IBM Data Science Capstone Project
Leveraging FOURSQUARE API

ON

Demonstration of Fire and Rescue Station need nearby Crash Location in Washington DC Region

Prepared By:
Roshan Dhakal

Table of Contents:

INTRODUCTION

The cities in US have high number of working people and hence, the traffic flow is maximum almost every hours. People, sometimes unfortunately, while going to their places may be involved in the road accident that might be fatal. This capstone project deals with the data of crash in Washington DC, the capital of USA, and get the nearby location of the crash using Foursquare location dataset and determine the best region to open new fire and rescue station.

PROBLEM STATEMENT

Cities are really busy when it comes to traffic flow in United States. People often take the shortest route to get to their destinations and some often get crashed in different situations like, going through the red light, taking turns while speeding, driving under influence etc. A small mistake in busy area would cause severe problems to the driver himself/herself and also can be dangerous to the other drivers. And if the crash occurs, in busy areas, the whole route or highway will be affected and the entire traffic flow gets locked. In order to avoid this situation, fire and rescue operators should be on time and as soon as possible to minimize the severity that has occurred to the driver and also to keep the flow of traffic moving since people needs to be on time for their appointments, offices, gatherings etc. Fire and rescue station nearby the location of the crash would be highly facilitate the work on getting things done as quickly as possible. For example, if the highway 286 near Arlington Boulevard exit suffers more crash, we can easily setup new fire station nearby that location.

DATASET

Since I am working on crash dataset of the Washington DC, I got the dataset from https://opendata.dc.gov/datasets/70392a096a8e431381f1f692aaa06afd_24/data . This dataset has the crash details, location, vehicle type, severity of casualties etc and is being updated. For the data manipulation purpose, I have decided to take only on 50000 rows since there are more than 200,000 rows. Also, the columns/attributes I have used are based on the requirement of needs for this project. There are columns/attributes which were unneeded and thus I have redfined the dataset and got the right attributes for the computation. I have taken following columns into the consideration:

1. RouteID
2. Address
3. Latitude
4. Longitude

RouteID and Address are the columns to determine which location is in need of the fire station after we the get the final ratio. Longitude and Latitude gives the power to use the foursquare location to get the necessary data of nearby fire station and thus do the computation of ratio of number of crashes (from original dataset) to number of station(from foursquare location dataset).

FOURSQUARE LOCATION DATASET

The foursquare location comes handy when determining the nearby neighborhood which has the fire and rescue station. This API helps in getting us the dataset that has location with latitude and longitude and as well as number of stations in the particular neighborhood. For example, if the highway 286 is the maximum crash road in 50th exit, with the 15th Baker Road, being the closest neighborhood that has fire station, the foursquare api call will return the number of stations in 15th Baker Road and as well as its, longitude and latitude. Then, with this dataset, we can merge it to our crash location dataset so as to find out the ratio of number of crash to number of stations. The highest ratio address needs new rescue operators nearby.

METHODOLOGY

With the problem and dataset we have, we do the following steps in order to complete our methodology.

i. Import Libraries:

The very first thing to do before starting to develop our model is to know what libraries are necessary for our model and we import those libraries.

Downloaded necessary libraries

In [1]:
```
import requests # library to handle requests
import pandas as pd # library for data analsysis
import numpy as np # library to handle data in a vectorized manner
import random # library for random number generation

#!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # module to convert an address into latitude and longitude values

# libraries for displaying images
from IPython.display import Image
from IPython.core.display import HTML

# tranforming json file into a pandas dataframe library
from pandas.io.json import json_normalize

#!conda install -c conda-forge folium=0.5.0 --yes
import folium # plotting library
from pyproj import Proj,transform
from folium.plugins import FastMarkerCluster
from folium.plugins import MarkerCluster
from sklearn.cluster import DBSCAN

import matplotlib.pyplot as plt
```

ii. Import dataset:

For the model, the important part is to have proper dataset. For simplicity and my machine purposes, I have decided to take on only 50,000 rows. We use pandas library to read csv dataset.

```
In [2]: dataframe = pd.read_csv('Crashes_in_DC.csv',nrows=50000)
```

```
/home/rdhakal2/anaconda3/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3020: DtypeWarning: Columns
(4,54,55) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

iii. Data cleaning/redefining:

There are numerous columns/attribute section for our dataset. Thus, I tried to take only on needed attributes. As I discussed earlier in Dataset section, I have redfined the dataset and got the right attributes for the computation. I have taken following columns into the consideration:

1. RouteID
2. Address
3. Latitude
4. Longitude

RouteID and Address are the columns to determine which location is in need of the fire station after we the get the final ratio.

Furthermore, I have dropped the non available (NaN) datas for the simplicity of our redefining/cleaning of data.

Data cleaning and Redefining process.

```
In [4]: dataframe = dataframe[['REPORTDATE','ROUTEID','ADDRESS','LATITUDE','LONGITUDE']]
```

```
In [5]: dataframe.head(10)
```

Out[5]:

	REPORTDATE	ROUTEID	ADDRESS	LATITUDE	LONGITUDE
0	2014-10-18T05:00:00.000Z	15065322	700 NORTH CAPITOL STREET NW	38.899699	-77.009458
1	2014-10-17T05:00:00.000Z	11091452	1000 WISCONSIN AVE NW	38.902633	-77.063004
2	2014-10-17T05:00:00.000Z	11087232	U ST NW / VERMONT AVE NW	38.916998	-77.025368
3	2014-10-18T05:00:00.000Z	11080082	GIRARD ST NW and SHERMAN AVE NW	38.925869	-77.025803
4	2018-03-26T02:36:40.000Z	11000602	600 6TH ST NW	38.898252	-77.019906
5	2013-11-22T05:00:00.000Z	11025152	4418 CONNECTICUT AVENUE NW	38.947134	-77.065632
6	2016-10-21T14:14:15.000Z	11040042	3700 GEORGIA AVENUE NW	38.937083	-77.024528
7	2016-10-21T14:21:24.000Z	47008562	1805 5TH STREET NW	38.914416	-77.018738
8	2014-10-18T05:00:00.000Z	11057852	34TH ST NW and M ST NW	38.905065	-77.067830
9	2015-10-27T18:18:59.000Z	11094252	3515 WOODLEY RD NW	38.931816	-77.069962

```
In [6]: dataframe = dataframe.dropna()
        dataframe.head()
```

Out[6]:

	REPORTDATE	ROUTEID	ADDRESS	LATITUDE	LONGITUDE
0	2014-10-18T05:00:00.000Z	15065322	700 NORTH CAPITOL STREET NW	38.899699	-77.009458
1	2014-10-17T05:00:00.000Z	11091452	1000 WISCONSIN AVE NW	38.902633	-77.063004
2	2014-10-17T05:00:00.000Z	11087232	U ST NW / VERMONT AVE NW	38.916998	-77.025368
3	2014-10-18T05:00:00.000Z	11080082	GIRARD ST NW and SHERMAN AVE NW	38.925869	-77.025803
4	2018-03-26T02:36:40.000Z	11000602	600 6TH ST NW	38.898252	-77.019906

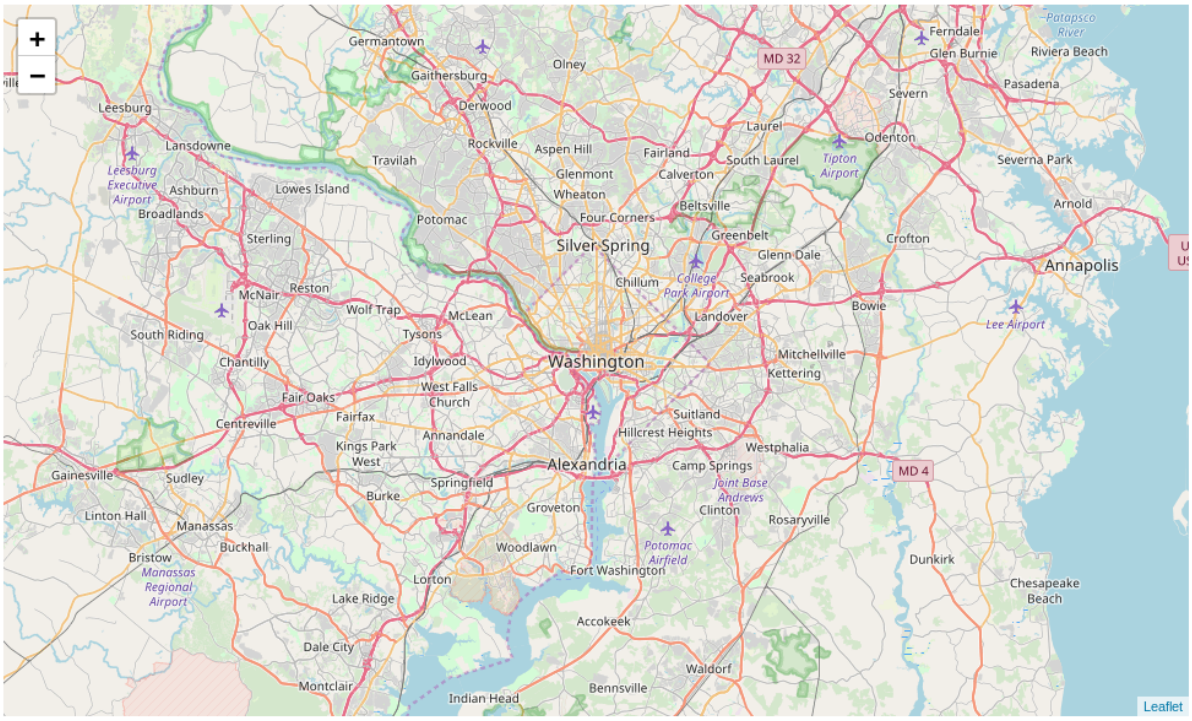
iv. DC map and Markers of crash location using Folium library:

For the geolocation map, we need to use folium library. I have added the markers of the crash location labeling them with the crash address and adding all of these components in DC map.

We use Nominatim library to convert address into longitude and latitude values so as to use them for the folium library to create map.

In [8]:
```
DCmap = folium.Map(location=[38.8950092,-77.0365625],zoom_start=10)
DCmap
```

Out[8]:



In [9]:
```
# add markers to map
for lat, lng, rid, address in zip(dataframe['LATITUDE'], dataframe['LONGITUDE'], dataframe['ROUTEID'] ,dataframe['ADDRESS
    label = '{},{}'.format(rid,address)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(DCmap)

DCmap
```

v. Redefining dataset in order of maximum number of crashes:

I have now created the new dataframe that has the crash area/location along with the number of crashes in those areas. The dataframe is in descending order.

Finding out the number of crashes per venues

In [14]:
```
df_top_frequency = dataframe.groupby(['ADDRESS','LATITUDE', 'LONGITUDE','ROUTEID'])['ADDRESS'].agg(
    {"countCrash": len}).sort_values(
    "countCrash", ascending=False).head(10).reset_index()
```

```
/home/rdhakal2/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWarning: using a dict on a Ser
ies for aggregation
is deprecated and will be removed in a future version
```

In [15]:
```
df_top_frequency
```

Out[15]:

	ADDRESS	LATITUDE	LONGITUDE	ROUTEID	countCrash
0	INTERSTATE 695 INTERSTATE BN	38.871965	-76.989817	15048473	36
1	2300 PENNSYLVANIA AVE SE	38.874461	-76.971676	47057082	32
2	INTERSTATE 295 INTERSTATE BN	38.845163	-77.007218	15048451	31
3	INTERSTATE 395 INTERSTATE BN	38.882536	-77.017728	15048463	30
4	2300 PENNSYLVANIA AVENUE SE	38.874461	-76.971676	47057082	30
5	100 NEW YORK AVENUE NE	38.908659	-77.005281	12064672	24
6	850 HOWARD ROAD SE	38.864858	-76.995716	58023012	23
7	BLADENSBURG RD NE and NEW YORK AVE NE	38.917329	-76.972439	12064672	21
8	2305 PENNSYLVANIA AVENUE SE	38.873990	-76.973406	47065562	21
9	FIRTH STERLING AVE SE and SUITLAND PKWY SE	38.862369	-76.997739	13083422	21

vi. FOURSQUARE API Call:

Now, that I have the dataset that number of crashes per venues, I have called foursquare api to get the new dataframe that has regions/venues nearby crash location those have fire and rescue station and arrange the dataset in descending order.

In [19]:
```
countHealthService = getNearbyVenues(names=df_top_frequency['ADDRESS'],
                                      latitudes=df_top_frequency['LATITUDE'],
                                      longitudes=df_top_frequency['LONGITUDE']
                                     )
```

```
INTERSTATE 695 INTERSTATE BN
2300 PENNSYLVANIA AVE SE
INTERSTATE 295 INTERSTATE BN
INTERSTATE 395 INTERSTATE BN
2300 PENNSYLVANIA AVENUE SE
100 NEW YORK AVENUE NE
850 HOWARD ROAD SE
BLADENSBURG RD NE and  NEW YORK AVE NE
2305 PENNSYLVANIA AVENUE SE
FIRTH STERLING AVE SE and  SUITLAND PKWY SE
```

In [20]:
```
df_top_freq = countHealthService.groupby(['Neighborhood'])['Neighborhood'].agg(
    {"counts": len}).sort_values(
    "counts", ascending=False).head(10).reset_index()
```

```
/home/rdhakal2/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWarning: using a dict on a Ser
ies for aggregation
is deprecated and will be removed in a future version
```

Number of Fire Rescues nearby crash area

In [22]:
```
df_top_freq
```

Out[22]:

	Neighborhood	counts
0	INTERSTATE 395 INTERSTATE BN	3
1	2300 PENNSYLVANIA AVE SE	1
2	2300 PENNSYLVANIA AVENUE SE	1
3	BLADENSBURG RD NE and NEW YORK AVE NE	1
4	INTERSTATE 295 INTERSTATE BN	1
5	INTERSTATE 695 INTERSTATE BN	1

The getNearbyVenue is a function that has the api call to get location of fire stations nearby.

We get the nearby venues in terms of Fire and Rescue Station

In [18]:
```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    search_query = 'Fire Station'
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&query={}&ll={},{}&r
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            search_query,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
#             lat,
#             lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['location']['distance'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['Neighborhood',
#                  'Neighborhood Latitude',
#                  'Neighborhood Longitude',
                 'Venue',
                 'Venue Latitude',
                 'Venue Longitude',
                  'Distance',
                 'Venue Category']

    return(nearby_venues)
```

vii. Merge two datasets.

I now merge two datasets, ie, the dataset which has crash counts and the dataset which has count of fire stations. I have also decided to use scatter plots to see the number of crash vs number of fire station ratio visually.

We create the final dataframe merging the count of fire and rescue station with out original dataframe.

In [26]:
```
finalDF = df_top_freq.merge(df_top_frequency)
```

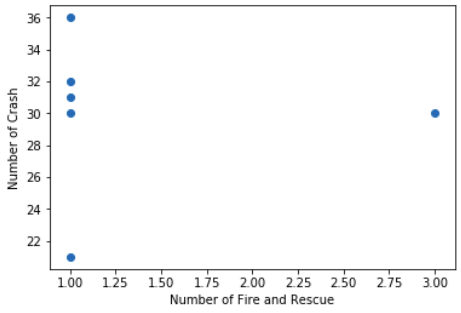
In [27]:
```
finalDF.head(10)
```

Out[27]:

	ADDRESS	counts	LATITUDE	LONGITUDE	ROUTEID	countCrash
0	INTERSTATE 395 INTERSTATE BN	3	38.882536	-77.017728	15048463	30
1	2300 PENNSYLVANIA AVE SE	1	38.874461	-76.971676	47057082	32
2	2300 PENNSYLVANIA AVENUE SE	1	38.874461	-76.971676	47057082	30
3	BLADENSBURG RD NE and NEW YORK AVE NE	1	38.917329	-76.972439	12064672	21
4	INTERSTATE 295 INTERSTATE BN	1	38.845163	-77.007218	15048451	31
5	INTERSTATE 695 INTERSTATE BN	1	38.871965	-76.989817	15048473	36

In [28]:
```
plt.scatter(finalDF['counts'],finalDF['countCrash'])
plt.xlabel("Number of Fire and Rescue")
plt.ylabel("Number of Crash")
plt.plot()
```

Out[28]: []



viii. Get the final ratio

Now, I calculate the ratio of number of crash (countCrash) to counts of fire station. The result gives me Interstate 395 Interstate BN, routeID of 15048463, latitude and longitude of 38.872 and -76.9898 respectively. This region should need new fire station .

In [23]:
```
finalDF['ratio'] = finalDF['countCrash'] / finalDF['counts']
```

In [24]:
```
print(finalDF.loc[finalDF['ratio'].idxmax()])
```

```
ADDRESS       INTERSTATE 695 INTERSTATE BN
counts                                   1
LATITUDE                            38.872
LONGITUDE                         -76.9898
ROUTEID                           15048473
countCrash                              36
ratio                                   36
Name: 5, dtype: object
```

We found that the interstate 395, the route id of 15048463 has highest ratio and thus needs the fire and rescue station nearby.

RESULTS

From the above discussion, I found out that the Interstate 395 Interstate BN, routeID of 15048463, latitude and longitude of 38.872 and -76.9898 respectively should have new fire station in order to cope with the high number of crashes.

CONCLUSION AND DISCUSSION

Foursquare API has the power to manipulate the geolocation dataset and hence not only we can determine the region to set up new fire station but also can figure out all the areas which are in need of fire and rescue operators nearby crash region. We can extend this project to different domains like finding out the location where particular vehicles looks like getting crashed and hence having the mobile fire operators inorder to minimize the crash in busy cities like DC.