

**SWE2009: DATA MINING AND TECHNIQUES**

**A report**

**By**

**Sanket Kisan Chavhan 22MIS1116**

**DIGITAL ASSIGNMENT 2:** Implement all types of clustering

using any one tool except Python/Google co-lab. One page

about your implementation algorithms/dataset details and

excepted output along with GitHub link.

**NAME OF THE FACULTY:** DR. PATTABIRAMAN V

**Language and platform used**: R and R studio.

**Link for GitHub**: https://github.com/sanket9673/DataMining2

**DATASET DETAILS**

The dataset contains information about the performance of senior students from a large fictional high school at the end of their final semester. Each student is uniquely identified by an ID, which is necessary as there may be multiple students with the same name. The dataset includes various attributes such as the student's first name, last name, email address, and gender. Additionally, it provides scores for subjects including mathematics, history, physics, chemistry, biology, English, and geography, each ranging from 0 to 100. Overall, the dataset offers insights into the academic achievements of senior students and factors that may influence their grades, providing valuable information for analysis and decision-making in the educational context.

**IMPLEMENTATION ALGORITHMS:**

**1. K-Mean:**

1. **Preparation**
   * Install and load the required R packages: ClusterR and cluster.
   * Import the student scores dataset from the specified file path.
2. **Data Preprocessing**
   * Select the relevant numerical columns for clustering (e.g., scores in various subjects).
   * Standardize the data to ensure equal weightage during clustering.
3. **Clustering Execution**
   * Set a random seed for reproducibility of results.
   * Apply the K-Means clustering algorithm to the standardized data, specifying the number of clusters and the number of starts for initialization.
4. **Cluster Assignment**
   * Retrieve the cluster assignments for each observation from the K-Means result.
5. **Model Evaluation and Visualization**
   * Perform Principal Component Analysis (PCA) on the scaled data to reduce dimensionality for visualization.
   * Create a scatter plot using the first two principal components, coloring the data points according to their cluster assignments.
   * Plot the cluster centers on the PCA scatter plot.
   * Use clusplot to visualize the clusters with a two-dimensional representation of the data.
6. **Post-Processing**
   * Transform the cluster centers using PCA for accurate placement in the plot.
   * Add additional visual elements such as titles, labels, and legends to enhance the interpretability of the plots.

A graph showing a number of student scores

Description automatically generated

**2. K-Medoids:**

1. **Initialization**
   1. Load the necessary R packages: cluster for clustering algorithms and ggplot2 for data visualization.
   2. Import the student scores dataset from the provided file path.
2. **Data Selection and Preprocessing**
   1. Extract the relevant numerical columns intended for clustering, such as absence days, self-study hours, and subject scores.
   2. Normalize the data using the scale function to ensure that each feature contributes equally to the distance calculations.
3. **Clustering Execution**
   1. Set a random seed to ensure reproducibility of the clustering results.
   2. Apply the K-Medoids clustering algorithm to the normalized data, specifying the desired number of clusters.
4. **Cluster Assignment and Medoid Identification**
   1. Obtain the cluster assignments for each student from the K-Medoids result.
   2. Identify the indices of the medoids within the dataset.
5. **Dimensionality Reduction for Visualization**
   1. Perform Principal Component Analysis (PCA) on the normalized data to reduce it to two principal components for a 2D visualization.
   2. Transform the coordinates of the medoids using the PCA rotation matrix.
6. **Visualization**
   1. Create a scatter plot of the students’ data projected onto the first two principal components, coloring points by their cluster assignments.
   2. Overlay the transformed medoid coordinates onto the scatter plot.
   3. Enhance the plot with a minimalistic theme and appropriate labels.
7. **Additional Cluster Visualization**
   1. Use the clusplot function to create a 2D visualization of the clusters, incorporating lines, shading, and color to distinguish between clusters.

A cluster of student scores

Description automatically generated

**3. PAM (Partitioning Around Medoids):**

1. **Initialization**
   1. Load the cluster library in R, which provides functions for cluster analysis.
   2. Import the student scores dataset from the specified file path.
2. **Data Selection and Preprocessing**
   1. Extract the ‘math\_score’ and ‘english\_score’ columns from the dataset for clustering.
   2. Normalize the selected data using the scale function to ensure that each feature contributes equally to the distance calculations.
3. **Clustering Execution**
   1. Set the desired number of clusters for the PAM algorithm.
   2. Perform PAM clustering on the normalized data using the pam function from the cluster library.
4. **Cluster Assignment**
   1. Retrieve the cluster assignments for each observation from the PAM result object.
5. **Visualization Preparation**
   1. Open a new plotting device using dev.new() to visualize the clusters.
6. **Visualization Execution**
   1. Create a scatter plot of the scaled data with points colored according to their cluster assignment.
   2. Overlay the cluster medoids onto the scatter plot using the points function with distinct plotting characters.
   3. Add a legend to the plot to indicate the cluster numbers.

A graph of a diagram

Description automatically generated with medium confidence

**4. CLARA (Clustering Large Applications):**

1. **Initialization**
   1. Load the cluster and factoextra libraries in R, which provide functions for cluster analysis and visualization.
2. **Data Importation**
   1. Import the student scores dataset from the specified file path.
3. **Data Preprocessing**
   1. Extract only the numeric columns from the dataset, which are suitable for clustering.
   2. Standardize the numeric data to ensure that each feature contributes equally to the distance calculations.
4. **Clustering Execution**
   1. Set a random seed to ensure reproducibility of the clustering results.
   2. Perform CLARA clustering on the standardized data, specifying the desired number of clusters (e.g., 3).
5. **Cluster Assignment**
   1. Retrieve the cluster assignments for each observation from the CLARA result object.
6. **Visualization of Clusters**
   1. Use the fviz\_cluster function from the factoextra package to create a scatter plot of the data points with ellipses around the clusters.
7. **Silhouette Analysis**
   1. Plot the silhouette diagram using the fviz\_silhouette function to evaluate the cluster quality and observe how well each object lies within its cluster.

A graph showing a variety of colored dots

Description automatically generated

**5. AGNES (Agglomerative Nesting):**

1. **Initialization**
   1. Load the 'cluster' library in R to utilize clustering functions.
2. **Data Importation**
   1. Read the student scores dataset from the provided CSV file path.
3. **Data Preprocessing**
   1. Subset the dataset to include only the first 100 observations.
   2. Select the 'math*\_score' and 'english\_*score' columns for clustering.
4. **Data Standardization**
   1. Scale the selected columns to normalize the data.
5. **Clustering Execution**
   1. Perform AGNES clustering on the scaled data using the 'average' linkage method.
   2. Convert the AGNES object to an hclust object for further analysis.
6. **Cluster Assignment**
   1. Determine the desired number of clusters (e.g., 3).
   2. Cut the dendrogram at the appropriate level to assign observations to clusters.
7. **Cluster Visualization**
   1. Plot the dendrogram for the AGNES clustering result.
   2. Highlight the clusters on the dendrogram with colored rectangles.
8. **Output**
   1. Print the cluster assignments to review the distribution of observations.

A diagram of a clustering structure

Description automatically generated

**6. DIANA (Divisive Analysis):**

1. Initialization

- Load the 'cluster' and 'factoextra' libraries in R for clustering and visualization functions.

2. Data Importation

- Import the student scores dataset from the specified CSV file path.

3. Data Preprocessing

- Select only the numeric columns from the dataset, which are appropriate for clustering.

4. Clustering Execution

- Perform DIANA clustering on the numeric data.

5. Cluster Determination

- Decide on the number of clusters (e.g., 3) for analysis.

6. Cluster Assignment

- Assign data points to clusters based on the DIANA clustering result.

7. Cluster Visualization

- Visualize the clustering with a dendrogram.

- Enhance the dendrogram visualization using the 'factoextra' package.

8. Silhouette Analysis

- Conduct silhouette analysis to assess the quality of the clusters.

- Plot the silhouette diagram for a visual representation of cluster cohesion and separation.

9. Output

- Print the cluster assignments for each observation.A diagram of a diagram

Description automatically generated

**7. ROCK (Robust Clustering using Links):**

**1. Initialization**

**-** Define functions for calculating Euclidean distance and identifying mutual nearest neighbors.

**2. Data Importation**

**-** Load the student scores dataset from the specified CSV file path.

**3. Data Preprocessing**

**-** Select the numerical columns 'math\_score' and 'english\_score' for clustering.

**- Normalize the data using the scale function.**

**4. Clustering Execution**

**-** Define and execute the ROCK clustering algorithm with the specified number of clusters and neighborhood radius.

**5. Cluster Assignment**

**-** Assign data points to clusters based on mutual nearest neighbors within the given radius.

**6. Cluster Visualization**

**-** Plot the normalized data points with cluster assignments.

- Add a legend to the plot to identify the clusters.

**7. Clustering Validation**

**-** Check for any NA or infinite values in the data.

- Verify that all data points have been assigned to a cluster.

**8. Output**

**-** Print the cluster assignments for each observation.A graph of a student score clustering

Description automatically generated

**8. Hierarchical Clustering:**

1. **Initialization**
   1. Load the 'cluster' library in R for access to clustering functions.
2. **Data Importation**
   1. Read the student scores dataset from the specified CSV file path.
3. **Data Preprocessing**
   1. Select only the numeric columns from the dataset for clustering.
   2. Remove any rows with missing values (NAs) to ensure data integrity.
4. **Dissimilarity Matrix Computation**
   1. Calculate the dissimilarity matrix for the numeric data using Euclidean distance.
5. **Clustering Execution**
   1. Perform hierarchical clustering on the dissimilarity matrix using Ward's method.
6. **Cluster Formation**
   1. Cut the resulting dendrogram to form a specified number of clusters, such as 5.
7. **Cluster Assignment**
   1. Assign each observation to one of the 5 clusters based on the dendrogram cut.
8. **Cluster Visualization**
   1. Plot the dendrogram and add rectangles to visually distinguish the 5 clusters.
9. **Output**
   1. Print the cluster assignments for each observation for review and analysis.

A diagram of clustering data

Description automatically generated

**9. CHAMELEON:**

1. **Initialization**
   * Load the 'dbscan' library in R for clustering functions.
2. **Data Importation**
   * Import the student scores dataset from the specified CSV file path.
3. **Data Preprocessing**
   * Subset the dataset to include only the first 100 observations.
   * Select the 'math*\_score' and 'english\_*score' columns for clustering.
4. **Data Standardization**
   * Scale the selected data to normalize the feature scales.
5. **Clustering Execution**
   * Perform DBSCAN clustering on the scaled data with specified 'eps' and 'minPts' parameters.
6. **Cluster Visualization**
   * Plot the scaled data points with different colors representing each cluster.
   * Add a legend to the plot to identify the clusters.
7. **Cluster Analysis**
   * Print the count of points in each cluster.
   * Output the cluster assignments for each data point.
8. **Title Correction**
   * Note: The main title of the plot in the code mentions "CHAMELEON Clustering," which should be corrected to "DBSCAN Clustering" to reflect the actual algorithm used.

A graph showing a number of students scores

Description automatically generated

**10.DBSCAN (Density-Based Spatial Clustering of Applications with Noise):**

**1. Load Libraries:**

- Load 'fpc', 'dendextend', and 'cluster' libraries for clustering operations.

**2. Data Importation:**

- Import dataset from the specified file path using read.csv() function.

**3. Data Preprocessing:**

- Ensure dataset contains only numeric columns suitable for clustering.

- Convert factor columns to numeric using one-hot encoding with model.matrix().

**4. Distance Computation:**

- Calculate the distance matrix for the numeric data with dist() function.

**5. DBSCAN Clustering:**

- Set parameters 'eps' and 'MinPts' for DBSCAN.

- Apply dbscan() function to perform clustering on the numeric data.

**6. Dendrogram Plotting:**

- Perform hierarchical clustering with hclust() and method 'ward.D2'.

- Convert to dendrogram and plot using plot() function.

7. Cluster Visualization:

- Plot DBSCAN clusters with plot() function.

- Visualize clusters on a scatter plot, specifying columns for axes.

**8. Legend Addition:**

- Add a legend to the scatter plot with unique cluster identifiers.

A diagram of clustering data

Description automatically generated