IBM Applied Data Science Capstone Project

**Subject: Impact of Local Venues in COVID-19 Spread :
A Comparison Between London and New York**

Semanur Kapusızoğlu
semanurkps@gmail.com

2020

Table of Contents

1. Introduction

Data science is the art of manipulating data to gain insights and knowledge from it with the help of knowledge, tools. The scope of this course is to teach us how this process is handled in order to find data-driven solutions to problems we face.

One of the biggest problems of humanity is COVID-19 pleague nowadays. It can be thought as highly contagious flu-like virus. Many data scientists are working on different cases regarding the issue. After acquiring some knowledge withthe help of the course track, the author wanted to apply them in a real-world case and see the effect of data driven solutions.

2. Business Problem

Main problem of the new virus type is that it is so contagious. Scientists say it can contaminate you by touching, from air etc. Although some countries did not take this seriously at first, now many declared quarantine to slow down the spread. During he negligence time, public places took an important role. The project aims to find the relation between the number o confirmed cases (till 05.04.2020) and commony visited public venues. We will cluster the neighborhoods by taking the number of cconfirmed cases into consideration, then inside the clusters, we will find which type of venues are more likely to be visited by those neighborhoods. We will comment on them afterwards in the discussion section.

The analysis concerns people who live in these neighborhoods and also maybe the places they visit in the neighborhood. Israel conducts a similar analysis during these times. They detect the location of patients, track down where they have been in the last 14 days and notify people around that places to take corrective measures and be evenore careful. Since no such data is available, this research will be a simplified version of it. Hopefully it will be enough to give people an idea to stay away from the most commonly visited places where the number of cases is high.

3. Data

Since it is a sensitive issue, it is hard to obtain timely and accurate data regarding the issue. That's why governmental websites are used as primary resource. United Kingdom posts the number of infected people in a Public Health England website. Excel reports including death rates, number of infected and number of healed cases are available. When it comes to United States, the number of cases by neighborhoods was available on Kaggle. Cases of New York neighborhoods are extracted from that file. Latitude and longitude values are obtained from combination of Wikipedia and Google. Data parts are combined with Excel, turned to a neat form that will allow to continue our analysis with python.

Getting to know the datasets:

usdata: Contains New York neighborhoods, their locations, population

Data source: Kaggle (for the number of cases in each neighborhood)

Google (for latitude ad longitude values)

```
Neighborhood     object
Population        int64
Latitude        float64
Longitude       float64
cases             int64
dtype: object
```

	Population	Latitude	Longitude	cases
count	5.700000e+01	57.000000	57.000000	57.000000
mean	2.245278e+05	42.447361	-75.532777	5608.403509
std	4.541461e+05	0.911117	1.743888	19083.955749
min	3.520000e+03	39.625500	-79.466800	4.000000
25%	3.004600e+04	42.008400	-76.623800	70.000000
50%	7.588000e+04	42.601200	-75.165200	148.000000
75%	2.237740e+05	43.025600	-73.962600	1021.000000
max	3.116069e+06	44.447300	-72.615100	102386.000000

ukdata: Contains Londan neighborhoods, their locations, population

Data source: Public Health England (for the number of cases in each neighborhood)

Wikipedia (for latitude and longitude values)

```
Neighborhood     object
Population      float64
Latitude        float64
Longitude       float64
Total Cases       int64
dtype: object
```

	Population	Latitude	Longitude	Total Cases
count	32.000000	32.000000	32.000000	32.000000
mean	277.503188	51.505666	-0.119197	363.531250
std	61.422621	0.072662	0.161904	163.914018
min	156.197000	51.361800	-0.476000	0.000000
25%	245.229000	51.456250	-0.205325	251.000000
50%	278.182500	51.505600	-0.114100	320.500000
75%	326.056250	51.558850	-0.011525	489.000000
max	392.140000	51.653800	0.183700	728.000000

Foursquare Data: Venue information will be withdrawn from Foursuare

usdata and ukdata will be used to cluster neighborhoods based on the number of confirmed cases, seperately. After the clusters are formed, with the help of Foursquare location data, most commonly visited venues will be listed and venue types will be compared to see if a certain type has affect on the spread. London and New York will be evaluated separately to see how different countries' big cities are affected by the virus

Descriptive statistics:

Is there a relationship between population and the number of cases? That is a tricky question because it might be a yes or no when you whink about it. If corrective measures are taken as early as possible. This will be examined for our two cities, New York and London.

Let's see the correlation coefficients.

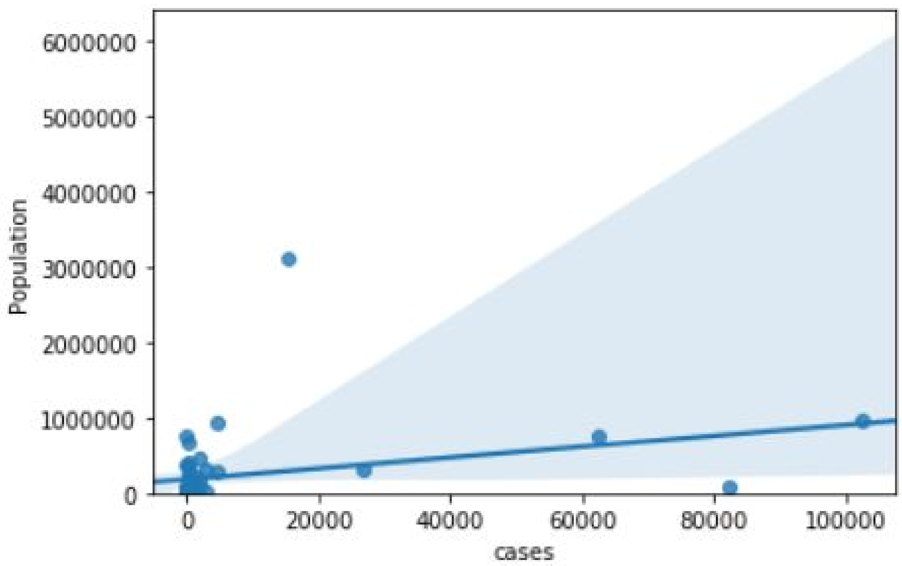
```
ny_df[["Population","cases"]].corr()
```

	Population	cases
Population	1.000000	0.301913
cases	0.301913	1.000000

For New York, correlation coefficient is 0.3019. So we can not say there exists a relationship. Here is the graph to see it clearly.

```
sns.regplot(x="cases", y="Population", data=ny_df)
plt.ylim(0,)
```

```
(0, 6395708.729889891)
```



For London, we can see that situation is different. Population and cases can be accepted as correlated. This can be result of taking corrective actions later.

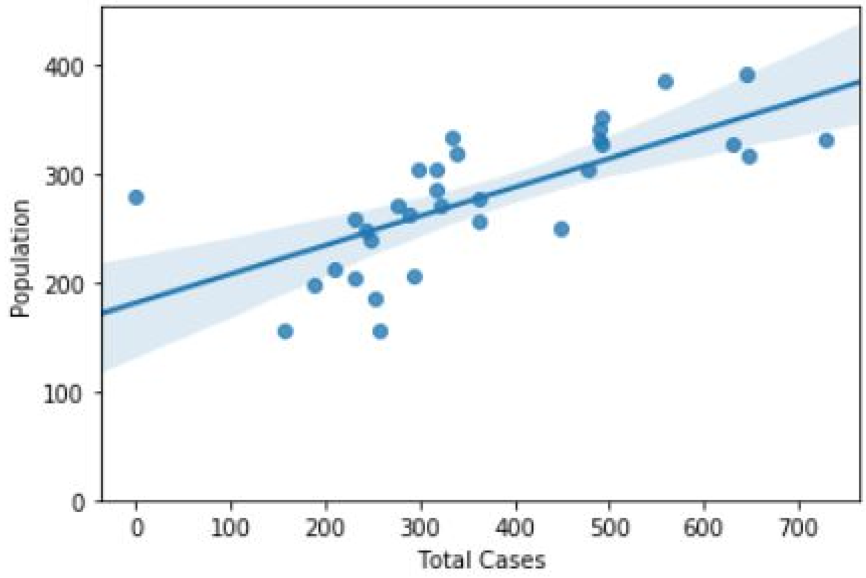
```
lon_df[["Population","Total Cases"]].corr()
```

	Population	Total Cases
Population	1.000000	0.708661
Total Cases	0.708661	1.000000

It can also be seen on the graph as follows.

```
sns.regplot(x="Total Cases", y="Population", data=lon_df)
plt.ylim(0,)
```

```
(0, 453.2948313607337)
```



4. Methodology

Clustering:

K-means clustering is used to separate neighborhoods in 4 different groups by taking the number if cases into consideration. The process is followed for both New York and London datasets.

Before starting the process, object type variables are eliminated from the dataset. Example code is as follows

```python
# dropping object columns to make the dataset ready for clustering
lon_kmeans=lon_df[:]
lon_kmeans.drop(columns=['Latitude','Longitude','Neighborhood'], inplace=True)
lon_kmeans.head()
```

```python
# set number of clusters
kclusters = 4

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(lon_kmeans)

# check cluster labels generated for each row in the dataframe
print(kmeans.labels_[0:10])

# add clustering labels
lon_df.insert(0, 'Cluster Labels', kmeans.labels_)
lon_df.head()
```

Foursquare Venue Data:

Foursquare is used after the clusters are formed. Each cluster will be evaluated in terms of venue types and most visited ones will be listed.

Required functions:

```python
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']
```

```python
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    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={}&ll={},{}&radius={}&limit={}'.forr
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            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']['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',
                  'Venue Category']

    return(nearby_venues)
```

Venue Calls:

```
#calling all venues in new york neighborhoods

ny_venues = getNearbyVenues(names=ny_df['Neighborhood'],
                                   latitudes=ny_df['Latitude'],
                                   longitudes=ny_df['Longitude']
                                  )
print(ny_venues.shape)
ny_venues.head()
```

```
# expanding the columns
ny_venues_exp = pd.get_dummies(ny_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
ny_venues_exp['Neighborhood'] = ny_venues['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [ny_venues_exp.columns[-1]] + list(ny_venues_exp.columns[:-1])
ny_venues_exp = ny_venues_exp[fixed_columns]

ny_venues_exp.head()
```

Most Common Venues:

```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

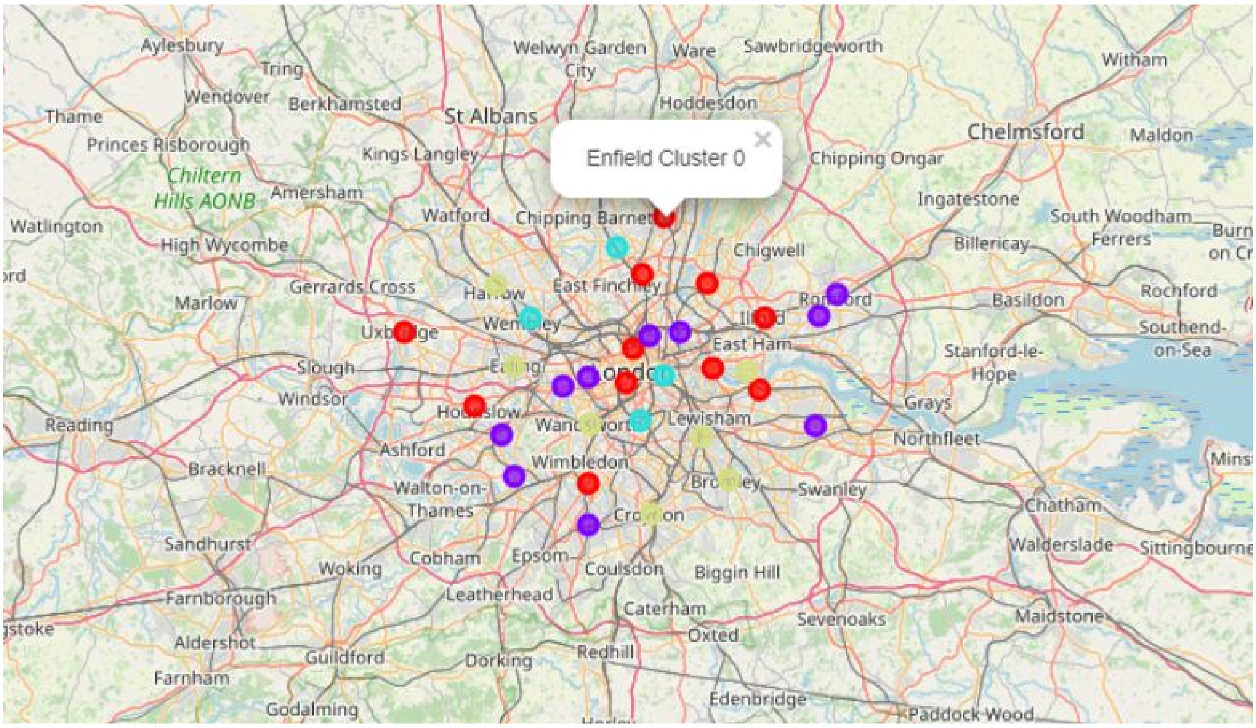
```
num_top_venues = 3

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
ny_venues_sorted = pd.DataFrame(columns=columns)
ny_venues_sorted['Neighborhood'] = ny_grouped['Neighborhood']

for ind in np.arange(ny_grouped.shape[0]):
    ny_venues_sorted.iloc[ind, 1:] = return_most_common_venues(ny_grouped.iloc[ind, :], num_top_venues)

ny_venues_sorted.head()
```

Folium:
Folium is used to cluster neighborhoods and visualize it on the map.

The code:

```
# Let's visualize

latitude=51.5074   #latitude and longitue values for New York
longitude=-0.1278
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(lon_df['Latitude'], lon_df['Longitude'], lon_df['Neighborhood'], lon_df['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Resulting Map:



5. Results

Now we both have the cclusters and the most commonly visited venues. When we combine them into one list, we can analyze each cluster. Following table summarizes all 8 clusters for both New York and London.

New York	**1st Common Venue**	**2nd Common Venue**	**3rd Common Venue**
Cluster 0	Pizza Place	Restaurants	Cafe-Coffee Places
Cluster 1	Deli / Bodega	Yoga Studio	Dessert Shop
Cluster 2	Yoga Studio	Cafe-Coffee Places	Beach
Cluster 3	Gym	Cafe-Coffee Places	Shops
London			
Cluster 0	Cafe-Coffee Places	Clothing store	Pub
Cluster 1	Pub	Cafe	Clothing Store
Cluster 2	Cafe-Coffee Places	Pub	Restaurants
Cluster 3	Cafe-Coffee Places	Clothing Store	Restaurants

6. Discussion

From the results, it can be seen that for New York, commonly visited places differ among clusters. Also New York clusters are highly different than each other in terms of size, that might be result of taking seperate measures for all neighborhoods. When it comes to London, we can say that Coffee places and pubs might be the places where people infected each other. Also clusters in London seems more equally distributed when compared to New York. Both cities faced the crisis starting from different dates, so that means the comparison is biased. London has a lot less casses.

However, the effect of public places is important when it comes to a highly contagious disease like this. It is obvious that crowded places like Coffee Places and Pubs most probably set an environment for containment.

7. Conclusion and Further Notice

Before diving into the details, the plan was to compare how the preventive actions taken by those big cities, what is the effect of public places in this pleague. I was hoping to emhasize the importance of taking effective measures earlier and slowing the spread. However, when I further think about it, I should have selected cities with similar timelines. For example first case seen on the same day or consecutive day to compare the two cities better, without bias.