Locating The Ideal Restaurant Venue

Thomas Trankle

June 07, 2020

1.Introduction

1.1. Background

The coronavirus has changed the way restaurants will have to conduct business in the foreseeable future. A lot of restaurants are debating whether or not to stay under the working operations/methodologies induced by COVID-19 or to return to things as they once were. I, and an investor, have been interested in opening a restaurant in Washington, D.C. based on one we ate at in Greenville, SC. The reason I chose Washington, D.C. is because I was limited in the data sets available and/or offered by Uber Movement/ API's. While we think it has the prospects of being an ideal fit for a suburban area in the greater Philadelphia area, we are unsure where the best location for it would be, and if a location is even necessary given the current times. Therefore, I would like to use Foursquare location data to statistically show the best place to open a restaurant. However, an important constraint is delivery time. As companies like DoorDash and Uber, more specifically UberEats, grow it is imperative to find a location where individuals will have minimal waiting time if they order food through these services. So, we will need to find a location that has the potential of receiving a lot of traffic through the restaurant's physical location and that can reach the most amount of customers with a minimal delivery time. Due to the delivery time constraint, we will have to create a limit, for example average delivery time of 20 minutes or less, that customers can expect to wait. From this we will be able to see what our potential reach is. Additionally, we can explore changing the delivery time and see what resulting information and analysis results.

1.2.Problem

Data to solve this problem would ideally consist of housing and location travel time relationships.This relationship could calculate the overall potential of a specific location for a restaurant's projected earnings . However, a specific location is difficult to define. We can not guarantee that a pon point location will be available (potentially already occupied) so we will restrict this to zip codes since the area of a zip code generally changes proportionally with the metropolitan area it belongs to. For example, the Washington metropolitan area covers approximately 4,000 square miles [1] and contains 53 zip codes [2].

1.3.Interest

Food trends will always be a potential market place. However, due to the global pandemic people are more reluctant than ever to go on expensive trips where they can experience other cultures and their respective cuisines. Entrepreneurs could however bring the cuisines to customers by offering them the opportunity to expand their palette, with a delicious meal, while also providing them the option of enjoying it in the comfort of their own home.

2.Data Acquisition and Cleaning

2.1.Data Sources

The two datasets required for this analysis are current restaurant information and a data set with travel time locations in the area. The first dataset can be acquired by making use of the Foursquare API to explore venue locations that have similar target groups to our restaurant and then using the venue feature once more to acquire the ratings of these restaurants. Once we have a list of restaurants, we can see if there is an area containing high reviews on average. We would like the restaurant to be in an area of this nature so that it has a high standard associated with it. The travel time to different location data will be retrieved by using multiple datasets from Uber Movement. The source for all the datasets can be found here but specific ones used after filtering are located here.

2.2.Data Cleaning

Data downloaded or scraped from multiple sources were processed one by one into a single python dictionary. Initially, the Foursquare API was able to list many restaurants in the same category, that is primarily serving breakfast and coffee, but once we tried to retrieve each of the restaurants rating it was apparent that there are a lot of missing values. The Foursquare API limits requests to no more than 100, of which 50 of these 100 restaurants had ratings. Nevertheless, we decided to proceed with the data available because other API’s have costs associated with them and using the Foursquare API is a required part of this capstone project. Like many returns from API, it has the datatype *object*. Since we needed to manipulate this ratings later on, we changed its datatype to *float* to avoid errors later on.

There are some problems with the data sets. Firstly, the coordinate locations in the JSON file were not as easy to access like the ones in previous assignments. Additionally, the coordinate section listed multiple latitudes and longitudes because it refers back to the travel time data. This means that when recording the data, there are multiple pings on the GPS, so there are multiple locations recorded along each trip. Thus, this dataset was stored with a geometry of a multipolygon as opposed to just a regular polygon. Since the other data sets already contained the travel time, I was only interested in the final location of each trip. Therefore, after a lot of experimentation I was able to isolate the final coordinates for each location and store them in a dataframe.

Secondly, for some reason the latitude and longitude coordinates that were extracted from the data set were in the incorrect order. Meaning it first displayed the longitude and then the latitude. This was an easy fix but took a while to realise and I thought I was having an issue with Folium and not the coordinates.

2.3. Feature Selection

After the decision was made to use zip codes as general areas for an ideal location, due to possibility and availability, I had to download a dataset from each different starting location (origin) separately. This was an unfortunate setback because while this feature can be manipulated on the front end of the Uber Movement website, if you chose to download the entire data set, which contains all the original locations, it doesn't organise it by zip code. Thus, we are simply left with origin ID numbers that are without a reference system. While this was a meanuel task, it provided simple datasets which contained the features we needed and had the same list of endpoint locations.

After cleaning the data and putting most of it in respective data frames, there were 520 sample locations and most of the features were not needed. Because the data we downloaded was over the period of 3 months, from January 1st ,2020 to March 31st ,2020, the most important metric was the average or mean travel time to these locations. Luckily this feature was already calculated for in seconds so we just had to convert it to minutes.

3. Methodology

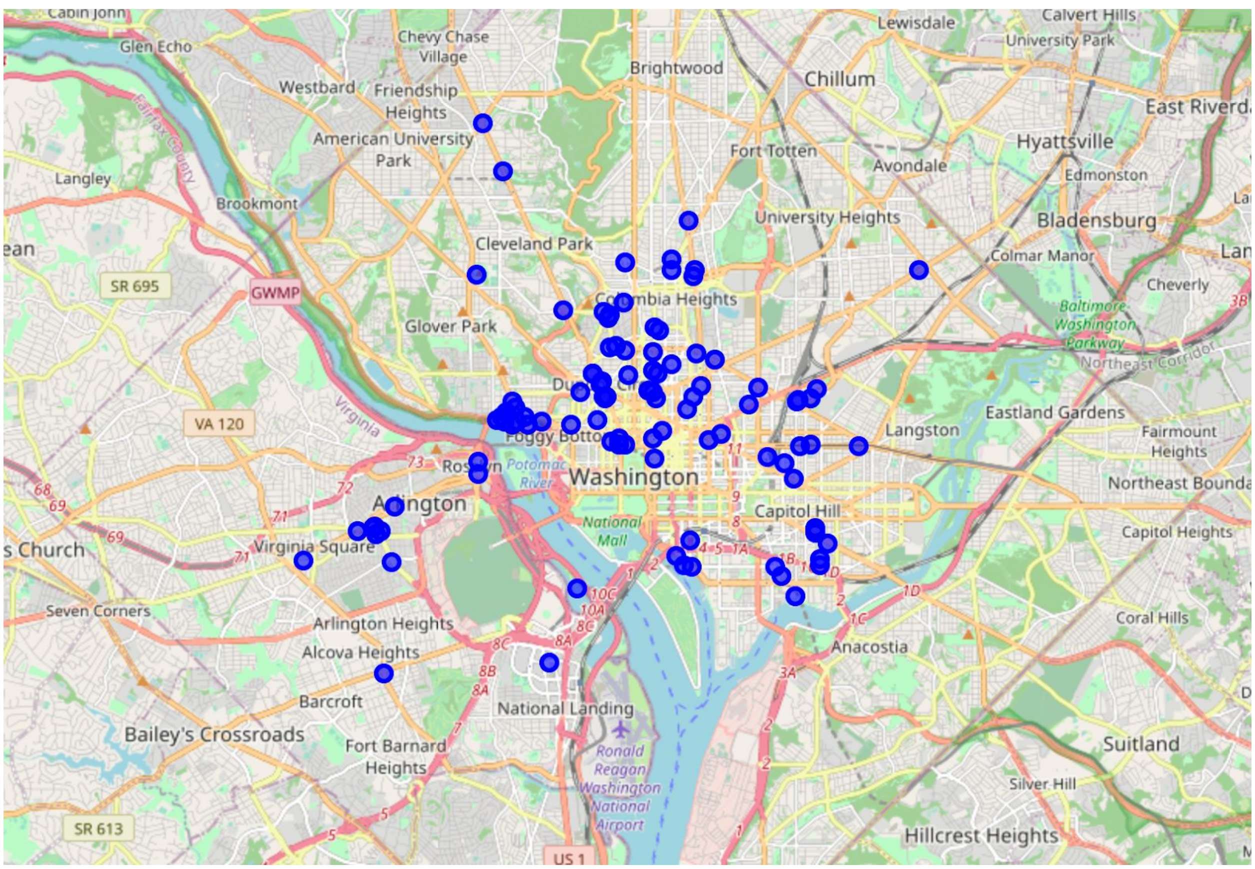
3.1. Calculation of target variable - End Goal

The target variable is the total number of random locations generated by Uber Movement data sources that are within an average of 20 minutes from the original location. Since all origin locations are areas with high ranking restaurants in, that condition is already satisfied.

3.2.Relationship between restaurants

Using the Foursquare API venues and explore features and reading the API documentation, we were able to add query and categoryID to the search function. By entering the categoryID for *food* and specifying the query as *'coffee'*, this method allowed us to find restaurants in the area that most likely serve breakfast but are not solely limited to this genre of food. Once we added all 100 restaurants provided by Foursquare, the categories linked to these restaurants indicated that this was the correct list of restaurants that served a similar target group but have diverse products.

```
url2 = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}&categoryId={}'.format(CLIENT_ID,
CLIENT_SECRET, latitude, longitude, VERSION, 'coffee', radius, LIMIT, '4d4b7105d754a06374d81259')
```



3.3.Acquiring the ratings for each restaurant

Once more the Foursquare API was utilized to look up specific venues by using their unique *client_id*. By iterating through the dataframe of restaurants found above and extracting each *client_id* we were able to retrieve the rating of each restaurant. However, some restaurants did not have ratings so we simply replaced what would be a numeric value with a string stating that this restaurant did not possess a rating.

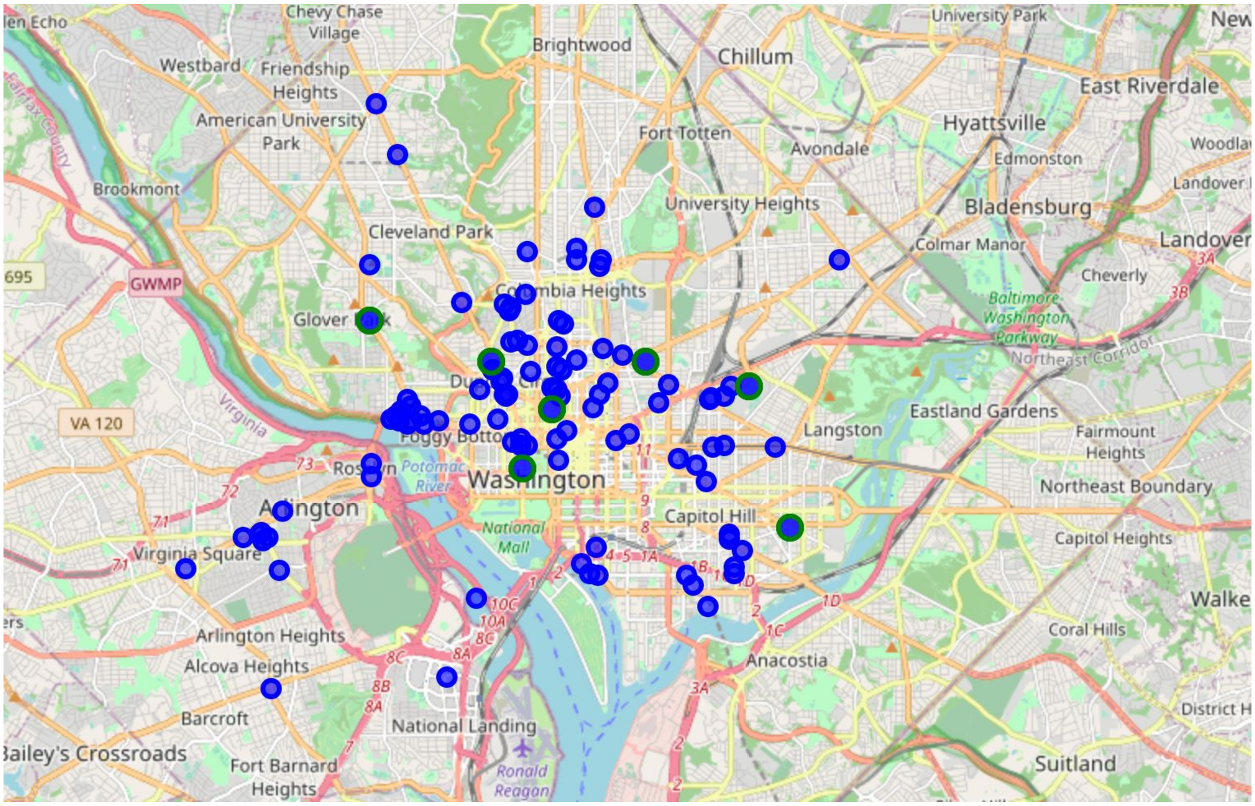
```
#Now we will create an empty list and iterate through the adding each locations rating
ratings_list = []
for i in range(len(df_filtered)):
    venue_id = df_filtered.id[i] # ID restraunt
    url = 'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secret={}&v={}'.format(venue_id, CLIENT_ID, CLIENT_SECRET, VERSION)

    result = requests.get(url).json()
    try:
        ratings_list.append(result['response']['venue']['rating'])
    except:
        ratings_list.append('This venue has not been rated yet.')
```

3.4.Sorting top rated restaurants into zip codes

Once we dropped all the restaurants without rating we sorted the data frame by grouping all the restaurants by zip codes and calculating the mean for each zip code respectively.

```
postalCode
20001    9.000000
20002    8.860000
20003    8.960000
20005    8.600000
20006    8.566667
20007    8.728571
20009    8.800000
20010    8.900000
20036    8.500000
20037    8.200000
22202    9.000000
22209    8.700000
```

We chose to take the top 7 zip codes, which was a completely random number. We now have 7 origin locations and must figure out which of these 7 highly rated zip codes can reach the most locations in under 20 minutes.



3.5. Refining JSON file to show all reachable locations

This task took a while. Essentially we wanted to access the last coordinate in JSON file that displayed the following:


```
043563,38.983988],[-77.043622,38.983992],[-77.045758,38.984151],[-77.046137,38.984454],[-77.047587,38.98536],[-77.048987,38.985632],[-77.050041,38.986067],[-77.
05177,38.986949],[-77.051623,38.987145]]]]},"properties":{"MOVEMENT_ID":"2","DISPLAY_NAME":"1400 Juniper Street Northwest, Northwest Washington, Washington"}},
{"type":"Feature","geometry":{"type":"MultiPolygon","coordinates":[[[[-77.119751,38.93435],[-77.118861,38.935356],[-77.117398,38.936398],[-77.117207,38.936527],
[-77.117146,38.936588],[-77.117059,38.936658],[-77.116984,38.936725],[-77.11691,38.936769],[-77.116844,38.93682],[-77.116826,38.936836],[-77.116808,38.936854],[-77.
116793,38.936864],[-77.116767,38.93689],[-77.116749,38.936918],[-77.116144,38.937463],[-77.115189,38.938164],[-77.114848,38.938435],[-77.114132,38.939003],[-77.
```


The final location was deeply embedded but was finally ascertained using the following line.

```
dc_data['features'][0][0]['geometry']['coordinates'][0][0][-1]

[-77.048009, 38.841266]
```

Which is quite different to the JSON files presented throughout the IBM certification course. However, once this was achieved, we could refine the list of locations by:

1.) Adding a column which showed the mean travel time in minutes - simply achieved by dividing the mean travel time in seconds (given) by 60.

```
df_travel['Mean Travel Time (Minutes)'] = df_travel['Mean Travel Time (Seconds)']/60
df_travel
```

	Origin Movement ID	Origin Display Name	final_loc_id	Destination Display Name	Date Range	Mean Travel Time (Seconds)	Range - Lower Bound Travel Time (Seconds)	Range - Upper Bound Travel Time (Seconds)	lat	lng	Mean Travel Time (Minutes)
0	168	Washington, DC	1	5400 Arnold Avenue Southwest, Southwest Washin...	1/1/2020 - 3/31/2020, Every day, Daily Average	1023	711	1471	38.841266	-77.048009	17.050000
1	168	Washington, DC	2	1400 Juniper Street Northwest, Northwest Washi...	1/1/2020 - 3/31/2020, Every day, Daily Average	1613	1297	2004	38.987145	-77.051623	26.883333
2	168	Washington, DC	3	4800 Canal Road Northwest, Northwest Washingto...	1/1/2020 - 3/31/2020, Every day, Daily Average	855	634	1151	38.934350	-77.119751	14.250000

2.) Eliminating all locations that were greater than 20 minutes away

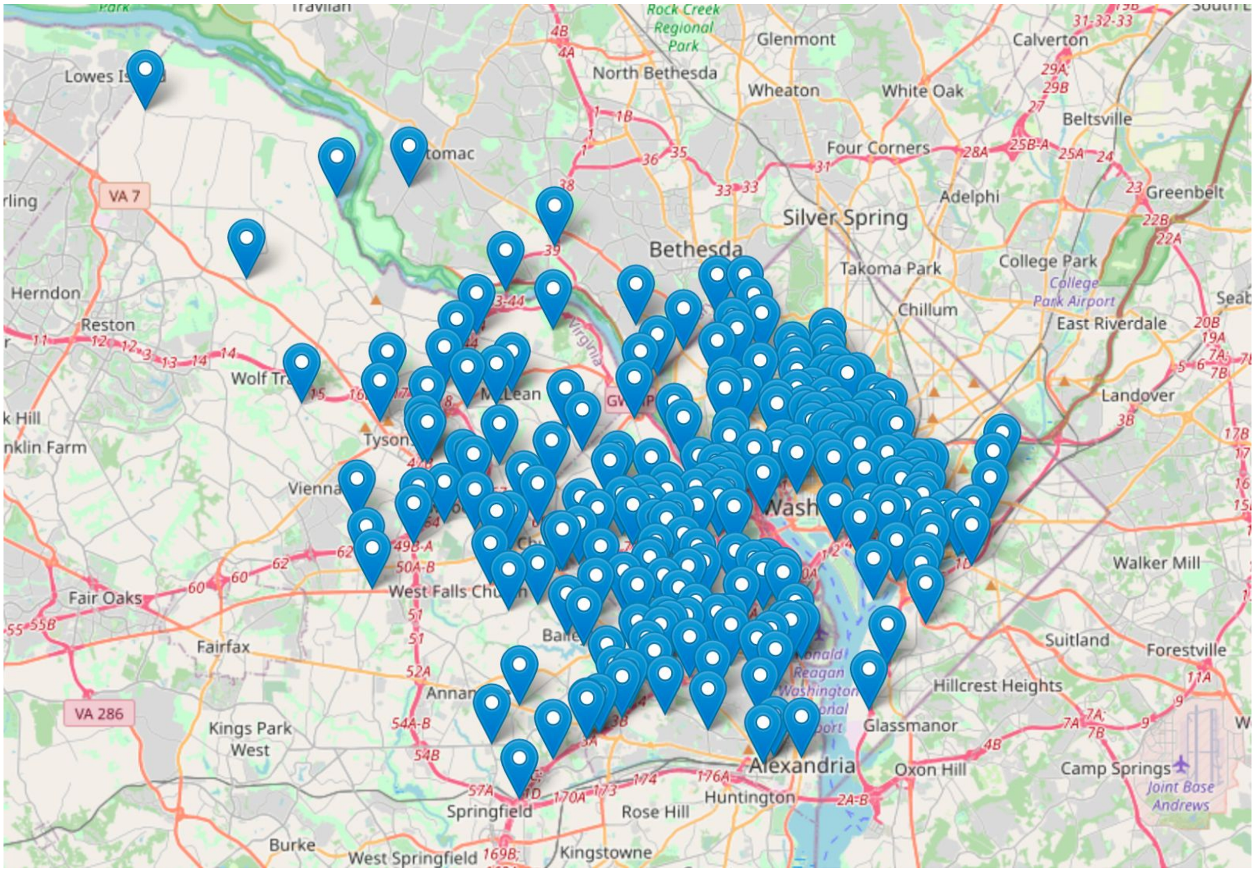
```
df_travel.drop(df_travel[df_travel['Mean Travel Time (Minutes)'].values > 20.0].index, inplace=True)
```

```
df_travel
```

	Origin Movement ID	Origin Display Name	final_loc_id	Destination Display Name	Date Range	Mean Travel Time (Seconds)	Range - Lower Bound Travel Time (Seconds)	Range - Upper Bound Travel Time (Seconds)	lat	lng	Mean Travel Time (Minutes)
0	168	Washington, DC	1	5400 Arnold Avenue Southwest, Southwest Washin...	1/1/2020 - 3/31/2020, Every day, Daily Average	1023	711	1471	38.841266	-77.048009	17.050000
2	168	Washington, DC	3	4800 Canal Road Northwest, Northwest Washingto...	1/1/2020 - 3/31/2020, Every day, Daily Average	855	634	1151	38.934350	-77.119751	14.250000
4	168	Washington, DC	5	4500 Q Place Northwest, Northwest Washington, ...	1/1/2020 - 3/31/2020, Every day, Daily Average	575	413	799	38.911209	-77.100859	9.583333
5	168	Washington, DC	6	Ridge Road Northwest, Northwest Washington, Wa...	1/1/2020 - 3/31/2020, Every day, Daily Average	1066	838	1355	38.954336	-77.069425	17.766667
6	168	Washington,	7	4500 Ohio Drive Southwest, Southwest	1/1/2020 - 3/31/2020, Every day, Daily	237	95	589	38.891852	-77.064696	3.950000

4.Results

The result of the above method leaves one with a dataframe containing random locations provided that are less than 20 minutes away from the defined original location, which is our list of zip codes.



Therefore, the final target variable is ascertained by simply calculating the length of the dataframe. Redoing the method for the other top zip codes results in a score for each of them respectively.

```
print(score)
{'20005': 200, '20002': 171, '20003': 235, '20007': 132, '20001': 202, '20009': 148, '20006': 231}

max(score, key=score.get)

'20003'
```

20003 is the best zip code to open a new restraunt

Hence, we can see that the zip code 20003 in Washington, D.C. is the best area to open a restaurant in the Washington metropolitan area with our criterium.

5.Discussions

The requirement to use Foursquare API made is this reason I chose this project. Ideally, if I was to use the results produced by this project, I would apply to many locations. There are many restaurants which would only survive in certain locations and that's how the idea came to be. Unfortunately many cities do not have the open source data needed for this kind of analysis. If you go to the Uber Movement website you will notice that not even every major city in the US is listed as an option. This makes it difficult to have a free and readily available dataset that the model can be applied to

Users could pay for API services and build their own travel time matrix but this is not an option for all which is unfortunate.

Also, I've mentioned throughout why we are restricted to zip codes. However if one were to pay for API services and make their own time-distance matrix, it would be possible to define an exact location. But, as before, the occupants might not be willing to sell or perhaps they could even be a future competitor.

6.Conclusions

Restaurant industries have really been hurt by the global pandemic which started rapidly spreading in March 2020. Now upcoming locations in the industry can operate as a normal restaurant by making sure they meet serving restrictions implemented by governing bodies while also maximising their profits by being able to deliver to their customers using whatever service they deem appropriate.

7.Future Direction

This project can also be used for franchise restaurants.

Additionally, the process could be reversed to see how many clients would potentially come to a store such as a pharmacy.

Data to starting point

8. References

[1] Washington Metropolitan

[2] Washington D.C. Zip Codes