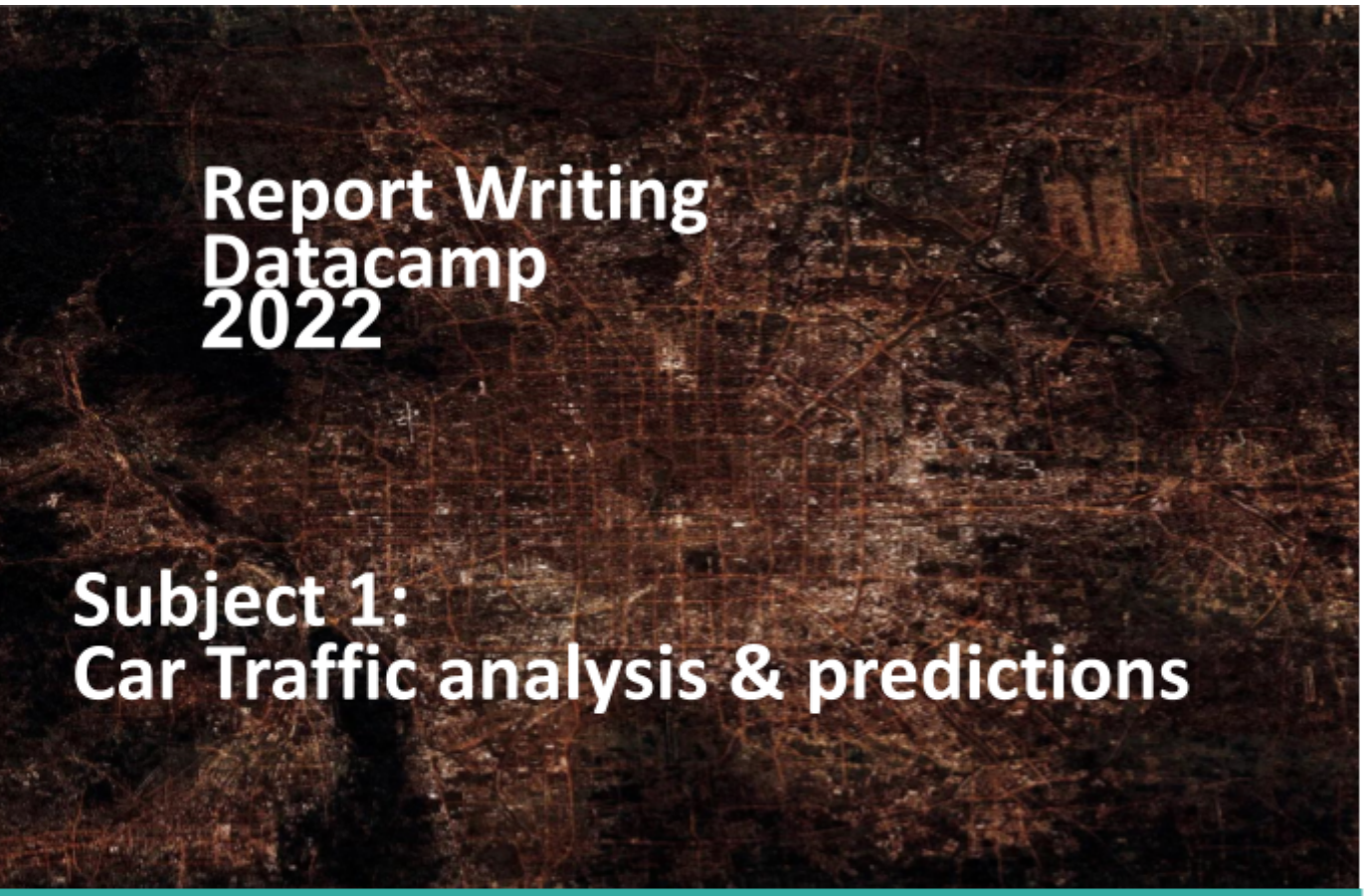


Report Writing Datacamp 2022

Subject 1: Car Traffic analysis & predictions

21 OCTOBRE

Member of group : -SHI Yunze
-Chicaté Glory
-Marc-Aurel Sèwèdo ALIOZA

1.Installation guides

The necessary packages are in '**requirements.txt'**. Just run '**pip install requirements.txt'** to install all the packages. If you are struggling to run our app you can access it with this link: https://aure-m-beijingtraffic-project-9sigu9.streamlitapp.com/

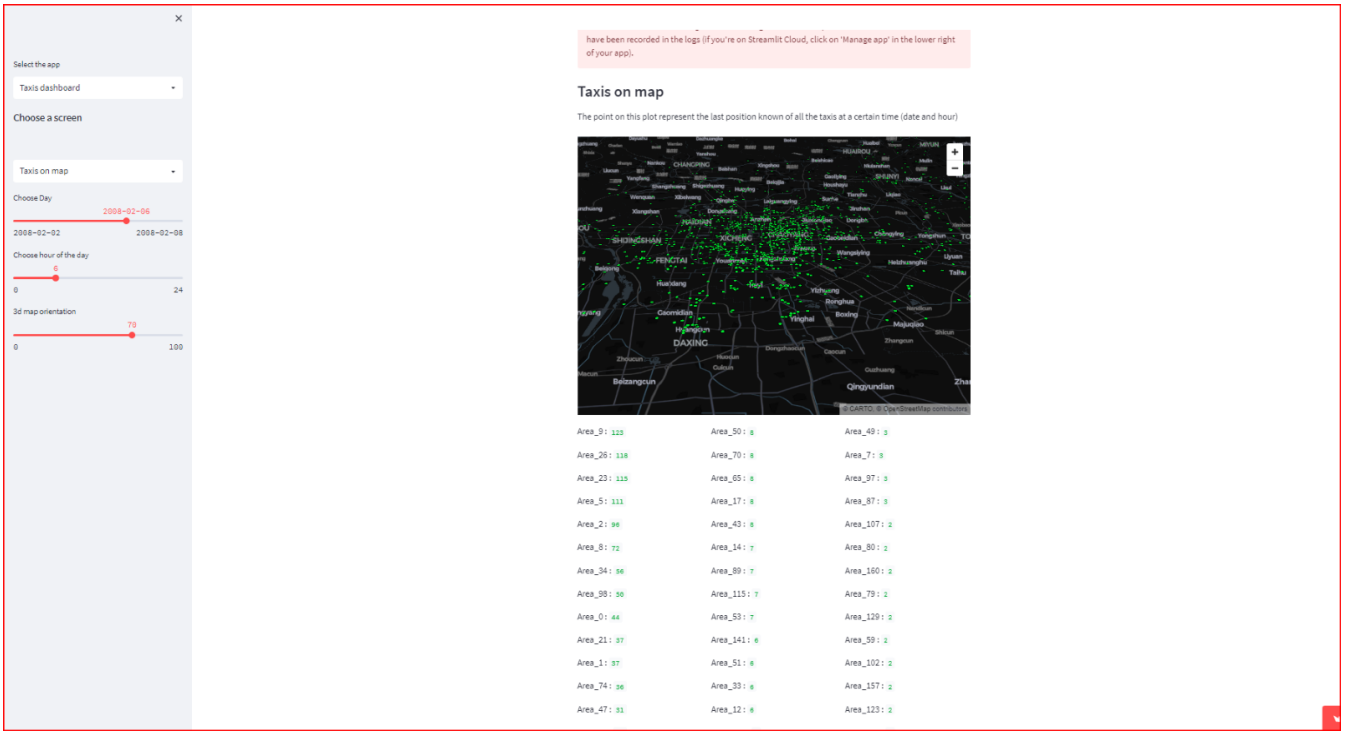
The project is also public on Github. GitHub Link: https://github.com/Aure-M/BeijingTraffic

2. User guide

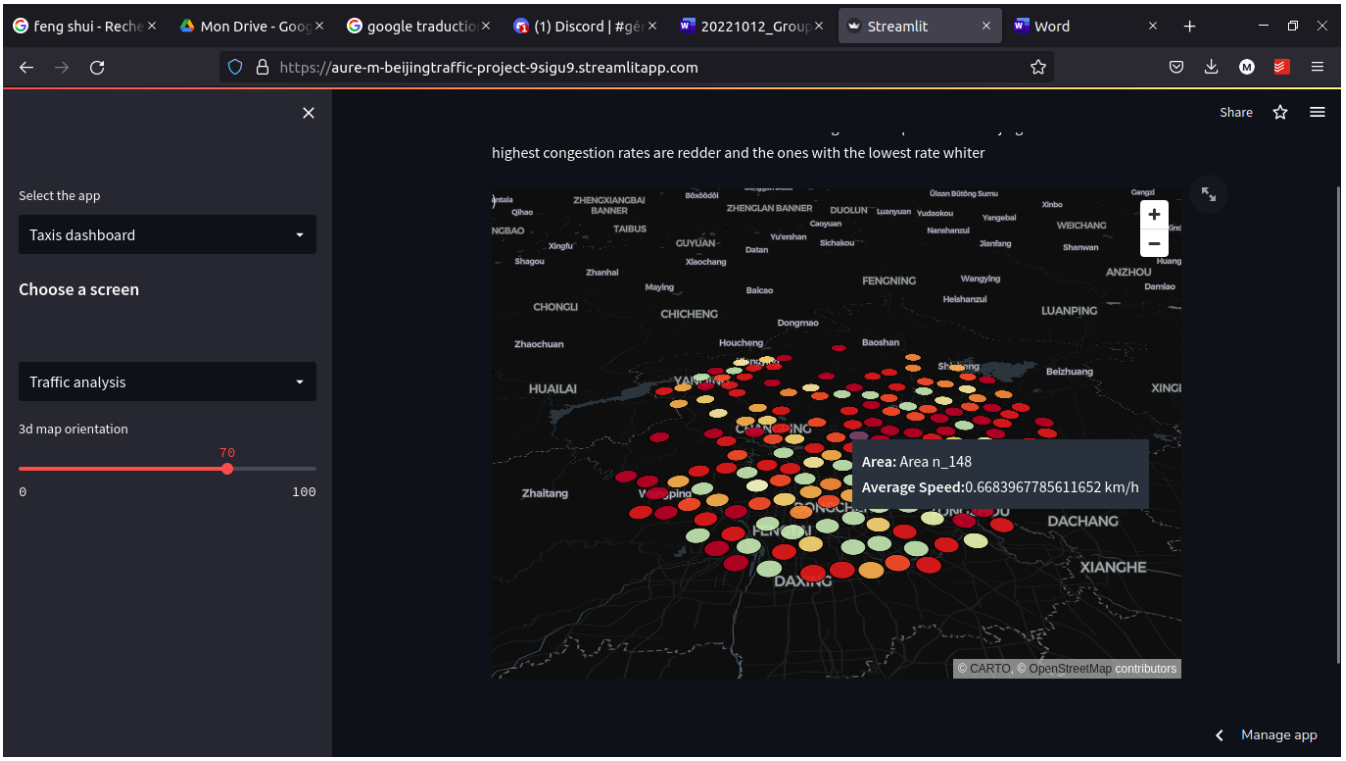
The application has 4 functionalities that give answers to problems of two different contexts.

- **1st context: A** company that owns cars hires taxi drivers in Beijing and wants a tool to better manage their taxis. We propose two functionalities to help them visualize the taxis distribution in Beijing:

    1. **The "Taxi on map" functionality will help them visualize where all the taxis are concentrated and then based on how they want to manage them they might send some taxis to cover certain areas**

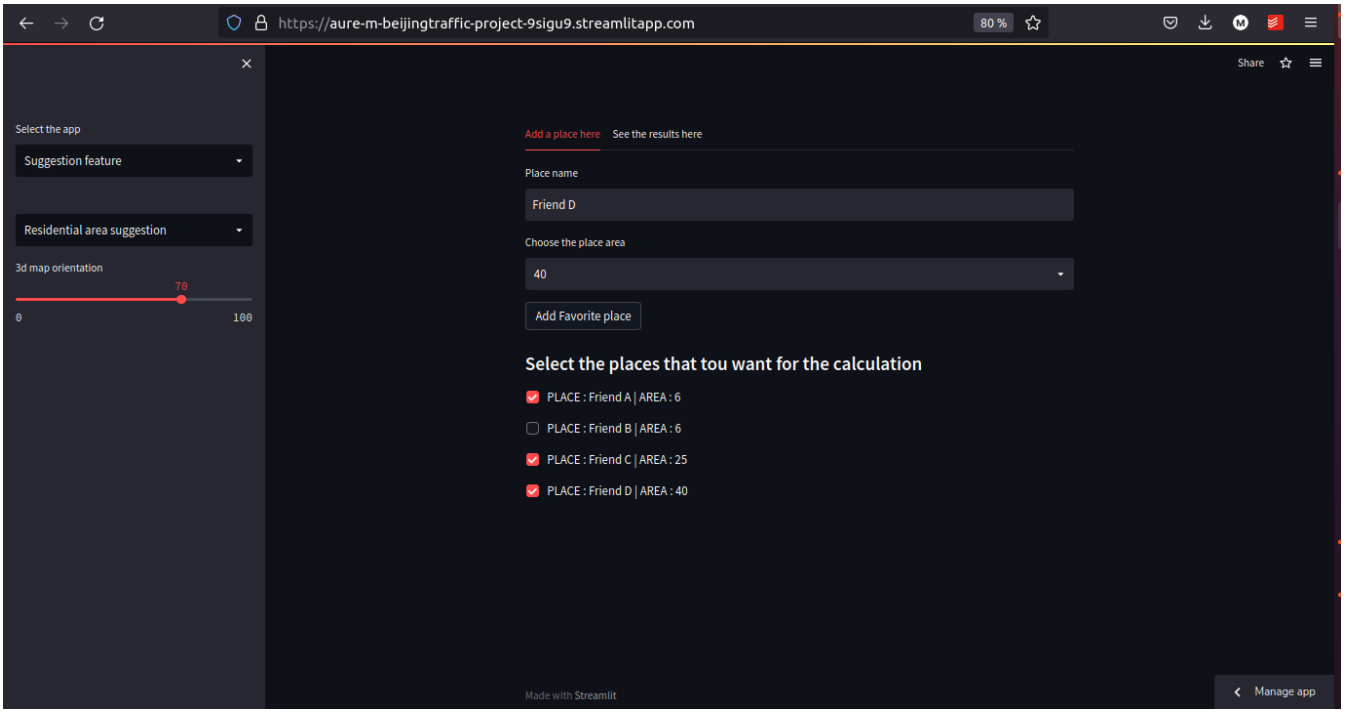


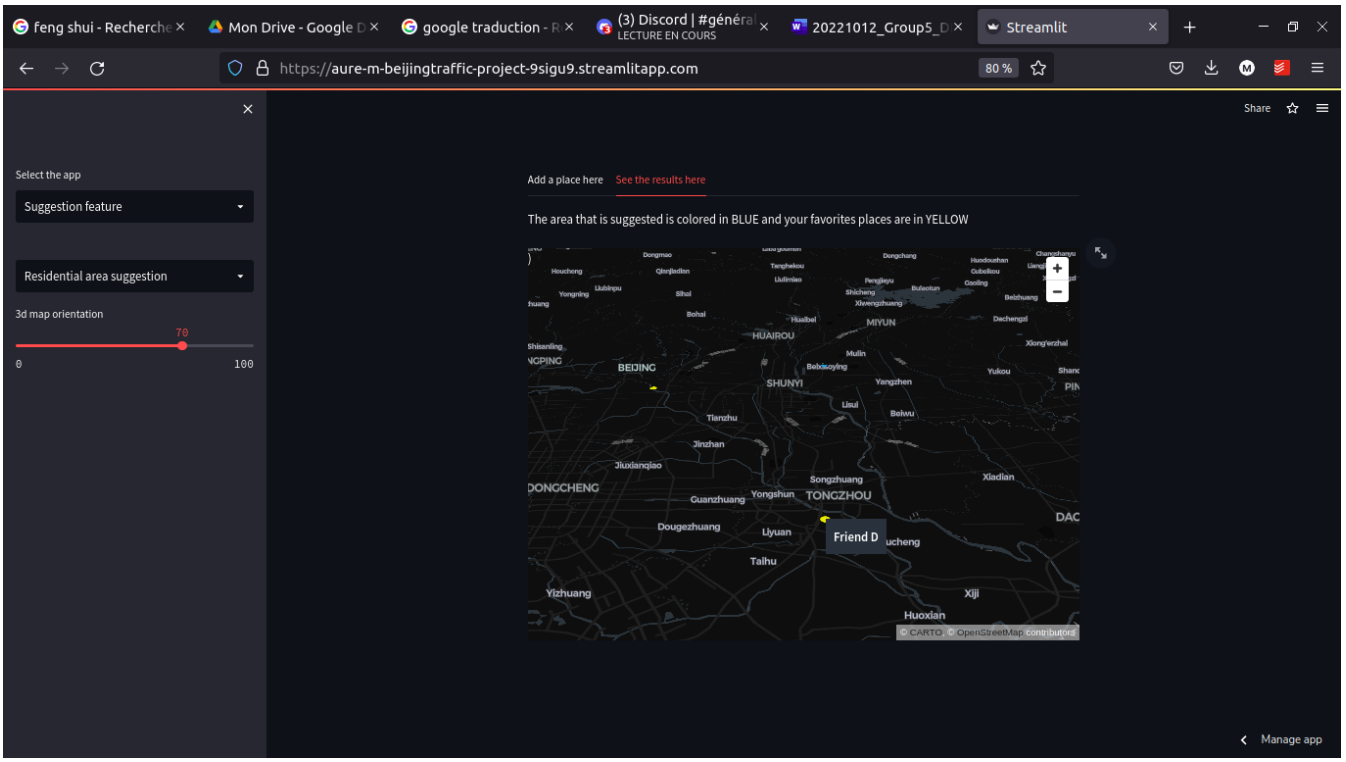
2. **The ”Traffic analysis“ functionality will help them visualize the areas with the highest conjunction rate**



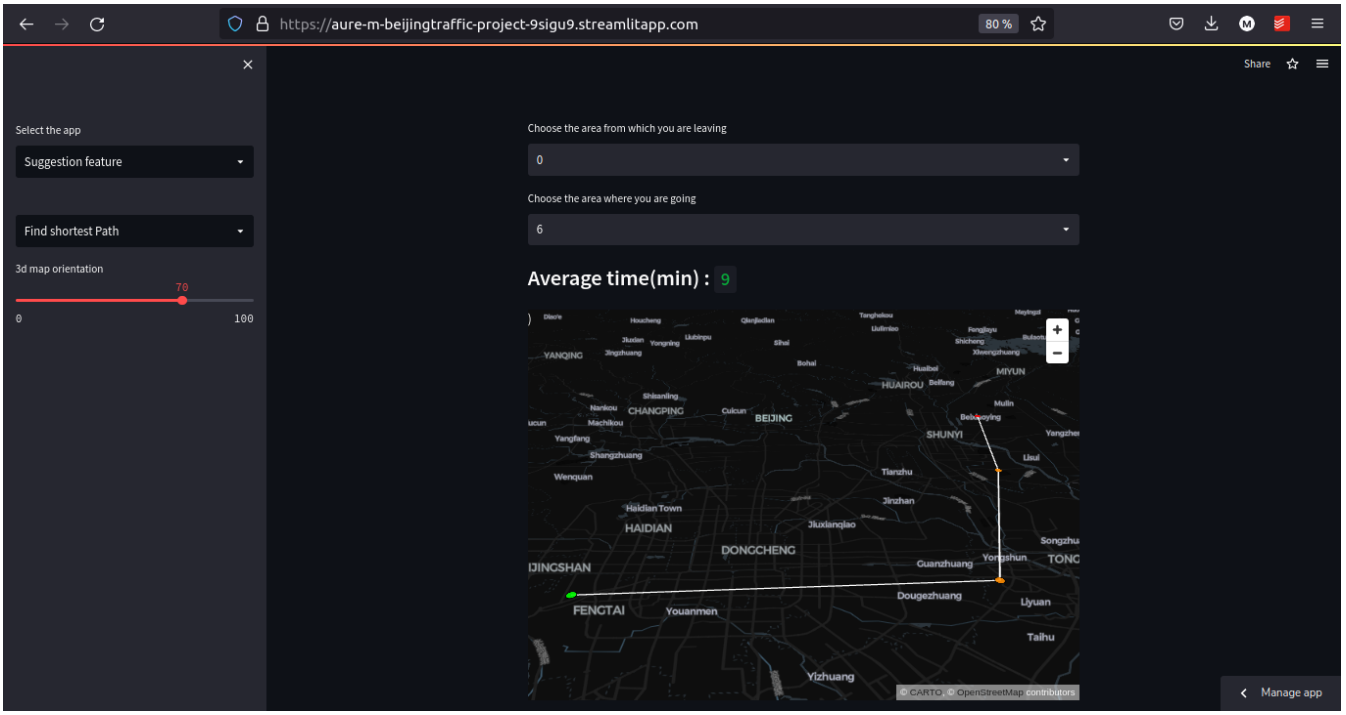
- **2nd context:** Foreigners want a tool to help them prepare for their arrival in Beijing.

1. **The ”Reidential areas suggestion“ functionality will so just them the best area to live in based on the places where they will frequently visit.**


https://aure-m-beijingtraffic-project-9sigu9.streamlitapp.com
80 %
Share
Select the app
Suggestion feature
Residential area suggestion
3d map orientation
70
0
100
Add a place here
See the results here
Place name
Friend D
Choose the place area
40
Add Favorite place
Select the places that tou want for the calculation
PLACE : Friend A | AREA : 6
PLACE : Friend B | AREA : 6
PLACE : Friend C | AREA : 25
PLACE : Friend D | AREA : 40
Made with Streamlit
Manage app


feng shui - Recherche
Mon Drive - Google D
google traduction - R
(3) Discord | #général
LECTURE EN COURS
20221012_Group5_D
Streamlit
https://aure-m-beijingtraffic-project-9sigu9.streamlitapp.com
80 %
Share
Select the app
Suggestion feature
Residential area suggestion
3d map orientation
70
0
100
Add a place here
See the results here
The area that is suggested is colored in BLUE and your favorites places are in YELLOW
BEIJING
SHUNYI
TONGZHOU
Friend D
Manage app

2. **The ”Shortest path“ functionality will help them find the shortest path to go from a point A to a point B in Beijing**



3.Tools we use:

The package we use:
-streamlit
-pandas
-numpy
-datetime
-pydeck
-math
-scipy
-cmath
-utils

4. Dataset preprocessing:

At the beginning of the project the dataset was divided in several files, we decided to compile all the files in a single txt file, to facilitate the use of pandas. Then we started to process the dataset by adding columns like to_zone,from_zone,isinBeinjing in order to use them later in the application.

Preprocessing steps:

1. Merging all the taxis data in one file
2. Filter of the dataframe (Some coordinates of taxis were not in the geographical space of Beijing so we decided to drop those data)
3. Calculating the time differencial and the distance between two consecutive positions for a certain taxi
4. Associating the positions with the closest area

datetime	taxiId	longitude	latitude	dayMoment	hours	days	res_T	res_D	speed	isinBeijing	to_zone	from_zone
2008-02-02 13:30:44	3015	116.41036	39.89171	afternoon	13	Sat	NaN	NaN	NaN	True	9	NaN
2008-02-02 13:30:45	3015	116.41028	39.89170	afternoon	13	Sat	0.016667	0.006915	0.414912	True	9	9.0
2008-02-02 13:30:46	3015	116.41022	39.89170	afternoon	13	Sat	0.016667	0.005119	0.307135	True	9	9.0
2008-02-02 13:30:47	903	116.20268	39.99162	afternoon	13	Sat	NaN	NaN	NaN	True	78	NaN
2008-02-02 13:30:48	366	116.45353	39.90732	afternoon	13	Sat	NaN	NaN	NaN	True	23	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
2008-02-08 17:39:15	181	116.46228	39.88309	afternoon	17	Fri	2.700000	0.229454	0.084983	True	23	23.0
2008-02-08 17:39:16	207	116.35942	39.97558	afternoon	17	Fri	0.083333	0.007968	0.095616	True	98	98.0
2008-02-08 17:39:17	71	116.30908	40.00386	afternoon	17	Fri	0.933333	0.472593	0.506349	True	11	11.0
2008-02-08 17:39:18	274	116.46982	39.87197	afternoon	17	Fri	5.000000	0.001112	0.000222	True	23	23.0
2008-02-08 17:39:19	1513	116.42762	39.93923	afternoon	17	Fri	3.116667	2.755628	0.884159	True	26	5.0

6.List of function:

-Display the different routes and the average travel time according to the position and the date.
-Calculate the average speed
-Display the best path for the travel
-Display the congestion of taxi by area in function of time of day and hour
-Suggest area for residential

7. Our algorithms:

Function to filter the dataset and display the taxi on the map according to the parameters.

```
@st.cache(suppress_st_warning=True)
def getTaxiLastPos(taxis,date,hour):
    filt = taxis[isDateEqualTo(taxis.index.map(getDate), date)]
    filt = taxis[(taxis["hours"]<=hour)&(taxis["hours"]>=hour-1)][["latitude","longitude","taxiId","to_zone"]]
    res = []
    taxis = filt["taxiId"].value_counts().index

    for t in taxis:
        res.append(filt[filt["taxiId"] == t].iloc[-1])
    return pd.DataFrame(res)
```

Function , we use for calculate the distance between two zone with latitude and longitude as argument.

```
def distance(lat1, lat2, lon1, lon2):

        # The math module contains a function named
        # radians which converts from degrees to radians.
        lon1 = radians(lon1)
        lon2 = radians(lon2)
        lat1 = radians(lat1)
        lat2 = radians(lat2)

        # Haversine formula
        dlon = lon2 - lon1
        dlat = lat2 - lat1
        a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)

        c = 2 * asin(sqrt(a))

        # Radius of earth in kilometers. Use 3956 for miles
        r = 6371

        # calculate the result
        return(c * r)
```

The function to find the shortest path in using algorithm of Dijkstra and a adjacent matrix.

```
def findShortestPath(a,b,adjacencyMatrix):
    dist_matrix,predecessors = dijkstra(csgraph = adjacencyMatrix,indices = a, directed = True, return_predecessors = True)
    path = []
    time = 0
    tmp = b
    while tmp != a:
        path.append(tmp)
        time+=dist_matrix[tmp]
        tmp = int(predecessors[tmp])
        if tmp == -9999:
            return None, None
    path.append(a)
    path.reverse()
    return path, time
```

8. Difficulties and approaches to solutions

- When calculating the average time, you should pay attention to the data classification, which should be carried out on the same day, and you cannot select information that is not on the same date. At the same time, it is necessary to group the taxis. When the taxis reach a fixed range, the average time between the two places can be calculated.
- Our adjacency matrix was based on an oriented graph read. That is why the “Shortest Path” functionality suggests sometimes a long path where the real shortest path is obviously shorter. We could have solved this problem by creating the adjacency matrix on a non oriented graph read of the data.
- We were missing important data and this sometimes distorted the calculations. In some cases the time difference between two consecutive runs of the same taxi was far too great. We thought that the anomaly observed was probably due to the fact that the taxi concerned was parked at the end of its working day. This could explain why, for example, between two areas that are very close, the time taken to travel is enormous.

Conclusion:

We tried to respect what was requested for the realization of the project despite all of the difficulties related to the data or even the app creation with streamlit. After two weeks of work on the project we have better ideas to solve the problems encountered. With more time we can surely do a better job on this project and since the code is on Github, it will be easy to maintain and evolve the application. We could imagine that the application is connected in real time to the position of the cab or that the application is a function in the future to display the cost of the trip based on the distance and the current price of gas.