Spatial Analysis of Taxi Log Data

1. Data Loading and Preparation

Process:

- Read multiple .txt files from a specified directory (E:\Summers\release\taxi_log_2008_by_id) into a Pandas DataFrame.
- Limited to the first 40 files for processing.
- Columns: ID (taxi identifier), Timestamp (datetime), Longitude, and Latitude.
- Converted ID to integer, Timestamp to datetime, and Longitude/Latitude to float for accurate analysis.

Output: A DataFrame with 63,593 rows and 4 columns, containing taxi GPS data from February 2–8, 2008.

2. Data Exploration and Visualization

I did it to understand the distribution and patterns in the taxi log data.

Process:

- Displayed the first 15,000 rows of the DataFrame to inspect the data.
- Created visualizations using Matplotlib and Seaborn:
  - **Time Series Plot**: Plotted longitude and latitude over time to observe movement patterns.
  - **Scatter Plot**: Showed longitude vs. latitude to visualize spatial distribution.
  - **Histograms**: Generated histograms for longitude and latitude to analyze their distributions.

Output: Visualizations revealed the geographic spread and density of taxi locations in the dataset.

3. Spatial Hashing and Banding

The goal was to organize data into spatial grids for efficient clustering.

Process:

- Implemented a hash_to_grid function to convert latitude and longitude into grid cells.
- Used a divide_into_bands function to group hashed coordinates into bands, creating buckets of nearby points.

Output: Points were grouped into spatial bands, enabling efficient identification of candidate pairs for clustering.

4. Candidate Pair Generation

To identify potential point pairs for clustering analysis.

Process:

- Generated candidate pairs within each band bucket, stored in a candidate_pairs dictionary.

Output: A set of point pairs likely to be close based on their spatial proximity within bands.

5. Distance Calculation and Clustering

The objective was cluster taxi locations based on geographic proximity.

Process:

- Used the haversine function to calculate great-circle distances between points.
- Applied **DBSCAN** clustering:
  - **Full Dataset**: Clustered all points in df[['Latitude', 'Longitude']].
  - Parameters: eps=0.5, min_samples=5, using Haversine distance.
  - **Candidate Pairs**: Clustered pairs identified from band buckets.
- Evaluated clustering quality using the **silhouette score**.
- Visualized clusters with a scatter plot, coloring points by DBSCAN cluster labels.

Output:

- **Execution Time**: 0.33 seconds for DBSCAN on the full dataset.
- **Silhouette Score**: 0.883, indicating good clustering quality (values close to 1 suggest well-separated clusters).
- The scatter plot showed distinct clusters of taxi locations, with colors representing different clusters.

6. Performance Evaluation

Assessed the efficiency and quality of the clustering process.

Process:

- Measured execution time for DBSCAN on the full dataset.
- Calculated the silhouette score to evaluate how well points were clustered.

Output:

- ➢ The clustering process was efficient (0.33 seconds for 63,593 points).
- ➢ A high silhouette score (0.883) confirmed that the clusters were well-defined and meaningful.

7. Conclusion

This analysis processed taxi log data to identify spatial clusters using DBSCAN. The steps included loading and cleaning data, visualizing spatial and temporal patterns, organizing data with spatial hashing, generating candidate pairs, and applying DBSCAN clustering. The results show efficient clustering with high quality results, as indicated by the silhouette score and visualized clusters. This approach is effective for analyzing large scale geographic data, such as taxi movements, to identify patterns and hotspots.