Problem 1:

Background

- We have a dataset containing information about the buses travelling in Bengaluru. We have obtained it from Bengaluru Metropolitan Transport Corporation (BMTC).
- The region of interest is an approximately 40km by 40km square area.

- The data was collected from around two thousand buses for one day, between 7:00am to 7:00pm.
- The buses follow different routes within the city.
- Each bus is identified with a unique ID. A bus carries a device which records the data: latitude, longitude, speed, and timestamp.

Task

Create a model to estimate the travel time, in minutes, between source-destination pairs using the provided dataset.

Dataset

We are providing the following three files in the dataset

1. BMTC.parquet.gzip: It contains the GPS traces of around two thousand buses.
2. Input.csv: It contains geographical coordinates of various sources-destination pairs.
3. GroundTruth.csv: It contains the ground truth travel times between the source-destination pairs provided in Input.csv. It is provided to help participants assess their solutions.

BMTC.parquet.gzip:
The file contains information in five columns, described as follows:

- *BusID*: The (unique) ID associated with the device present in a bus.
- *Latitude*: Latitude (geographical coordinate) of a bus, as recorded by the device.
- *Longitude*: Longitude (geographical coordinate) of the bus, as recorded by the device.
- *Speed*: Instantaneous speed of the bus in kmph.
- *Timestamp*: Timestamp in IST format. The format of datetime is yyyy-mm-dd HH:MM:SS.

For better understanding, following is a snapshot from the dataset:

BusID	Latitude	Longitude	Speed	Timestamp
0	150212121	13.06593	77.45269	20 2019-08-01 18:59:18

1	150212121	13.06627	77.45211	27	2019-08-01 18:59:28
2	150212121	13.06661	77.45152	24	2019-08-01 18:59:38
3	150212121	13.06697	77.45089	28	2019-08-01 18:59:48
4	150212121	13.06727	77.45035	26	2019-08-01 18:59:58
5	150218000	13.00571	77.68619	46	2019-08-01 07:22:33
6	150218000	13.00525	77.68542	35	2019-08-01 07:22:42
7	150218000	13.00504	77.68509	0	2019-08-01 07:22:51
8	150218000	13.00504	77.68509	0	2019-08-01 07:23:01
9	150218000	13.00498	77.68497	13	2019-08-01 07:23:11

Input.csv:
The file contains four columns, described as follows:

- *Source_Lat*: The latitude of a source.
- *Source_Long*: The longitude of a source.
- *Dest_Lat*: The latitude of a destination.
- *Dest_Long*: The longitude of a destination.

For better understanding, following is the format of a typical input file:

	Source_Lat	Source_Long	Dest_Lat	Dest_Long
0	13.067272	77.45035	13.00525	77.68542
1	13.005042	77.68509	13.06627	77.45211
2	13.065925	77.45269	13.00498	77.68497
3	13.005247	77.68542	13.06661	77.45152

The file contains one column *TT*, i.e. the actual travel time between a source-destination pair. The value in the *i*-th row corresponds to the travel time between *i*-th source-destination pair in Input.csv.
For better understanding, following is the format of a typical ground truth file:

	TT
0	1.99
1	6.21
2	7.34
3	5.20

You can use the ground truth from the dataset to check if your code is working well.

Output (Estimated Travel Time)

Your output will be the estimated travel time (ETT), in minutes, between a given source-destination pair. For each source-destination pair, you should fill this value in the ETT column of a pandas dataframe, as illustrated below:

	Source_Lat	Source_Long	Dest_Lat	Dest_Long
0	13.067272	77.45035	13.00525	77.68542
1	13.005042	77.68509	13.06627	77.45211
2	13.065925	77.45269	13.00498	77.68497
3	13.005247	77.68542	13.06661	77.45152

- The is is to build *models* to estimate the travel time between a source and destination, and code should work beyond the provided inputs. The model should try to minimize the mean absolute difference between the actual and predicted values

https://drive.google.com/file/d/12kIeXmePg9QHqnFMS-8Evl14BXHtHMGb/view?usp=drive_link

Problem 2:

The aim of the challenge is to predict the Air quality index (AQI); given the data and parameters for a particular area (e.g. Railway station) in Pune, create a model to predict Air quality index (AQI) for other areas (e.g. Bus stop, IT hubs).

- What are the key drivers that influence the AQI ?
- What rules can be used to predict the AQI ?
- Devising a predictive model to predict the AQI for other Areas
- Make it scalable to be used across the country
- Use the predictions to automate the actions to improve the air quality

Task

Create a model to estimate to predict the Air quality index (AQI) for the selected points

- Identify correlated observations
- Identify bias in data
- Identify correlations
- Take care of missing values
- Verify rmse on test-set
- Validate against submission score

https://drive.google.com/file/d/1K5wOKNLN_7g7ebIhI0IPAsXyg_rQipX0/view?usp=drive_link

Problem 3:

This data contains a record of user interactions with the restaurant recommendation system. This is an interactive system that recommends restaurants to the user based on factors such as cuisine, price, style, atmosphere, etc. or based on similarity to a restaurant in another city (e.g. find me a restaurant similar to the Patina in Los Angeles). The user can then provide feedback such as find a nicer or less expensive restaurant.

Task

Implement a recommender system based on Entree Chicago Recommendation Data. The Entree Chicago website makes its recommendations by finding restaurants in Chicago similar to restaurants the user knows and likes in all cities.

Data Characteristics

This data records interactions with Entrée Chicago restaurant recommendation system from September, 1996 to April, 1999. The data is organized into files roughly spanning a quarter year -- with Q3 1996 and Q2 1999 each only containing one month.

Each line in a session file represents a session of user interaction with the system. The (tab-separated) fields are as follows:

Date, IP, Entry point, Rated restaurant1, ..., Rated restaurantN, End point

Entry point

- Users can use a restaurant from any city as a entry point, but they always get recommendations for Chicago restaurants. The entry point therefore draws from a larger universe of restaurants than the rest of the data.
- Entry points have the form nnnX, where nnn is a numeric restaurant ID and X is a character A-H that encodes the city.

A = Atlanta

B = Boston

C = Chicago

D = Los Angeles

E = New Orleans

F = New York

G = San Francisco

H = Washington DC

Rated Restaurant

- These are all Chicago restaurants.
- These entries have the form nnnX, where nnn is a numeric restaurant ID and X is a character L-T that encodes the navigation operation.

L = browse (move from one restaurant in a list of recommendations to another)

M = cheaper (search for a restaurant like this one, but cheaper)

N = nicer (" " , but nicer)

O = closer (unused in the production version of the system)

P = more traditional (search for a restaurant like this, but serving more traditional cuisine)

Q = more creative (search for a restaurant serving more creative cuisine)

R = more lively (search for a restaurant with a livelier atmosphere)

S = quieter (search for a restaurant with a quieter atmosphere)

T = change cuisine (search for a restaurant like this, but serving a different kind of food) Note that with this tweak, we would ideally like to know what cuisine the user wanted to change to, but this information was not recorded.

End point

- Just the numeric id for the (Chicago) restaurant that the user saw last. In our experiments, we are assuming that this was a good suggestion, but it is also possible that user just gives up.

Some potentially useful data is missing. In many cases, we don't know the starting point because the user input a set of selection criteria (such as "inexpensive traditional Mexican") using a form submission, rather than starting from a known restaurant. These queries were not recorded. This is denoted by a 0 in the entry point field. Some sessions do not have a known end point. This is marked by -1 in the end point field.

https://drive.google.com/file/d/1mw_W3j1zd-6XN8L2jYL5KsJIZXEa1vXF/view?usp=drive_link