_pos));
31     library.push_back(current_data);
32 }
```

I also ensured that the code writes the Gaussian fit and direct method results to separate output files, enabling further analysis and comparison of data following the procedure below.

3.2.1.1 *Storing Direct Method Results*

- I created a vector dir_max_and_x_value to store the maximum value (peak_voltage) and its corresponding position (max_current) obtained from the direct method calculations.

- I also used the `std::make_pair()` function to create a pair of these values, and the pair is then pushed into the `dir_max_and_x_value` vector.
- I then called `appendDataToAllDataSets()` function to append the direct method results to the global `all_direct_data` vector.

3.2.1.2 *Storing Gaussian Fit Results*

- I created another vector `gaussian_max_and_x_value` to store the Gaussian peak value (`gaussian_peak_value`) and its corresponding position (`peak_current`) obtained from the Gaussian fitting calculations.
- Similar to the direct method, I created a pair of these values using `std::make_pair()` and pushed into the `gaussian_max_and_x_value` vector.
- The `appendDataToAllDataSets()` function is called to append the Gaussian fit results to the global `all_gauss_data` vector.

3.2.1.3 *Saving to Output Files*

- The `save_data()` function is called twice to save the direct method and Gaussian fit results to separate output files, "`dirMaxOutput.txt`" and "`Gaussoutput.txt`", respectively as seen below.

Screenshot:

```

64 |
65 |     std::cout << "Gaussian fit mean (mu): " << mu << ", standard deviation (sigma): " << sigma << std::endl;
66 |     std::cout << "Gaussian peak value is: " << gaussian_peak_value << ", occurring at x = " << peak_current << std::endl;
67 |     std::cout << "Better approximation: " << (peak_current + max_current) / 2 << std::endl;
68 |
69 |     // storing the maximum value for the direct method and the gaussian fit
70 |     std::vector<std::pair<double, double>> dir_max_and_x_value;
71 |     dir_max_and_x_value.push_back(std::make_pair(peak_voltage, max_current));
72 |     appendDataToAllDataSets(peak_voltage, max_current, all_direct_data);
73 |     std::vector<std::pair<double, double>> gaussian_max_and_x_value;
74 |     gaussian_max_and_x_value.push_back(std::make_pair(gaussian_peak_value, peak_current));
75 |     appendDataToAllDataSets(gaussian_peak_value, peak_current, all_gauss_data);
76 |
77 |     save_data("dirMaxOutput.txt", dir_max_and_x_value);
78 |     save_data("Gaussoutput.txt", gaussian_max_and_x_value);
79 |

```

3.3 Issues faced

I tried implementing the same steps like in the python code but face compiler issues which I finally had to change my compiler. Some libraries I have to use to be able to do file operations for example `<filesystem>` is not supported by the current version of my compiler so I had to update my compiler but when I tried to do so and then edit the include path the whole code crashed and even the `<iostream>` library wasn't recognized anymore meaning I could not even run a simple hello world program. I tried to fix this issue countless times but I haven't succeeded yet so today I switch from running my code on visual studio code to dev cpp the basic libraries can be recognized and I made some updates to the code but I don't have the final expected output graph as am still to update the compiler path so I can use the `<filesystem>` library which permits me to loop in a folder and read the datasets.

If the files are ran one after another their outputs will be collected and stored accordingly but this will mean the code will be ran for a 1k+ times which is not efficient enough.

4 General Conclusion

We have seen how the given task has been handled so far in both python and c++, the issues faced and some attempted solutions. Concerning the c++ code I'll try to complete it as soon as possible after I succeed to update my g++ version.

User guide.

To run the code in c++ the commands are

It has not been an easy working week given the bulk of work but so far this is a summary report of what I was able to archive

5 References

- Google/ <https://www.bloodshed.net/>
- <https://chat.openai.com/>
- Friends
- YouTube/