IBM Data Science

Coursera Capstone

Coursera Capstone

Stewart Foodservice Inc. POS in Beirut, Lebanon

Detailed Report

By Ayham Mhd

July, 2020

Contents

1. Introduction

This report illustrates Coursera Capstone related to IBM Data science course.

1.1. Stake holder overview

Stewart Foodservice is owned by industry veteran Aubrey MacMillan. He started out in the business over 50 years ago at Burgess Wholesale, then an egg-grading station, working his way up through deliveries, sales and management to owner of what has become one of the leading Canadian owned foodservice distributors.

1.2. Problem Definition

Stewart Foodservice Inc, who supplies restaurants and so on, decided to open a store or a point of sale POS in Beirut, Lebanon, and asked me an engineer having some background in data science that collected from IBM data science courses on Coursera, to suggest the place where to install this POS.

1.3. Initial thinking

As the company is the first point in food industry supply chain, and because the distance between the supplier and the client can shorten the time to market (TTM) which is a cost, in addition, the more this distance the more the cost of delivery, so I thought that selecting the right closest place of POS to the targeted market (restaurant, coffee shops and so forth) could be the most important factor.

p.s. TTM is the length of time it takes from a product being conceived until its being available for sale (Wikipedia).

2. Description of data

I've collected data from Forsquare API using my credentials like the following:

```
CLIENT_ID = 'My Foursquare ID'
CLIENT_SECRET = 'Foursquare Secret'
VERSION = '20180604'
LIMIT = 100
RADIUS = 2000
```

```
request_parameters = {
    "client_id": CLIENT_ID,
    "client_secret": CLIENT_SECRET,
    "v": '20180605',
    "section": "restaurant",
    "near": "Beirut",
    "radius": RADIUS,
    "limit": LIMIT}

data = requests.get("https://api.foursquare.com/v2/venues/explore", params=request_parameters)
```

This gave me a comprehansive data enabling me to extract many meaningful pointers, I extracted the data filtered as I want certain categories of venues, i.e. only related to targeted customers who are restaurants, coffee shops and so forth, and I chose the following useful fields: name of venue, category, longitude and latitude as the following:

name	categories	lat	lng
Moscow Mule	Bar	33.896835	35.478505
Cafe Younes	Café	33.895523	35.479906
Classic Burger Joint	Burger Joint	33.896285	35.477883
Roadster Diner	Diner	33.896052	35.479539
Socrate	Middle Eastern Restaurant	33.897890	35.478478

3. Methodology

It was a descriptive research consists of the following steps after introductory part:

3.1. Preparing the environment

I used ***numpy*** to handle data in a vectorized manner, and will use ***pandas*** library for data analysis and json library to handle JSON files, and will use geopy to be able to use ***geocoders***.***Nominatim*** to convert an address into latitude and longitude values, and ***requests*** library to handle requests, and ***json_normalize*** that will transform JSON file into a pandas data frame.

I used ***Matplotlib*** and associated plotting modules, and ***k-means*** for clustering stage.

Then I used ***folium*** map rendering library.

First importing prereqisites

```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files

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

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

#import k-means from clustering stage
from sklearn.cluster import KMeans

!conda install -c conda-forge folium=0.5.0 --yes
import folium # map rendering library

print('Prerequisit Packages were Loaded...')
```

3.2. Exploring Beirut City

I used geopy library to get the latitude and longitude values of Beirut City,

```
address = 'Beirut'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Beirut City are {}, {}.'.format(latitude, longitude))
```

```
The geograpical coordinate of Beirut City are 33.8959203, 35.47843.
```

then using my Forsquare credentials I got the top 100 venues that are in Beirut within a radius of 500 meters.

```
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius

# create URL
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)
url # display URL
```

Then I used a function that extracts the category of the venue then clean the json and structure it into a pandas data frame.

```
#From Foursquare lab all the information is in the items key. Before we proceed, let's borrow the get_category_type function from the Foursquare lab.
# 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']
```

```
#Now we are ready to clean the json and structure it into a pandas dataframe.
venues = results['response']['groups'][0]['items']

nearby_venues = pd.json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]

nearby_venues.head()
```

Then I filtered the result categories to match our target (Restaurants and etc.)

```
nearby_venues = nearby_venues.loc[nearby_venues['categories'].str.contains('Restaurant') | nearby_venues['categories'].str.contains('Bar')
                        | nearby_venues['categories'].str.contains('Sandwich')| nearby_venues['categories'].str.contains('Burger')
                        | nearby_venues['categories'].str.contains('Break')| nearby_venues['categories'].str.contains('Coffee')
                        | nearby_venues['categories'].str.contains('Diner')| nearby_venues['categories'].str.contains('Diner')
                        | nearby_venues['categories'].str.contains('Caf')]
```

```
nearby_venues.head()
```

	name	categories	lat	lng
3	Moscow Mule	Bar	33.896835	35.478505
5	Cafe Younes	Café	33.895523	35.479906
6	Classic Burger Joint	Burger Joint	33.896285	35.477883
9	Roadster Diner	Diner	33.896052	35.479539
10	Socrate	Middle Eastern Restaurant	33.897890	35.478478

3.3. Exploring Neighborhoods

I used a function to repeat the same process to all the neighborhoods in Beirut using Foursquare API as well, creating a new data frame which also must be filtered.

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
3	Moscow Mule	33.896835	35.478505	Moscow Mule	33.896835	35.478505	Bar
5	Moscow Mule	33.896835	35.478505	Cafe Younes	33.895523	35.479906	Café
6	Moscow Mule	33.896835	35.478505	Fiber	33.898009	35.479450	Restaurant
7	Moscow Mule	33.896835	35.478505	Classic Burger Joint	33.896285	35.477883	Burger Joint
8	Moscow Mule	33.896835	35.478505	Socrate	33.897890	35.478478	Middle Eastern Restaurant

Then I analyzed each neighborhood by one hot encoding.

Analyze Each Neighborhood

```
# one hot encoding
t_onehot = pd.get_dummies(t_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
t_onehot['Neighborhood'] = t_venues['Neighborhood']

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

t_onehot.head()
```

	Neighborhood	American Restaurant	Bar	Beach Bar	Bed & Breakfast	Breakfast Spot	Burger Joint	Café	Chinese Restaurant	Cocktail Bar	Coffee Shop	Diner	Eastern European Restaurant	Fast Food Restaurant	French Restaurant	Hookah Bar	Italian Restaurant	Juice Bar	Lebanese Restaurant	Mediterranean Restaurant
3	Moscow Mule	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	Moscow Mule	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
6	Moscow Mule	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	Moscow Mule	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	Moscow Mule	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Then grouping rows by neighborhood by taking the mean of the frequency of occurrence of each category, then putting that into a pandas data frame.

```
t_grouped = t_onehot.groupby('Neighborhood').mean().reset_index()
t_grouped
```

	Neighborhood	American Restaurant	Bar	Beach Bar	Bed & Breakfast	Breakfast Spot	Burger Joint	Café	Chinese Restaurant	Cocktail Bar	Coffee Shop	Diner	Eastern European Restaurant	Fast Food Restaurant	French Restaurant	Hookah Bar	Italian Restaurant	Juice Bar	Lebanese Restaurant
0	Abo Wasseem	0.046512	0.046512	0.000000	0.000000	0.023256	0.023256	0.139535	0.000000	0.046512	0.093023	0.046512	0.000000	0.046512	0.023256	0.000000	0.046512	0.000000	0.046512
1	Abu Hassan	0.000000	0.034483	0.034483	0.000000	0.034483	0.034483	0.275862	0.000000	0.000000	0.103448	0.000000	0.034483	0.000000	0.000000	0.034483	0.034483	0.000000	0.068966
2	Abu Naim Restaurant مطعم أبو نعيم	0.054054	0.027027	0.000000	0.000000	0.027027	0.027027	0.135135	0.000000	0.054054	0.108108	0.081081	0.000000	0.000000	0.054054	0.000000	0.027027	0.000000	0.081081
3	Al Kahwa	0.035714	0.053571	0.000000	0.017857	0.017857	0.017857	0.160714	0.000000	0.035714	0.071429	0.053571	0.000000	0.053571	0.035714	0.000000	0.053571	0.000000	0.017857
4	Ali's Nescafe/Tea Truck	0.034483	0.068966	0.000000	0.000000	0.000000	0.034483	0.172414	0.000000	0.000000	0.103448	0.034483	0.034483	0.000000	0.000000	0.000000	0.103448	0.034483	0.068966
5	Appetito Trattoria	0.041667	0.041667	0.000000	0.000000	0.041667	0.020833	0.145833	0.000000	0.041667	0.083333	0.041667	0.000000	0.062500	0.020833	0.000000	0.041667	0.000000	0.041667
6	B.Hive	0.046512	0.046512	0.000000	0.000000	0.023256	0.023256	0.139535	0.000000	0.046512	0.116279	0.023256	0.000000	0.046512	0.023256	0.000000	0.046512	0.000000	0.046512
7	Bagatelle	0.034483	0.051724	0.000000	0.017241	0.000000	0.017241	0.155172	0.000000	0.034483	0.103448	0.051724	0.000000	0.034483	0.034483	0.000000	0.051724	0.000000	0.051724

3.4. *Clustering Neighborhoods*

I clustered neighborhoods using Kmean as following:

Cluster Neighborhoods

```
# set number of clusters
kclusters = 5

t_grouped_clustering = t_grouped.drop('Neighborhood', 1)

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

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

```
array([0, 3, 1, 4, 2, 0, 0, 4, 1, 0], dtype=int32)
```

```
#Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

t_merged = t_venues

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighborhood
t_merged = t_merged.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

t_merged.head() # check the last columns!
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
3	Moscow Mule	33.896835	35.478505	Moscow Mule	33.896835	35.478505	Bar	0	Restaurant	Café	Middle Eastern Restaurant	Coffee Shop	Sandwich Place	Fast Food Restaurant	Turkish Restaurant	Sushi Restaurant	Bar	Breakfast Spot
5	Moscow Mule	33.896835	35.478505	Cafe Younes	33.895523	35.479906	Café	0	Restaurant	Café	Middle Eastern Restaurant	Coffee Shop	Sandwich Place	Fast Food Restaurant	Turkish Restaurant	Sushi Restaurant	Bar	Breakfast Spot
6	Moscow Mule	33.896835	35.478505	Fiber	33.898009	35.479450	Restaurant	0	Restaurant	Café	Middle Eastern Restaurant	Coffee Shop	Sandwich Place	Fast Food Restaurant	Turkish Restaurant	Sushi Restaurant	Bar	Breakfast Spot
7	Moscow Mule	33.896835	35.478505	Classic Burger	33.896285	35.477883	Burger Joint	0	Restaurant	Café	Middle Eastern Restaurant	Coffee Shop	Sandwich Place	Fast Food Restaurant	Turkish Restaurant	Sushi Restaurant	Bar	Breakfast Spot

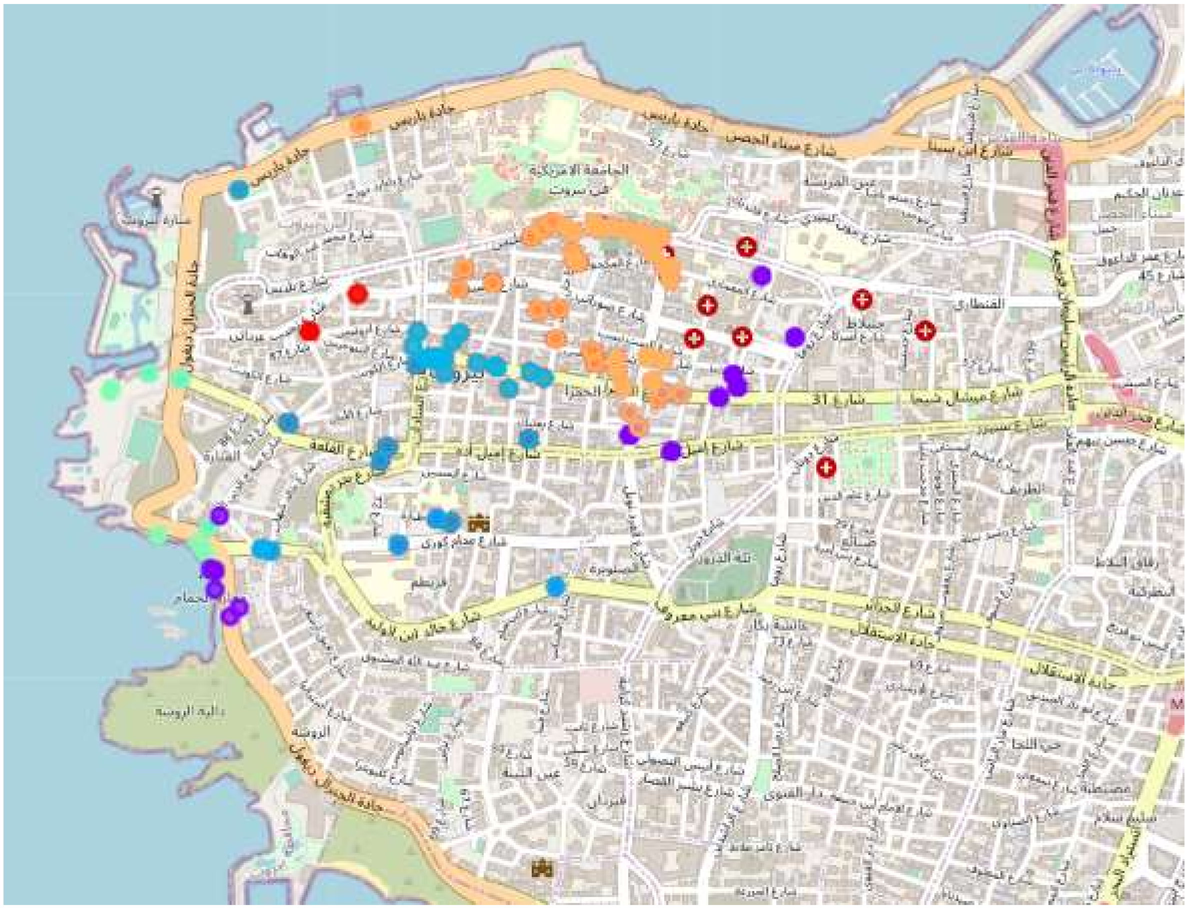
and visualize the resulting clusters using Folium library:

```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=15)

# 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(t_merged['Venue Latitude'], t_merged['Venue Longitude'], t_merged['Neighborhood'], t_merged['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
```



4. The Result

I chose the cluster that have the largest number of neighborhoods, get the average coordinates to plot the suggested POS location, this way it will be close to maximum number of the customers, and by this way I can consider that I had chosen the optimal place for the POS because of reducing TTM and delivery and shipping fees.

```
cluster0 = t_merged.loc[t_merged['Cluster Labels'] == 0]
```

```
t_merged.groupby(t_merged['Cluster Labels']).count()
```

Cluster Labels	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	835	835	835	835	835	835	835	835	835	835	835	835	835	835	835	835	835
1	534	534	534	534	534	534	534	534	534	534	534	534	534	534	534	534	534
2	133	133	133	133	133	133	133	133	133	133	133	133	133	133	133	133	133
3	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
4	654	654	654	654	654	654	654	654	654	654	654	654	654	654	654	654	654

```python
Lat = cluster0['Venue Latitude'].mean()
Lng = cluster0['Venue Longitude'].mean()
print('Latitude is {}, and Longitude is {}'.format(Lat, Lng))
```

```
Latitude is 33.89664370405219, and Longitude is 35.479637994214215
```

Let's add a circle marker where suggested POS place

```python
folium.CircleMarker(
    [Lat, Lng],
    radius=15,
    popup="Suggested POS Place!",
    color='green',
    fill=True,
    fill_color='#3186cc',
    fill_opacity=0.7,
    parse_html=False).add_to(map_clusters)
map_clusters
```