**CS-221 Data Structure & Algorithm**

**Semester Project Report**

**Dr. Zahid Halim**

**Ghulam Ishaq Khan Institute of Engineering Sciences & Technology, Topi, Swabi 23460, Pakistan**

**Logo

Description automatically generated**

**Clustering Techniques**

Group Members:

* **Muhammad Rehan (2021438)**
* **Adeen Amir (2021051)**
* **Zaid Bin Muzammil (2021718)**
* **Muhammad Arsal Sheikh (2021355)**

**Language used:**

Python

**Libraries used:**

* Pandas
* Matplotlib
* OpenCV
* Numpy
* NetworkX

**Introduction:**

Designed a program which uses two types of clustering techniques, and the application can be used for any dataset.

The first step was to read text file that had the iris dataset and convert it into a csv file and reading the csv file into our program using python data manipulation library called pandas.

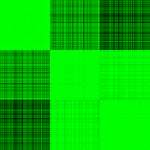
**Task 1:**

The correlation matrix of the data was computed through Pearson’s correlation coefficient. We did this by taking the transpose of the iris dataset’s data frame and using built-in function to calculate the Pearson’s correlation. The mean of each column was calculated by using the built-in, “.mean()” function. Image of the discretized matrix was produced using matplotlib.

**Discretized Matrix:**



To produce a color-coded image, the instruction in the task were followed, and to give the green shade to the pixels in similarity matrix, OpenCV was used to set the reds and blues to zero which left behind the green shade. Following is the image of similarity matrix with green shape:



**Task 2:**

We obtained the permuted matrix by shuffling the rows among each other, of the original iris dataset. Numpy shuffle method was used to randomize the rows. Shuffling the original matrix of 150 x 4 to obtain permuted matrix does not change the dimensions of this matrix. Therefore, the 150 x 4 matrix image looks like this:



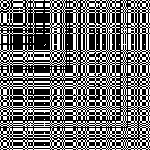
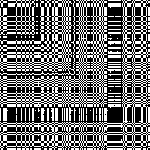
Correlation matrix of this shuffled data is now created by sending the permuted matrix to the Pearson Correlation function. The permuted correlation matrix is further used to generate the signature by following the mentioned steps. The correlation matrix is converted into data frame and signatures column is appended to this data frame. Based on this signatures’ column, the values of the permuted correlation matrix are sorted in an ascending order. The resulting matrix is sent to the Task1 function which produces the discretized and color-coded image of the matrix. Since each time the program runs, the matrix is shuffled randomly, it results in different matrix values each time. Here are the images below of two permuted correlated matrix that were generated during testing:

**Bitmap of iris data Before Permutation**

A picture containing diagram

Description automatically generated

**Bitmap of iris data After Permutation**



**Discretized matrix after performing task one on signature correlation matrix:**

Application, calendar

Description automatically generated with medium confidence

**Green color-coded image after performing task one on signature correlation matrix:**



**Task3:**

In Task 3, firstly, Permutation is done on the original data set and once all the rows are randomized, we apply Pearson correlation on our matrix which results in a NxN (150x150) permuted adjacent matrix.

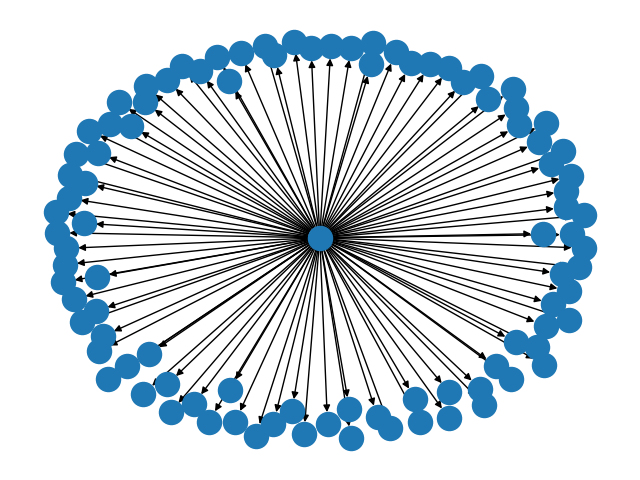
Next, user is asked for a threshold input, which makes all the values below the threshold in the matrix equal to zero. Error handling has been done if the user enters a value outside of the range (lowest value in matrix to one only accepted).

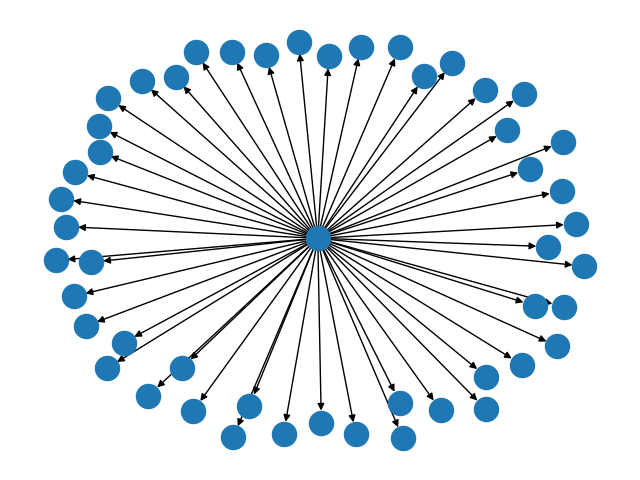
Once the required values are set to zero due to user’s threshold input. We find the weight of the node for each row which is the sum of weights of all the edges connected to it. Then, we search for the highest weight in the entire array and start our cluster from that row.

We then print an image of the highest weighted node using NetworkX which gives us an image in the form of graph where our highest weighted index is the main node, and all the edges are connected from there. The picture produced is now our first cluster. Next, we traverse the highest weighted node’s row and columns and equate it to zero. Similarly done for the node’s neighbors.

This process is repeated until all the rows and columns are not zero.

Depending on the threshold (using **0.98**), images are produced with decreasing number of edges in every traverse:





Chart, line chart

Description automatically generatedLine chart

Description automatically generated with low confidence

**Comparison of task 2 and 3:**

In task two, we permute the original dataset matrix ­and then apply signature technique to recover the image clusters, and then visualize the image.

In task three, we again permute the original dataset matrix and then use graphs to find the highest weighted node and its immediate neighbors and count that as our clusters. This process is repeated until there are no more clusters left. These clusters are then visualized using graphs. In every repetition, the size of the cluster decreases.

The result of both the tasks is the same, but in task two, we only recover the clusters from the permuted dataset while, in task three, we extract the clusters from the permuted dataset and visualize them.

**Work Distribution among group members:**

Task1, Task2, Task3 : work distributed equally among group members.

Task 4: Adeen Amir 2021051 and Zaid Bin Muzammil 2021718.

Image conversion/ Processing : Muhammad Arsal 2021355.

GUI : Muhammad Rehan 2021438

Everyone was present at the time of making the program and contributed to logic building.