**CS-112 Final Semester Project**

**Clustering – Unsupervised Learning**

**Muhammad Ahmad Amjad 2023361**

**Abdullah Ejaz Janjua 2023038**

**Instructor: Prof. Dr. Zahid Halim**

**Faculty of Computer Science**

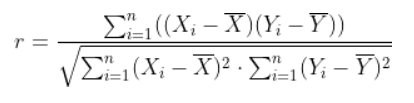
In this project, we implemented a clustering technique using an Object-Oriented Programming (OOP) approach in C++/CLI. The project consisted of three main tasks, and in this write-up, we will explain each step, provide relevant screenshots, and discuss the work distribution among team members.

**Task 1: Calculation Correlation Matrix and Discretization**

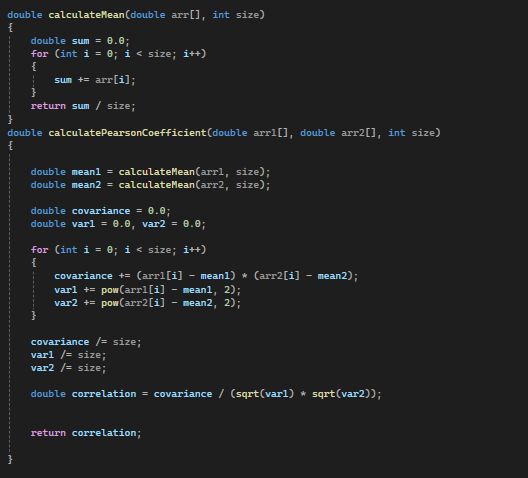
**Step 1: Calculation of Correlation Matrix**

We created a correlation matrix from the given data matrix using Pearson's correlation coefficient. The correlation matrix is an N×N matrix, where N is the number of records in the input dataset. Each element in the correlation matrix represents the Pearson's correlation coefficient between the corresponding rows in the data matrix.

The Pearson's correlation coefficient formula used is:

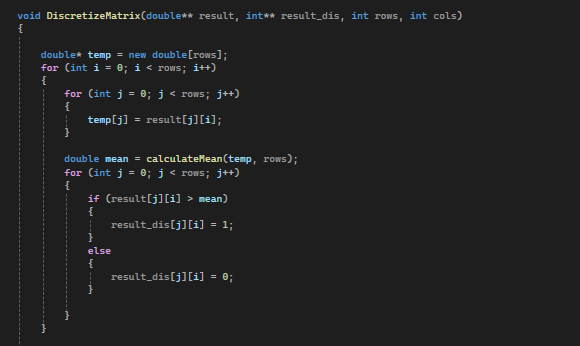


The correlation matrix was calculated using the **CalPearsonCoefficeint function** in the Form1.h file. This function takes in the original data matrix (dataSet) and an empty matrix (result) to store the correlation coefficients.



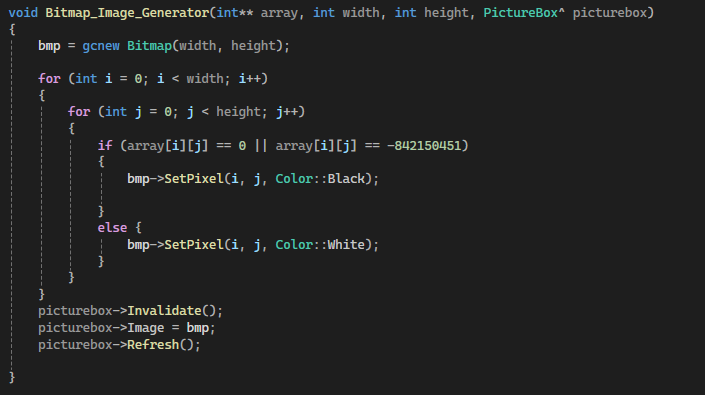
**Step 2: Discretization**

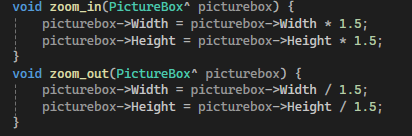
After calculating the correlation matrix, we discretized it by calculating the mean of each column. All values in that column above the calculated mean were set to 1, and the rest were set to 0. The discretization step is performed in the **DiscretizeMatrix function:**



**Step 3: Visualization**

We converted the discretized matrix into a bitmap image, providing functionality for zooming in and out. The **Bitmap\_Image\_Generator function** is used to generate the bitmap image from the discretized matrix:

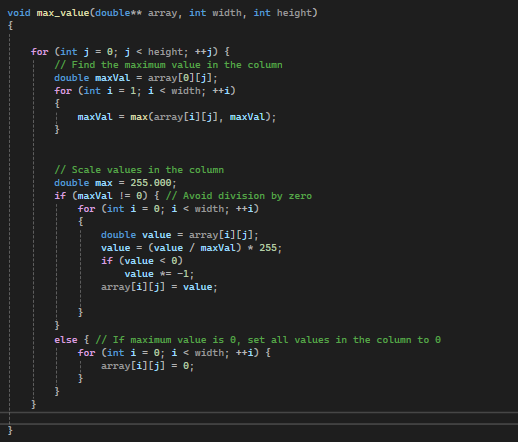




Additionally, we displayed a color-coded image of the similarity matrix by following these steps:

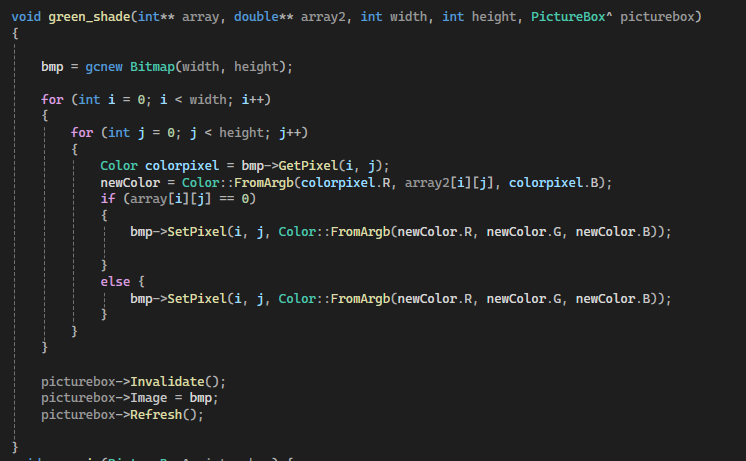
1. For each column in the matrix, find the maximum value.
2. Divide each value in the column by the maximum value and multiply it by 255.
3. The resulting values will be in the range of 0 to 255.
4. Use these values to apply a green shade to the corresponding pixels in the image

The **max\_value function** is used to scale the values in the correlation matrix for color-coding:

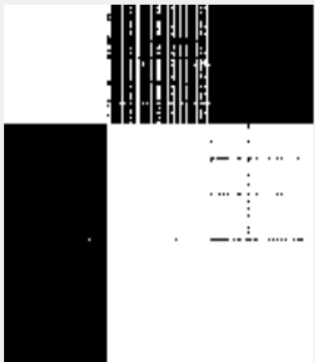


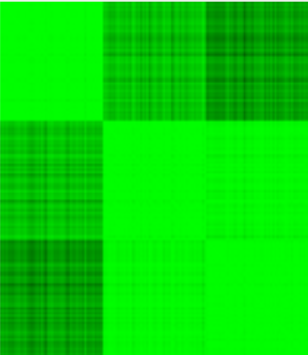
.

Finally, **the green\_shade function** is used to apply the green shade to the image based on the scaled values



**Screenshots for Task 1:**



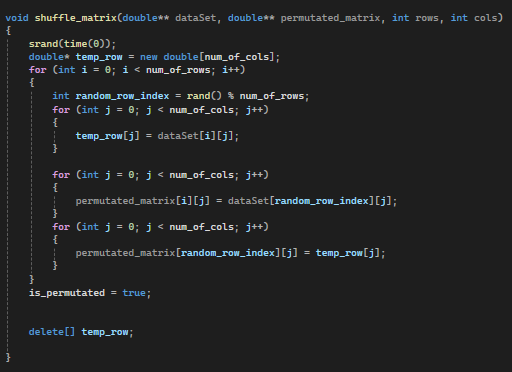


**Task 2: Permutation and Signature Technique**

**Step 1: Permutation of the Data Matrix**

We permuted the data matrix by shuffling the individual rows in the dataset.

The **shuffle\_matrix function** is used to permute the data matrix by shuffling the rows:



**Step 2: Display of Permuted Data Matrix**

The color-coded image of the permuted data matrix was displayed using the same visualization functions as in Task 1.

**Step 3: Signature Technique**

To recover the image clusters, we used the signature technique. For each row in the permuted matrix, we followed these steps:

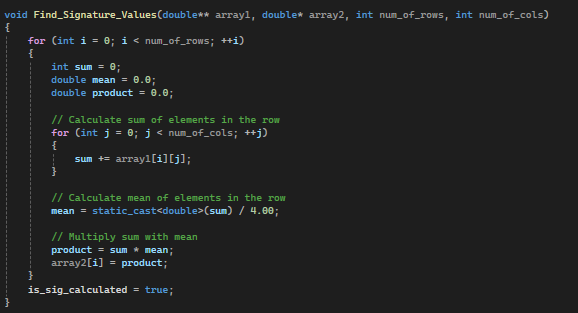
1. Sum all the values in the row.

2. Calculate the mean of the row.

3. Multiply the sum of the row with its mean.

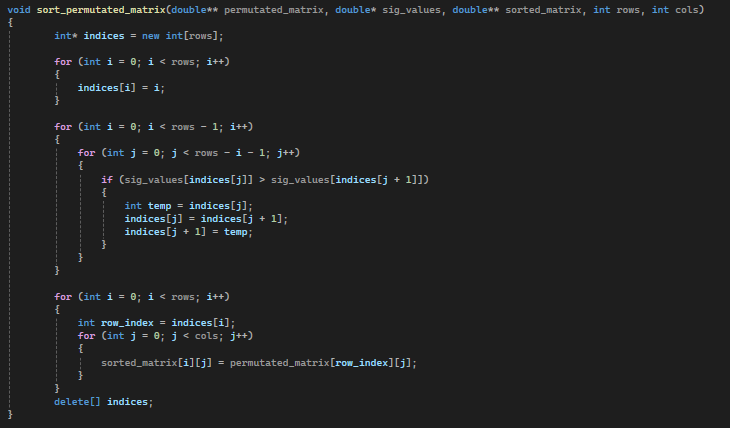
The above three steps produced a signature value for each row.

The signature technique was implemented in the **Find\_Signature\_Values function:**



**Step 4: Rearrangement and Visualization**

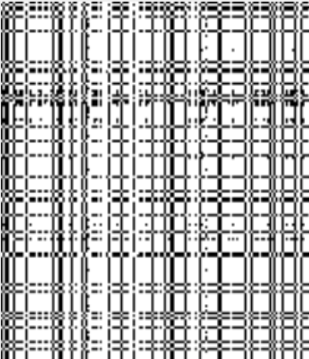
The **sort\_permutated\_matrix function** was used to rearrange (sort) the similarity matrix based on the signature values:

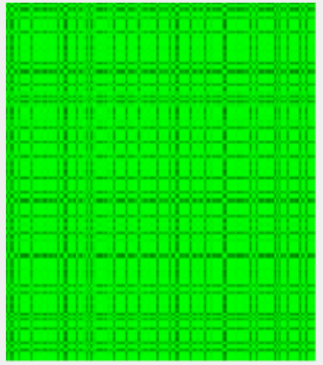


After the matrix was sorted, Task 1 (calculation of the correlation matrix, discretization, and visualization) was applied to the rearranged matrix, and the color-coded image was displayed.

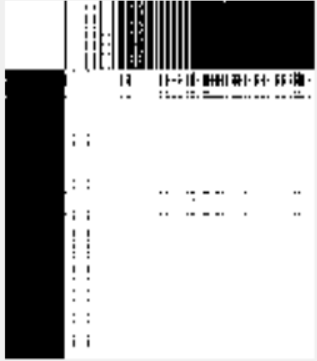
**Screenshots for Task 2:**

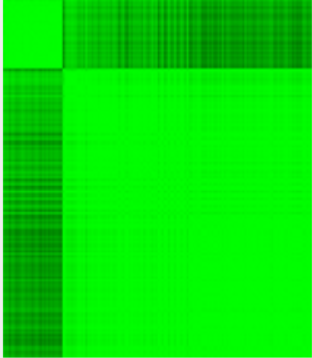
**Permutated:**





**Sorted:**

****

****

**Work Distribution:**

The implementation of **Task 1** was a collaborative effort between **Muhammad Ahmad Amjad (2023361)** and **Abdullah Ejaz Janjua (2023038).** Initially, Ahmad Amjad implemented the Bitmap\_Image\_Generator function for generating black and white bitmap images, as well as the green\_shade function for color-coding the images. Meanwhile, Abdullah Ejaz Janjua focused on implementing the Pearson's correlation coefficient calculation and then both worked on reading the data from the input file.

The calculations were first carried out in Visual Studio Code, and then the code was integrated into the Visual Studio project. During the integration process, issues arose with the file reading functionality, which required extensive debugging efforts from both team members. They worked together, printing various outputs and tracing the code execution to identify and resolve the root cause of the problem successfully.

For **Task 2**, Muhammad Ahmad Amjad took the lead in implementing the signature value calculation and the associated Find\_Signature\_Values function. On the other hand, Abdullah Ejaz Janjua focused on implementing the permutation of the data matrix through the shuffle\_matrix function. Both team members collaborated on the sorting algorithm used in the sort\_permutated\_matrix function, bouncing ideas off each other and working together to ensure its correct implementation.

After successfully meeting the project requirements, the team shifted their focus towards creating a user-friendly interface. They exchanged ideas and suggestions, ultimately developing an intuitive GUI that seamlessly integrated the project's functionalities.

Throughout the project, Muhammad Ahmad Amjad and Abdullah Ejaz Janjua maintained a strong collaborative approach, dividing tasks and responsibilities evenly. They worked closely together, debugging issues, and ensuring the successful integration of each component. Their effective teamwork and equal contributions were instrumental in the successful completion of this project.