Indian Institute of Science

Bengaluru, Karnataka

CiSTUP

Internship Report

Test 1

Student Name

Nishkarsh Gupta

From,:

I.I.T, Hyderabad

Figure 1: Example image

Submission Date : 09/04/2023

Question 1 Part 1

Step1: Data Analysis

As a preliminary step,*Bicycle-sharing system dataset* consisting of 6,867 bicycle trips over one day was thoroughly analyzed. I leverage the tools from *panadas* library for the same. Following observation were made:

- Dataset was read as a Dataframe cossisting of 6867 rows and 7 columns. Each row denotes a unique datapoint/trip. Every datapoint/trip has following charateristics:

  `trip_id`: Unique trip identifier.

  `started_at`: Start time of the trip.

  `ended_at`: End time of the trip.

  `start_lat`: Latitude of the starting depot.

  `start_lng`: Longitude of the starting depot.

  `end_lat`: Latitude of the end depot.

  `end_lng`: Longitude of the end depot.

Program Logic

- The bicycle trip dataset is loaded into a *pandas DataFrame*.
- The *start* and *end* time columns in the DataFrame are converted to *datetime objects*.
- Trip duration in minutes is calculated by subtracting the start time from the end time and dividing the result by 60.
- Trips with a duration of 0 minutes are filtered out.
- The maximum and minimum trip durations are calculated and printed.
- The total number of trips corresponding to the minimum duration is calculated and printed.
- Circular trips are identified based on the start and end latitude and longitude being equal.
- The percentage of total circular trips is calculated and printed.
- The total runtime of the function is calculated by subtracting the start time from the end time. [*df.datetime.now()* was used to record the runtime]

Output

- The maximum and minimum trip durations are printed.
- The total number of trips corresponding to the minimum duration is printed.
- The percentage of total circular trips is printed.

- The total runtime of the function is printed.

```
In [30]: runfile('C:/Users/nishk/Downloads/untitled1.py', wdir='C:/Users/nishk/
Downloads')
Maximum duration of the trip (in minutes): 518.0000000000001
Minimum duration of the trip (in minutes): 1.0000000000000002
Total number of trips corresponding to the minimum duration: 89
Percentage of total circular trips: 2.4776425744025805 %
Total runtime for the function (in seconds): 0.137262
```

Question 1 Part 2

Analyzing the Data

- The data is loaded from the *"bike_data_new.csv"* file using Pandas library.
- The *"started_at"* column is converted to datetime format using the *"pd.to_datetime()"* function.
- Trips starting between 6:00 AM and 6:00 PM are filtered using datetime filtering methods.

```
In [33]: df
Out[33]:
      trip_id          started_at  ...    end_lat    end_lng
277       278 2023-01-02 07:00:00  ...  38.905737 -77.022270
278       279 2023-01-02 07:00:00  ...  38.881185 -77.001828
279       280 2023-01-02 07:00:00  ...  38.902760 -77.038630
280       281 2023-01-02 07:00:00  ...  38.887010 -77.095257
281       282 2023-01-02 07:00:00  ...  38.928743 -77.012457
...       ...                 ...  ...        ...        ...
4991     4992 2023-01-02 17:59:00  ...  38.908640 -77.022770
4992     4993 2023-01-02 17:59:00  ...  38.905578 -77.027313
4993     4994 2023-01-02 17:59:00  ...  38.900930 -77.018677
4994     4995 2023-01-02 17:59:00  ...  38.876697 -77.017898
4995     4996 2023-01-02 17:59:00  ...  38.847977 -77.075104

[4719 rows x 7 columns]
```

Program Logic

- The dataset is joined to itself using the *"pd.merge()"* function.
- The program filters feasible pairs of trips based on their start and end locations and start and end times.
- The total number of feasible pairs of trips is counted and printed.
- Feasible pairs of trips for a specific trip ID (4611) are filtered and a new DataFrame is created from the results.

```
In [49]: pairs
Out[49]:
       trip_id_x        started_at_x  ...  end_lat_y  end_lng_y
0            278 2023-01-02 07:00:00  ...  38.903040 -77.019027
1            278 2023-01-02 07:00:00  ...  38.905303 -77.050264
2            278 2023-01-02 07:00:00  ...  38.897283 -77.022191
3            278 2023-01-02 07:00:00  ...  38.898243 -77.026235
4            278 2023-01-02 07:00:00  ...  38.899032 -77.033354
...          ...                 ...  ...        ...        ...
85625       4877 2023-01-02 17:50:00  ...  38.813474 -77.053734
85626       4877 2023-01-02 17:50:00  ...  38.805317 -77.049883
85627       4933 2023-01-02 17:54:00  ...  38.810741 -77.044633
85628       4996 2023-01-02 17:59:00  ...  38.862478 -77.086599
85629       4996 2023-01-02 17:59:00  ...  38.847977 -77.075104

[85630 rows x 14 columns]
```

```
In [35]: feasible_pairs
Out[35]:
       trip_id_x        started_at_x  ...  end_lat_y  end_lng_y
0            278 2023-01-02 07:00:00  ...  38.903040 -77.019027
1            278 2023-01-02 07:00:00  ...  38.905303 -77.050264
2            278 2023-01-02 07:00:00  ...  38.897283 -77.022191
3            278 2023-01-02 07:00:00  ...  38.898243 -77.026235
4            278 2023-01-02 07:00:00  ...  38.899032 -77.033354
...          ...                 ...  ...        ...        ...
85576       4171 2023-01-02 17:00:00  ...  38.880761 -77.005741
85600       4611 2023-01-02 17:32:00  ...  38.885434 -77.173605
85601       4611 2023-01-02 17:32:00  ...  38.885434 -77.173605
85602       4611 2023-01-02 17:32:00  ...  38.887403 -77.176992
85603       4611 2023-01-02 17:32:00  ...  38.887403 -77.176992

[42346 rows x 14 columns]
```

Output

- The total number of feasible pairs of trips is printed.
- A new DataFrame containing feasible pairs of trips for trip ID 4611 is printed.
- The total runtime for the function is calculated and printed.

```
In [48]: feasible_pairs_4611.iloc[3,:]
Out[48]:
trip_id_x                      4611
started_at_x    2023-01-02 17:32:00
ended_at_x         01-02-2023 17:36
start_lat_x               38.885621
start_lng_x              -77.166917
end_lat_x                 38.883601
end_lng_x                -77.173438
trip_id_y                      4922
started_at_y    2023-01-02 17:54:00
ended_at_y         01-02-2023 17:57
start_lat_y               38.883601
start_lng_y              -77.173438
end_lat_y                 38.887403
end_lng_y                -77.176992
Name: 85603, dtype: object
```

```
In [41]: feasible_pairs_1733.iloc[0,:]
Out[41]:
trip_id_x                     1733
started_at_x   2023-01-02 10:02:00
ended_at_x        01-02-2023 10:13
start_lat_x              38.899972
start_lng_x             -76.998347
end_lat_x                38.897108
end_lng_x               -77.011616
trip_id_y                     1965
started_at_y   2023-01-02 10:57:00
ended_at_y        01-02-2023 11:09
start_lat_y              38.897108
start_lng_y             -77.011616
end_lat_y                38.878854
end_lng_y               -77.005727
Name: 30357, dtype: object
```

```
In [78]: runfile('C:/Users/nishk/Downloads/untitled2.py', wdir='C:/Users/nishk/Downloads')
Total number of feasible pairs of trips: 45540
       trip_id_x  trip_id_y  ...  end_lat_y  end_lng_y
92326       4611       4710  ...  38.885434 -77.173605
92327       4611       4792  ...  38.885434 -77.173605
92328       4611       4842  ...  38.887403 -77.176992
92329       4611       4922  ...  38.887403 -77.176992

[4 rows x 14 columns]
Total runtime for the function (in seconds): 5.079494
```

Question 1 Part 3

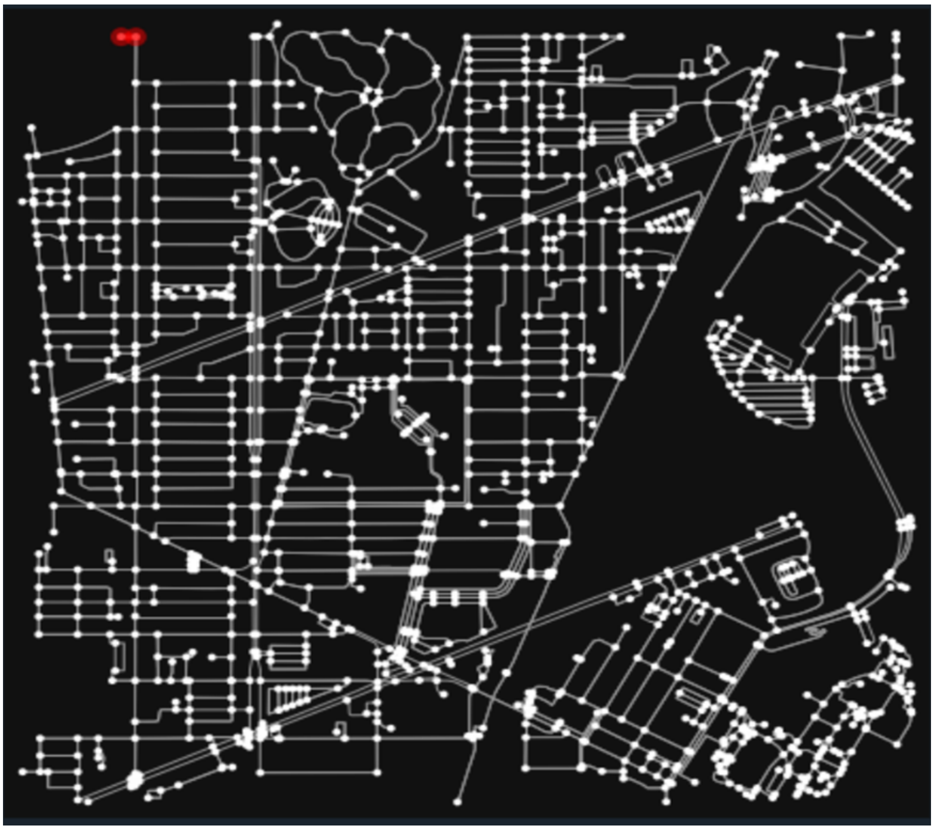
Analyzing the Data

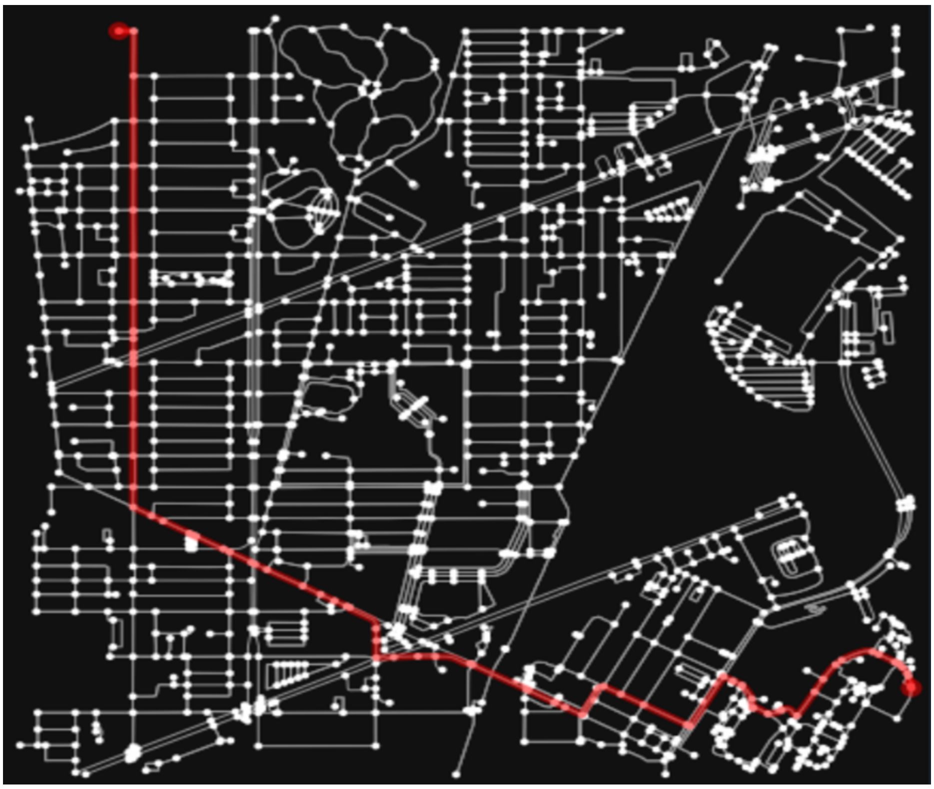
- The code reads a CSV file named *"bike_data_new.csv"* containing information about bicycle rides.
- The data has columns including start and end coordinates, start and end times, and trip IDs.
- The first 100 rows of the dataset are loaded and used for analysis.
- The unique depots are extracted from the start and end coordinates of the bike rides.
- The *OSMnx* package is used to create a street network graph for the first depot.

Program Logic

- The program reads in a CSV file and extracts the unique depots used by bike riders.
- For each depot, it finds the nearest node on the street network graph using the *OSMnx* package.
- The program then computes the shortest path between each pair of depots using the *bidirectional Dijkstra algorithm* from the *NetworkX* package.
- If there is no path between two depots, the distance is set to -1.

- The program then finds the pair of depots with the minimum and maximum distance between them and plots the shortest routes on the street network graph.





Output

- The program generates two plots showing the shortest routes between the pair of depots with the minimum and maximum distance between them.
- Total runtime for the function (in seconds) is also outputted.

```
In [63]: min_distance
Out[63]: 32.917

In [64]: max_distance
Out[64]: 3593.251
```

```
In [62]: runfile('C:/Users/nishk/Downloads/untitled3.py', wdir='C:/Users/nishk/Downloads')
Number of unique depots: 147
Total runtime for the function (in seconds): 15.788253
```

Question 2 Part 1

Data Analysis

- The dataset contains location data of users collected from GPS-enabled mobile devices over a period of time.
- It includes latitude, longitude, altitude, and time-stamp information for each location point.
- The data is organized by individual users, with each user having multiple trajectories.
- Trajectories correspond to outdoor movements, including daily routines such as commuting and non-routine activities like leisure and sports.
- The dataset can be used to analyze mobility patterns and develop location-based applications.

Program Logic

- The program defines two functions - *calculate_distance()* and *calculate_user_distance()* - to calculate the distance between two locations and the total distance traveled by a user, respectively.
- The program uses the multiprocessing module (*multiprocessing library in python*) to parallelize the calculation of distances for each trajectory, which improves the program's efficiency.
- The program loads the dataset using pandas and iterates over each user to calculate the total distance traveled by that user.
- The total distances are stored in a list and printed for each user.

Output

- The program outputs the total distance traveled by each user in the dataset.
- The output includes the individual ID and the total distance traveled for each user.
- The distances are measured in kilometers.
- *The code didn't run on my system since the dataset was too large and involved multiprocesses running at the same time.*

Question 2 Part 3

The use of GPS-tracking datasets can aid in the identification and analysis of commuting patterns of individuals. Commuting patterns are essential for urban planners and policymakers as they help in understanding the transportation needs of the population and improving the transportation infrastructure accordingly.

To solve this problem, I would use a combination of data analysis and visualization techniques. Firstly, I would preprocess the dataset by filtering out irrelevant data such as stationary points, data points outside the city limits, and data points with low accuracy.

Next, I would segment the data into individual trips using a clustering algorithm that groups the data points into different trajectories. Each trajectory would represent a single trip, such as from home to work or from work to leisure activity.

After that, I would extract features from the trajectories such as trip duration, distance, speed, and mode of transportation. This would require further analysis to differentiate between walking, cycling, driving, and public transport.

Lastly, I would utilize visualization tools to present the results of the analysis in a comprehensible way. For instance, heat maps could be created to indicate the most common routes taken by individuals during their commutes, while pie charts could show the percentage of people using different modes of transportation for their commutes.

Overall, this methodology would offer insights into the commuting patterns of individuals in a specific region, which would aid urban planners and policymakers in optimizing the transportation infrastructure accordingly. Additionally, it would help individuals make informed decisions about their transportation choices based on factors such as speed, cost, and convenience.