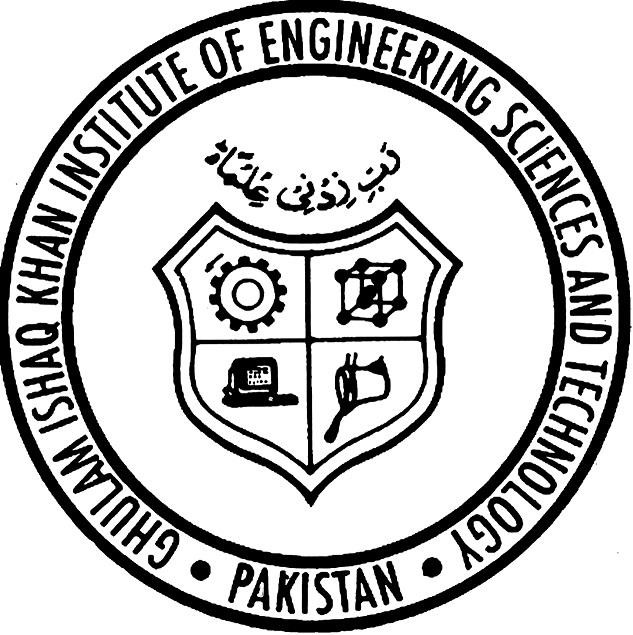
­­­**Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi**



**Course: Introduction to AI**

**Instructor: Sir Zahid Halim**

**Assignment 2**

**Reg no: 2020526**

**Name: Zartaj Asim**

**Introduction**

The objective of this project was to implement various clustering techniques on a given dataset using suitable data structures. The dataset provided was in a specific format, where the first row contained the number of rows, the second row contained the number of columns, and the rest of the rows contained the actual data.

**Task1**

The first task involved calculating the correlation matrix using Pearson's correlation coefficient and then discretizing it using mean/median. The resulting discretized matrix was converted into a bitmap and displayed as a color-coded image.

**Task 2**

The second task involved permuting the data matrix by shuffling the individual rows, displaying the color-coded image of the permuted matrix, and recovering image clusters using a signature technique. The similarity matrix was rearranged by signature value, and Task 1 was applied to the rearranged matrix, followed by displaying the color-coded image.

**Task 3**

The third task involved creating a weighted graph for the permuted data set, removing edges with weights below a certain threshold, and creating clusters based on the weight of nodes. Each extracted cluster was visualized.

**Breakdown of Task 1**

After reading in the dataset, the code creates a heatmap of the correlation matrix using the Seaborn library. The correlation matrix is calculated using Pearson's correlation coefficient. Then, the code calculates the mean and median of each column of the correlation matrix and discretizes the values by setting all values above the calculated mean or median to 1 and the rest to 0. Two binary matrices, "Mean\_graph" and "Median\_graph", are created to visualize the discretized correlation matrix using the "imshow" function.

The code visualizes the discretized correlation matrix using a green color map. The color of each pixel represents the value of the correlation coefficient, with brighter pixels indicating higher values.



**Breakdown of Task 2**

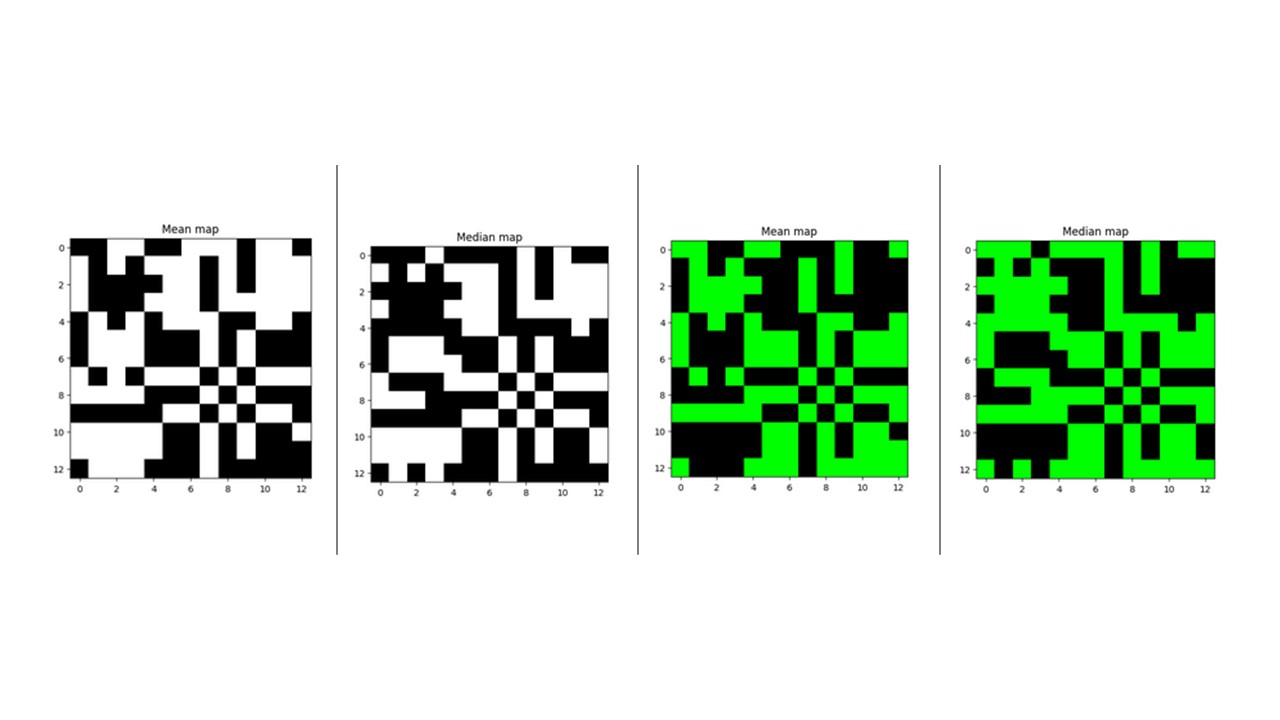
For each dataset, it checks if a column can be divided by 2 without error; if so, it is assumed to be a numeric column, and nothing is done. If an error occurs, it is assumed to be a categorical column and is transformed using LabelEncoder().the code generates a heatmap of the Pearson correlation coefficients between the columns in each dataset using the seaborn package. The resulting heatmap is displayed using matplotlib. After that, the code converts the correlation matrix to a binary adjacency matrix, where each entry is either 0 or 1 depending on whether the correlation coefficient is below or above the mean/median of the row. Two adjacency matrices are generated, one for the mean values and another for the median values.

The adjacency matrices are displayed as binary images using matplotlib, with white representing 0 and green representing 1. The images represent the correlation structure of the dataset, where a green square indicates a strong positive correlation between two variables, and a white square indicates a weak or negative correlation.



Qr code

Description automatically generatedA picture containing text, crossword puzzle

Description automatically generated

**Breakdown of Task 3**

the permuted dataset is loaded and the correlation matrix is calculated using the corr() function. The user is prompted to enter a threshold for the edge weights, below which edges are removed by setting the corresponding values in the correlation matrix to zero.

Next, a weighted graph is created where each node has a weight equal to the sum of weights of all the edges connected to it. The weights of the nodes are calculated by summing the values in each row of the correlation matrix. The graph is initialized as a 2D array of zeros, and then populated with the node weights where an edge exists in the correlation matrix. Clusters are then extracted using a greedy algorithm, where the node with the highest weight is selected as the first cluster, and all its neighbors are added to the same cluster. This process is repeated until there are no more nodes with non-zero weights left.

Finally, each extracted cluster is visualized using a heatmap. The sns.heatmap() function is used to plot the correlation matrix of the nodes in each cluster. The plt.title() function is used to label each cluster with its cluster number.