National University of Computer and Emerging Sciences

Data Mining

Spring 2024

Assignment #2 - Clustering

Due Date: Sunday, April 14th by 11:59 pm

Instructor: Mr. Basharat Hussain & Mr. Muhammad Farrukh Bashir

Faraz Razi
i201866
Section: A

Data Overview:

The dataset provided for the study consisted two collections. One has the track trajectories and other is points on each trajectory. Following is short description of each feature:

1. go_track_tracks.csv:

- id_android - it represents the device used to capture the instance;

- speed - it represents the average speed (Km/H)

- distance - it represent the total distance (Km)

- rating - it is an evaluation parameter. Evaluation the traffic is a way to verify the volunteers perception about the traffic during the travel, in other words,

- if volunteers move to some place and face traffic jam, maybe they will evaluate 'bad'.

3 - good,

2 - normal,

1 - bad.

- rating_bus - it is other evaluation parameter.

1 - The amount of people inside the bus is little,

2 - The bus is not crowded,

3 - The bus is crowded.

- rating_weather - it is another evaluation parameter.

1 - raining.

2 - sunny,

- car_or_bus

1 - car,

2 - bus

- linha - information about the bus that does the pathway

2. go_track_tracks.csv:

- id: unique key to identify each point

- latitude: latitude from where the point is

- longitude: longitude from where the point is

- track_id: identify the trajectory which the point belong

- hour: datehour when the point was collected (GMT-3)

Data Visualization:

Correlation:

The **Error! Reference source not found.** shows the data features relation with each other. Here the derived coulmns show strong coorelattion. The Most of the features not highly coorelated.

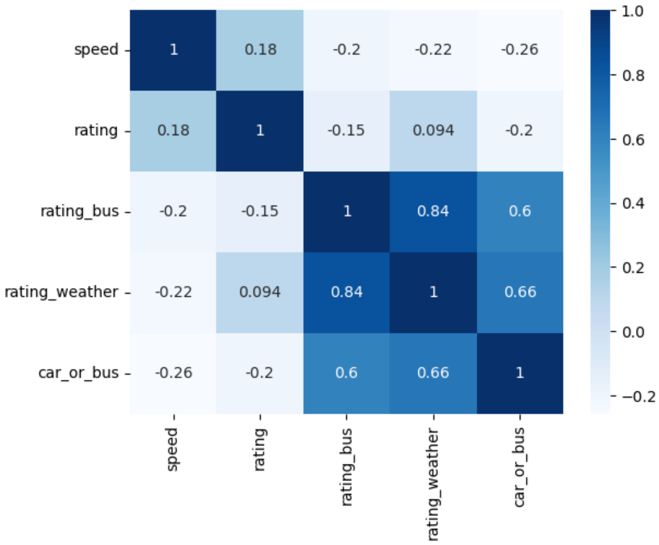

Figure 1 – Correlation

Pairplot:

The Figure 2 - Pairplot shows the whole dataset overview and distribution.

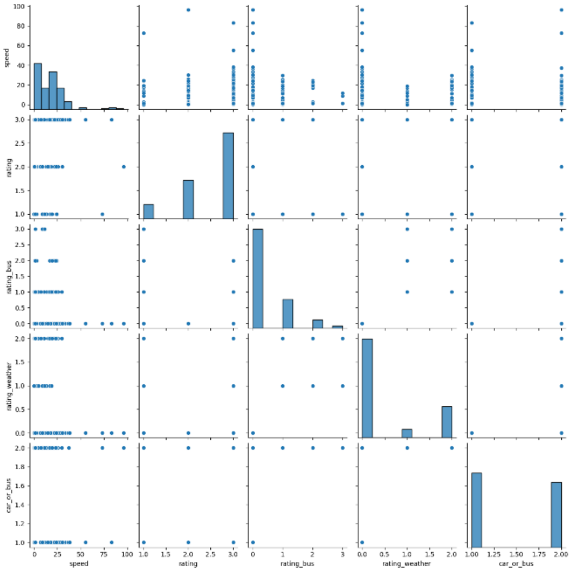

Figure 2 - Pairplot

Data Preprocessing:

Steps:

1. Merge:

   Both data frames were merged based on the track ids. We take “id” from track list and “track_id” from track points. This give us single merged dataset.

2. Time slots:

   Next the time from points data frame is used to generate time slots. Following is the condition for time slot selection.

   ```
   # Convert time to slots (morning, afternoon, evening)
   def convert_time_to_slot(hour):
         if hour < 12:
               return 1 # "morning"
         elif hour < 18:
               return 2 # "afternoon"
         else:
               return 3 # "evening"
   ```

3. Zones division:

   The longitude and latitude are used to create zones for the dataset. The Figure 3 - Dataset Points Plotting has colored points which represent zones they are in.

Plotting Data points:

After the final Data processing steps, we get to this state where we can plot all the data points on a 2D map of longitude and latitude. The Figure 3 - Dataset Points Plotting shows data points plotted between Latitude and longitude.

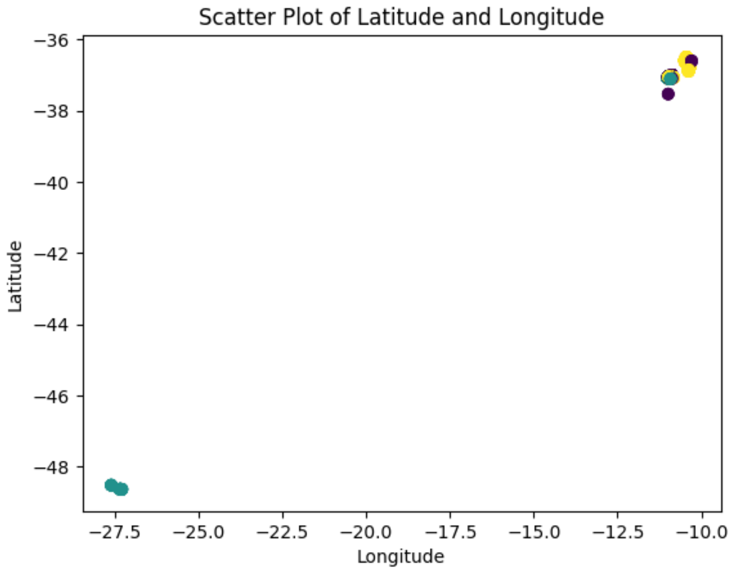

Figure 3 - Dataset Points Plotting

Task 1: Implementing the ROCK Algorithm

The implementation for ROCK algorithm in my case was too slow and taking too much time so I decided to go for a subset of data. This subset successfully encapsulate the trends of the dataset.

Algorithm:

```python
# rock clustering
from pyclustering.cluster import cluster_visualizer
from pyclustering.cluster.rock import rock

# Load list of points for cluster analysis.
sample = tracks_sample[["time_slot", "latitude", "longitude"]].values

# Set ROCK parameters
eps = 0.5  # Maximum diameter of the neighborhood to search for the cluster
threshold = 0.9  # Threshold parameter for ROCK algorithm
number_clusters = 3  # Number of clusters to generate

# Perform clustering using ROCK algorithm
rock_instance = rock(sample, eps, number_clusters, threshold, ccore=True)
rock_instance.process()

# Obtain results of clustering
clusters = rock_instance.get_clusters()

# Visualize clustering results
visualizer = cluster_visualizer()
visualizer.append_clusters(clusters, sample)
visualizer.show()

```

Results:

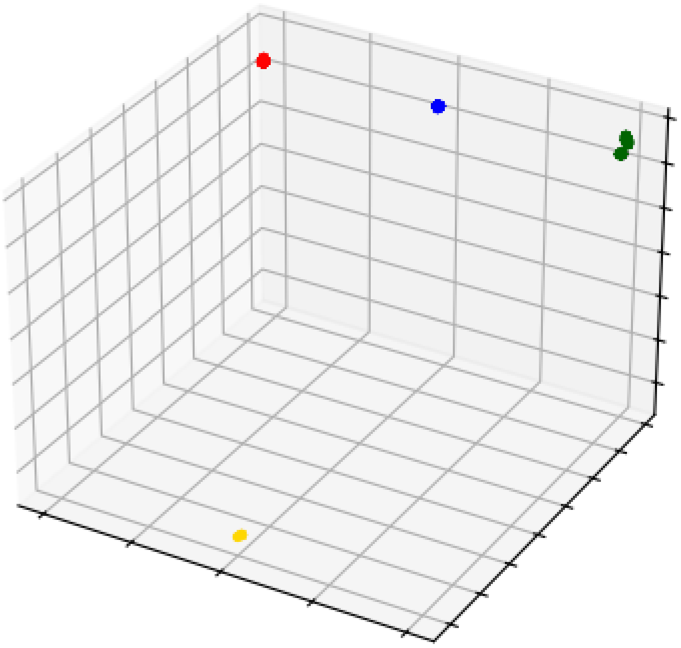

Figure 4 - Rock Algorithm Results

Task 2: Implementing the Chameleon Algorithm

The Chameleon Algo consists of three stages.

First is Graph Partitioning: For this we used K-means, Following are the results of k-means clustering. Figure 5 - Chameleon Kmeans (Full) shows the full graph generated by kmeans algo.

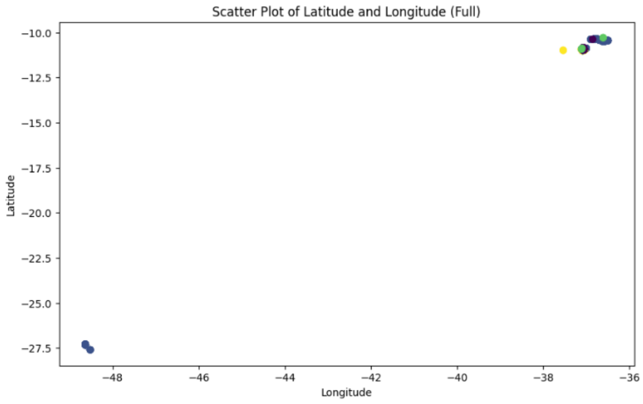

Figure 5 - Chameleon Kmeans (Full)

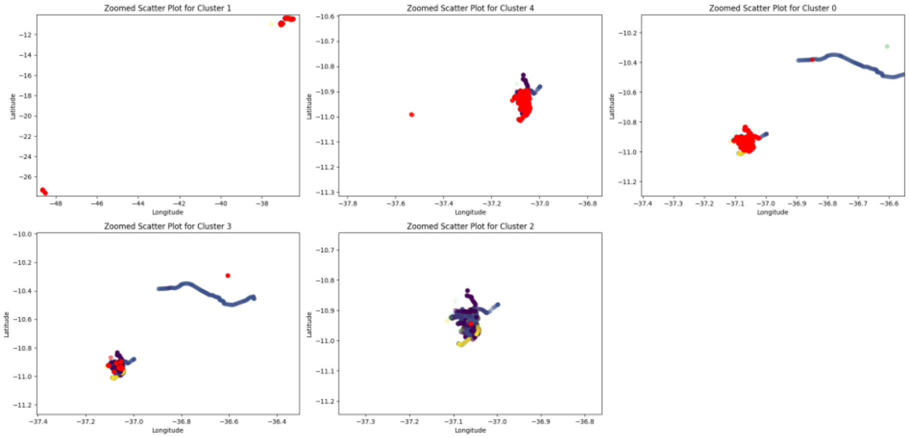

Figure 6 - Chameleon K-means (Sub Plots)

Second is Agglomerative Clustering: Following are the results of k-means clustering. shows the full graph generated by Agglomerative clustering algo.

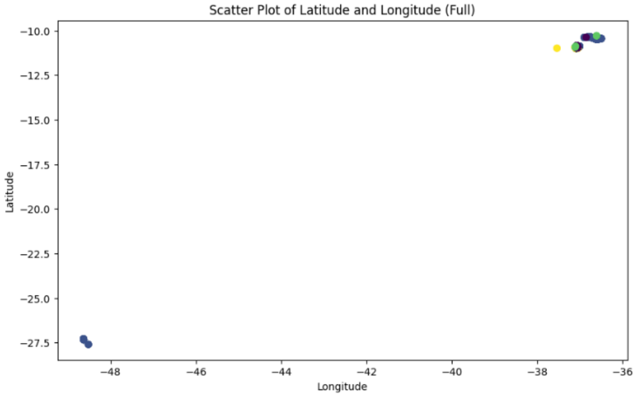

Figure 7 - Chameleon Agglomerative (Full)

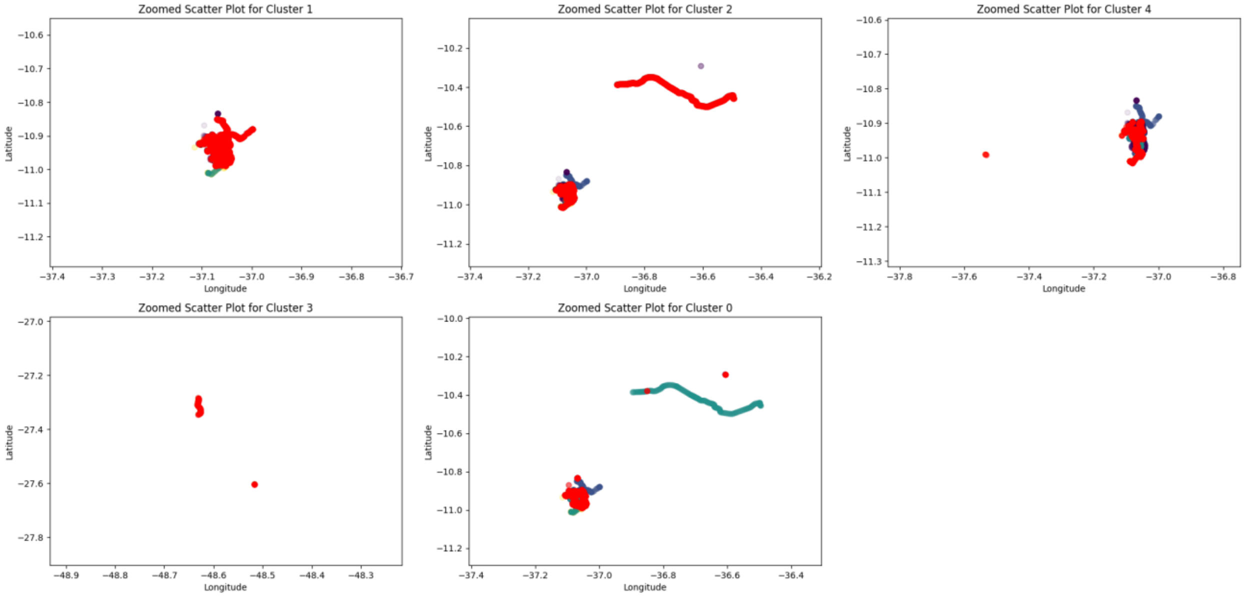

Figure 8 - Chameleon Agglomerative (Sub Plots)