Battle of The Neighborhoods Capstone

Coffeeshop Project

Introduction

As a part of the final IBM Capstone Project, we have to determine what data scientists go through in real life. Objectives of the final assignments were to define a business problem, look for data in the web and use Foursquare, and Google location data to compare different districts within Jeddah city to find out out which neighborhood is suitable to start a **coffeeshop** business. As preparation for this assignment, I will go through the problem discussion, data preparation, and final analysis section, moreover; a detailed codes and images are given in GitHub and link can be found at the end of the post.

Business Problem

Jeddah is the second largest city in Saudi Arabia, it is located at the red-sea coast in the western province, it is far from the holy city of Makkah by 70km. However; in this project I will try to find a suitable location for to start a coffeeshop business point at specific districts that are located between street of Tahlia south to north, Madinah rod east to west. Since there are lots of coffeeshops in the areas mentioned I will try to detect locations that are not already crowded with coffeeshops. I will also particularly be interested in areas with no or few coffeeshops in the neighborhood. I would also prefer locations as close to Salamah, Zahra districts circle as possible, assuming that first two conditions are met. I will use the data science powers to generate a few most promising neighborhoods based on this criterion. Advantages of each area will then be clearly expressed so that best possible final location can be selected.

Data

Based on the criteria of the business problem, the factors that will affect our decision are:

1. Number of existing **coffeeshops** in the neighborhood.
2. Number of and distance to **coffeeshops** in the neighborhood, and
3. Distance of neighborhood from Salamah, Zahra center

We decided to use regularly spaced circular grids of locations, centered around Salamah, Zahra Circle center, to define our neighborhoods.

Following data sources will be needed to extract/generate the required information:

1. Centers of candidate areas will be generated algorithmically and approximate addresses of centers of those areas will be obtained using Google Maps API reverse geocoding.
2. Number of **coffeeshops** and their type and location in every neighborhood will be obtained using Foursquare API.
3. Coordinate of center will be obtained using Google Maps API geocoding of well-known Salamah, Zahra districts in Jeddah city.

Neighborhood Candidates

We'll create latitude and longitude coordinates for centroids of our candidate neighborhoods. We will create a grid of cells covering our area of interest which is around 12x12 kilometers centered around Salamah and Zahra center.

First, we find the latitude and longitude of Salamah and Zahra Districts center, using specific, well known address in Google Maps geocoding API.

Importing required Libs

```
import pandas as pd
import numpy as np
import requests
#!conda install -c conda-forge shapely -y
import shapely.geometry
#!conda install -c conda-forge pyproj -y
import pyproj
import math
#!conda install -c conda-forge folium
import folium
import pickle
from folium import plugins
from folium.plugins import HeatMap
from sklearn.cluster import KMeans
import json
%matplotlib inline
```

Assigning Google API parameter

```
google_api_key='AIzaSyCAG1JdKY1RpxQt_RACOXaKxTjI-InMWrQ'
```

Finding the Coordinate of Salama, Jeddah, Saudi Arabia

```
def get_coordinates(api_key, address, verbose=False):
    try:
        url = 'https://maps.googleapis.com/maps/api/geocode/json?key={}&address={}'.format(api_key, address)
        response = requests.get(url).json()
        if verbose:
            print('Google Maps API JSON result =>', response)
        results = response['results']
        geographical_data = results[0]['geometry']['location'] # get geographical coordinates
        lat = geographical_data['lat']
        lon = geographical_data['lng']
        return [lat, lon]
    except:
        return [None, None]

address = 'Salama, Jeddah, Saudi Arabia'
jeddah_center = get_coordinates(google_api_key, address)
print('Coordinate of {}: {}'.format(address, jeddah_center))
```

```
Coordinate of Salama, Jeddah, Saudi Arabia: [21.5710199, 39.1360345]
```

We will create a grid of area candidates, equally spaced, centered around city center and within ~3km from Salama. Our neighborhoods will be defined as circular areas with a radius of 300 meters, so our neighborhood centers will be 600 meters apart. To accurately calculate distances, we need to create our grid of locations in Cartesian 2D coordinate system which allows us to calculate distances in meters (not in latitude/longitude degrees). Then we will project those coordinates back to latitude/longitude degrees to be shown on Folium map. Hence, will create functions to convert between WGS84 spherical coordinate system (latitude/longitude degrees) and UTM Cartesian coordinate system (X/Y coordinates in meters).

```
def lonlat_to_xy(lon, lat):
    proj_latlon = pyproj.Proj(proj='latlong',datum='WGS84')
    proj_xy = pyproj.Proj(proj="utm", zone=33, datum='WGS84')
    xy = pyproj.transform(proj_latlon, proj_xy, lon, lat)
    return xy[0], xy[1]

def xy_to_lonlat(x, y):
    proj_latlon = pyproj.Proj(proj='latlong',datum='WGS84')
    proj_xy = pyproj.Proj(proj="utm", zone=33, datum='WGS84')
    lonlat = pyproj.transform(proj_xy, proj_latlon, x, y)
    return lonlat[0], lonlat[1]

def calc_xy_distance(x1, y1, x2, y2):
    dx = x2 - x1
    dy = y2 - y1
    return math.sqrt(dx*dx + dy*dy)

print('Coordinate transformation check')
print('-------------------------------')
print('Jeddah center longitude={}, latitude={}'.format(jeddah_center[1], jeddah_center[0]))
x, y = lonlat_to_xy(jeddah_center[1], jeddah_center[0])
print('Jeddah center UTM X={}, Y={}'.format(x, y))
lo, la = xy_to_lonlat(x, y)
print('Jeddah center longitude={}, latitude={}'.format(lo, la))
```

```
Coordinate transformation check
-------------------------------
Jeddah center longitude=39.1360345, latitude=21.5710199
Jeddah center UTM X=3054136.389771756, Y=2591669.758443189
Jeddah center longitude=39.136034499999994, latitude=21.5710199
```

Creating a hexagonal grid of cells by offset every other row, and adjust vertical row spacing so that every cell center is equally distant from all its neighbors.

```
jeddah_center_x, jeddah_center_y = lonlat_to_xy(jeddah_center[1], jeddah_center[0]) # City center in Cartesian coordinates

k = math.sqrt(3) / 2 # Vertical offset for hexagonal grid cells
x_min = jeddah_center_x - 6000
x_step = 600
y_min = jeddah_center_y - 6000 - (int(21/k)*k*600 - 12000)/2
y_step = 600 * k

latitudes = []
longitudes = []
distances_from_center = []
xs = []
ys = []
for i in range(0, int(21/k)):
    y = y_min + i * y_step
    x_offset = 300 if i%2==0 else 0
    for j in range(0, 21):
        x = x_min + j * x_step + x_offset
        distance_from_center = calc_xy_distance(jeddah_center_x, jeddah_center_y, x, y)
        if (distance_from_center <= 6001):
            lon, lat = xy_to_lonlat(x, y)
            latitudes.append(lat)
            longitudes.append(lon)
            distances_from_center.append(distance_from_center)
            xs.append(x)
            ys.append(y)

print(len(latitudes), 'candidate neighborhood centers generated.')
```

```
364 candidate neighborhood centers generated.
```

Visualizing the data, we have so far, city center location and candidate neighborhood centers.

```
jeddah_map = folium.Map(location=jeddah_center, zoom_start=13)
folium.Marker(jeddah_center, popup='Alexanderplatz').add_to(jeddah_map)
for lat, lon in zip(latitudes, longitudes):
    folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_color='blue', fill_opacity=1).add_to(jeddah_map)
    #folium.Circle([lat, lon], radius=300, color='blue', fill=False).add_to(jeddah_map)
    #folium.Marker([lat, lon]).add_to(jeddah_map)
jeddah_map
```

We will use Google Maps API to get approximate addresses of those locations, as long as we have the coordinates of centers of neighborhoods/areas to be evaluated, equally spaced (distance from every point to its neighbors is exactly the same) and within ~3km from Salama.

```
def get_address(api_key, latitude, longitude, verbose=False):
    try:
        url = 'https://maps.googleapis.com/maps/api/geocode/json?key={}&latlng={},{}'.format(api_key, latitude, longitude)
        response = requests.get(url).json()
        if verbose:
            print('Google Maps API JSON result =>', response)
        results = response['results']
        address = results[0]['formatted_address']
        return address
    except:
        return None

addr = get_address(google_api_key, jeddah_center[0], jeddah_center[1])
print('Reverse geocoding check')
print('-----------------------')
print('Address of [{}, {}] is: {}'.format(jeddah_center[0], jeddah_center[1], addr))
```

```
Reverse geocoding check
-----------------------
Address of [21.5710199, 39.1360345] is: 3124 As Salam, Al Khalidiyah District, Jeddah 23423 8757, Saudi Arabia
```

```
print('Obtaining location addresses: ', end='')
addresses = []
for lat, lon in zip(latitudes, longitudes):
    address = get_address(google_api_key, lat, lon)
    if address is None:
        address = 'NO ADDRESS'
    address = address.replace(', Saudi Arabia', '') # We don't need country part of address
    addresses.append(address)
    print(' .', end='')
print(' done.')
```

```
Obtaining location addresses:  . . . . . .
```

```
addresses[150:170]
```

```
['Prince Mohammed Bin Abdulaziz St, Jeddah 23432',
 '8909 Al Mustashar, Al Aziziyah District, Jeddah 23334\xa02597',
 '3123 Abdullah Al Qasabi, Al Aziziyah District, Jeddah 23334\xa08782',
 'Saudi Arabia',
 'Al Kurnaysh Rd, Ash Shati, Jeddah 23415',
 '8854 Al Kurnaysh Rd, Ash Shati, Jeddah 23413',
 '8854 Al Kurnaysh Rd, Ash Shati, Jeddah 23413',
 '8854 Al Kurnaysh Rd, Ash Shati District, Jeddah 23413\xa02053',
 '8726, Ash Shati District, Jeddah 23414\xa02554',
 '3151 Ali Abu Al Ula, Ash Shati District, Jeddah 23414\xa08611',
 '8548 Mitab Ibn Awf, Ash Shati District, Jeddah 23414\xa03589',
 '8352 Zumrat As Salihin, Al Khalidiyah District, Jeddah 23422\xa02498',
 '8232 Ar Rawd Al Bahij, Al Khalidiyah District, Jeddah 23423\xa03030',
 '3575 Al Ikram, Al Khalidiyah District, Jeddah 23423\xa08153',
 '2319 Al Kayyal, AR Rawdah District, Jeddah 23433\xa07922',
 '7777 Abdul Malik Ibn Marwan, AR Rawdah District, Jeddah 23433\xa02902',
 '3431 Muhammad Iqbal, AR Rawdah District, Jeddah 23433\xa07660',
 '3989 Abdullah Balkhayr, AR Rawdah District, Jeddah 23434\xa07592',
 '4509 Abdullah Ibn Galawi, AR Rawdah District, Jeddah 23432\xa07500',
 'Al Khuraiji Building, Ar Rawdah, Jeddah 23432']
```

Using results to create Pandas Dataframe.

```
df_locations = pd.DataFrame({'Address': addresses,
                             'Latitude': latitudes,
                             'Longitude': longitudes,
                             'X': xs,
                             'Y': ys,
                             'Distance from center': distances_from_center})

df_locations.head(10)
```

	Address	Latitude	Longitude	X	Y	Distance from center
0	Unnamed Road, Al Andalus, Jeddah 23311	21.526354	39.111886	3.052336e+06	2.585954e+06	5992.495307
1	Unnamed Road, Al Andalus, Jeddah Saudi Arabia	21.525541	39.117171	3.052936e+06	2.585954e+06	5840.376700
2	Unnamed Road, Al Andalus, Jeddah 23311	21.524727	39.122457	3.053536e+06	2.585954e+06	5747.173218
3	Unnamed Road, Al Andalus, Jeddah 23311	21.523914	39.127742	3.054136e+06	2.585954e+06	5715.767665
4	Unnamed Road, Al Andalus, Jeddah 23311	21.523101	39.133026	3.054736e+06	2.585954e+06	5747.173218
5	Unnamed Road, Al Andalus, Jeddah Saudi Arabia	21.522287	39.138311	3.055336e+06	2.585954e+06	5840.376700
6	Unnamed Road, Al Andalus, Jeddah Saudi Arabia	21.521473	39.143595	3.055936e+06	2.585954e+06	5992.495307
7	Unnamed Road, Al Andalus, Jeddah 23311	21.531856	39.104710	3.051436e+06	2.586474e+06	5855.766389
8	Unnamed Road, Al Andalus, Jeddah 23311	21.531043	39.109996	3.052036e+06	2.586474e+06	5604.462508
9	Unnamed Road, Al Andalus, Jeddah 23311	21.530230	39.115281	3.052636e+06	2.586474e+06	5408.326913

Save this data into local file.

```
df_locations.to_pickle('./locations.pkl')
```

Foursquare

We have our location candidates; we will use Foursquare API to get information on coffeeshops in each neighborhood. We are interested in 'specialized coffee' category in the venues, and only the proper coffeeshop. Therefore, we will include in our list only venues that have 'coffeeshop' in the category name, and we will make sure to detect and include all the subcategories of specific 'specialized coffee' category, as we need info on coffeeshop in the neighborhood.

```
# The code was removed by Watson Studio for sharing.
foursquare_client_id='KKKSFAGH5Z1BU1HW5TKRGJ4RSRLPL0BFAPPZ4OPMBS3R5RCM'
foursquare_client_secret='G4TKQFJGCJSHBSLCHHKTBUJMLQHOMMQXF11MBGVSXJIEZIBX'
```

```
# Category IDs corresponding to coffeeshop were taken from Foursquare web site (https://developer.foursquare.com/docs/resources/categories):

food_category = '4bf58dd8d48988d1e0931735' # 'Root' category for all food-related venues

#coffee_shop_categories = ['4bf58dd8d48988d1e0931735']

def is_coffeeshop(categories, specific_filter=None):
    coffeeshop_words = ['coffee shop', 'coffeeshop', 'coffee', 'specialized coffeeshop', 'specialized coffee']
    coffeeshop = False
    specific = False
    for c in categories:
        category_name = c[0].lower()
        category_id = c[1]
        for r in coffeeshop_words:
            if r in category_name:
                coffeeshop = True
        if 'fast food' in category_name:
            coffeeshop = False
        if not(specific_filter is None) and (category_id in specific_filter):
            specific = True
            coffeeshop = True
    return coffeeshop, specific

def get_categories(categories):
    return [(cat['name'], cat['id']) for cat in categories]

def format_address(location):
    address = ', '.join(location['formattedAddress'])
    #address = address.replace(', Deutschland', '')
    address = address.replace(', Saudi Arabia', '')
    return address

def get_venues_near_location(lat, lon, category, client_id, client_secret, radius=500, limit=100):
    version = '20180724'
    url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&v={}&ll={},{}&categoryId={}&radius={}&limit={}'.format(
        client_id, client_secret, version, lat, lon, category, radius, limit)
    try:
        results = requests.get(url).json()['response']['groups'][0]['items']
        venues = [(item['venue']['id'],
                   item['venue']['name'],
                   get_categories(item['venue']['categories']),
                   (item['venue']['location']['lat'], item['venue']['location']['lng']),
                   format_address(item['venue']['location']),
                   item['venue']['location']['distance']) for item in results]
    except:
        venues = []
    return venues
```

```
# Let's now go over our neighborhood locations and get nearby coffeeshops; we'll also maintain a dictionary of all found co
ffeeshops and all found coffeeshops

def get_coffeeshops(lats, lons):
    coffeeshops = {}
    coffee_shops = {}
    location_coffeeshops = []

    print('Obtaining venues around candidate locations:', end='')
    for lat, lon in zip(lats, lons):
        # Using radius=350 to meke sure we have overlaps/full coverage so we don't miss any coffeeshop (we're using diction
aries to remove any duplicates resulting from area overlaps)
        venues = get_venues_near_location(lat, lon, food_category, foursquare_client_id, foursquare_client_secret, radius=3
50, limit=100)
        area_coffeeshops = []
        for venue in venues:
            venue_id = venue[0]
            venue_name = venue[1]
            venue_categories = venue[2]
            venue_latlon = venue[3]
            venue_address = venue[4]
            venue_distance = venue[5]
            #is_res, is_italian = is_coffeeshop(venue_categories, specific_filter=coffeeshop_categories)
            #if is_res:
            x, y = lonlat_to_xy(venue_latlon[1], venue_latlon[0])
            coffeeshop = (venue_id, venue_name, venue_latlon[0], venue_latlon[1], venue_address, venue_distance, is_italia
n, x, y)
            if venue_distance<=300:
                    area_coffeeshops.append(coffeeshop)
            coffeeshops[venue_id] = coffeeshop
            if is_italian:
                    coffee_shops[venue_id] = coffeeshop
        location_coffeeshops.append(area_coffeeshops)
        print(' .', end='')
    print(' done.')
    return coffeeshops, coffee_shops, location_coffeeshops

# Try to load from local file system in case we did this before
coffeeshops = {}
coffee_shops = {}
location_coffeeshops = []
loaded = False
try:
    with open('coffeeshops_350.pkl', 'rb') as f:
        coffeeshops = pickle.load(f)
    with open('coffee_shops_350.pkl', 'rb') as f:
        coffee_shops = pickle.load(f)
    with open('location_coffeeshops_350.pkl', 'rb') as f:
        location_coffeeshops = pickle.load(f)
    print('Coffeeshop data loaded.')
    loaded = True
except:
    pass

# If load failed use the Foursquare API to get the data
if not loaded:
    coffeeshops, coffee_shops, location_coffeeshops = get_coffeeshops(latitudes, longitudes)

    # Let's persists this in local file system
    with open('coffeeshops_350.pkl', 'wb') as f:
        pickle.dump(coffeeshops, f)
    with open('coffee_shops_350.pkl', 'wb') as f:
        pickle.dump(coffee_shops, f)
    with open('location_coffeeshops_350.pkl', 'wb') as f:
        pickle.dump(location_coffeeshops, f)
```

```
Coffeeshop data loaded.
```

```
print('Total number of Coffeeshops:', len(coffeeshops))
print('Total number of Specialized Coffeeshops:', len(coffee_shops))
print('Percentage of Coffeeshops: {:.2f}%'.format(len(coffee_shops) / len(coffeeshops) * 100))
print('Average number of Coffeeshops in neighborhood:', np.array([len(r) for r in location_coffeeshops]).mean())
```

```
Total number of Coffeeshops: 662
Total number of Specialized Coffeeshops: 0
Percentage of Coffeeshops: 0.00%
Average number of Coffeeshops in neighborhood: 1.804945054945055
```

```
print('List of all coffeeshops')
print('-----------------------')
for r in list(coffeeshops.values())[:10]:
    print(r)
print('...')
print('Total:', len(coffeeshops))
```

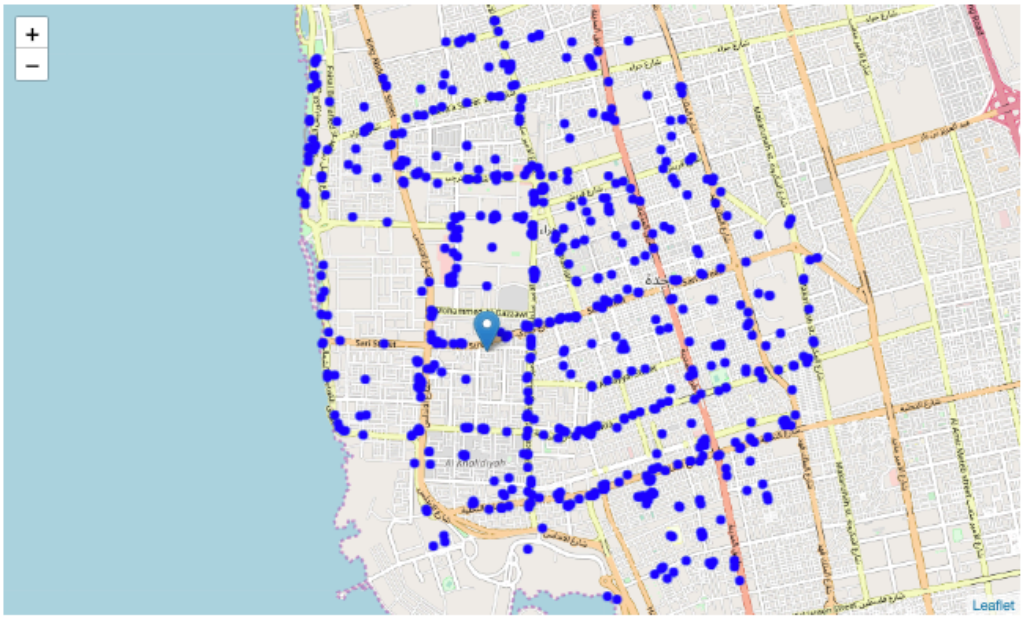
```
List of all coffeeshops
-----------------------
('56501612498e73a009f11ead', 'BARISTA Café', 21.526951366522898, 39.153394786224474, '286 ,المملكة العربية السعودية', F
alse, 3056913.1786803952, 2586779.6346825054)
('5b9aecec5455b20039d6376f', 'Moroccan Taste Cafe (المذاق المغربي)', 39.151071 ,21.524856, 'Corniche Road, المملكة ,بدة
129 ,العربية السعودية', False, 3056697.0518322648, 2586489.7500894764)
('5c9ef96b57a537002c0b7e8b', '133 ,المملكة العربية السعودية', جدة ,39.156437 ,21.529373 ,جده الكورنيش', False, 305720
2.3676790097, 2587121.164466249)
('54d956da498e7176d24e57ea', '# = ميدان الأمير عبدالمجيد ,'حي الحمرا ,39.15955205673164 ,21.528887928898236 ,عرة_وقهوة',
204 ,المملكة العربية السعودية', جدة 23321, مقابل جامع العناني (شارع الأندلس = طريق الكورنيش), False, 3057556.1903226143,
2587120.592192725)
('5d0d537f3e8ac400239cdf89', 'شارع الأندلس, جدة 23321, المملكة العربية السعودية', 39.162035, 21.526393 ,المذاق المغربي',
174 ,'ة, False, 3057879.2705842615, 2586871.0283656185)
('5c79af5b780eee002cb215b2', 'La Sera Lounge', 21.527589, 39.160174, '80 ,المملكة العربية السعودية', جدة ,23321, False,
3057650.2383445287, 2586978.4440595144)
('51a185f3498e1d8e280e11a6', 'Mochachino', 21.5265015076418, 39.162010733577844, 'Alandlus Road, ا المملكة العربية ,جدة
163 ,للسعودية, False, 3057874.4751045313, 2586883.407450842)
('515b2f87e4b068d5667491e3', 'Divede جانب عالم .. المعشي .. شارع فلسطين' ,39.16245775548089 ,21.52638049973463 ,يفيدي'
206 ,العصائر و ميد, المملكة العربية السعودية', False, 3057926.253163936, 2586877.2529285126)
('5ce5aa73a22db70039e03e9c', 'Overdose', 21.533925, 39.15595, '242 ,المملكة العربية السعودية', False, 3057059.911006369
6, 2587650.130360235)
('5d2251e7d29cbb0023016349', '202 ,المملكة العربية السعودية', جدة 23321, 39.157495 ,21.533371 ,واجهة جده البحرية', Fal
se, 3057241.4939057166, 2587612.8183931345)
...
Total: 662
```

```
print('Coffeeshops around location')
print('--------------------------')
for i in range(100, 110):
    rs = location_coffeeshops[i][:8]
    names = ', '.join([r[1] for r in rs])
    print('Coffeeshops around location {}: {}'.format(i+1, names))
```

```
Coffeeshops around location
--------------------------
Coffeeshops around location 101:
Coffeeshops around location 102: Oras Cafe, Mochachino
Coffeeshops around location 103:
Coffeeshops around location 104: STARCUP Coffee, PRESSO, Bon Cafe, @Work Coffee Shop
Coffeeshops around location 105: Castana Cafe, Best Friends cafè, Chance Coffee, marakesh cafe, saudia city
Coffeeshops around location 106: Alamad Cafe
Coffeeshops around location 107: اوفر دوز, Glow Lounge Restaurant, West Comp Coffeeshop, Mirage Cafe
Coffeeshops around location 108: Al Aqeeq Tea Lounge (العقيق تي لاونج), Glow Lounge Restaurant, Al Aqeeq
Coffeeshops around location 109: illy Caffè, Nespresso (نيسبريسو), Square Restaurant & Café, Fauchon (فوشون)
Coffeeshops around location 110: The Coffee House (دارة القهوة), Cafe Aziz, Casper & Gambini's (كاسبر اند جامبينيز), Al
Forn (الفرن), Al-Montazah Cafe' (كافيه المنتزه), La Promenade Coffee, Crepe Cafe
```

Let's now see all the collected coffeeshops in our area of interest on map, and let's also show coffeeshops in different color.

```
jeddah_map = folium.Map(location=jeddah_center, zoom_start=13)
folium.Marker(jeddah_center, popup='Salamah').add_to(jeddah_map)
for res in coffeeshops.values():
    lat = res[2]; lon = res[3]
    is_coffeeshop = res[6]
    color = 'red' if is_coffeeshop else 'blue'
    folium.CircleMarker([lat, lon], radius=3, color=color, fill=True, fill_color=color, fill_opacity=1).add_to(jeddah_map)
jeddah_map
```



Now we have all the coffeeshops in area within few kilometers from Salamah, and we know which ones are coffeeshops! We also know which coffeeshops exactly are in vicinity of every neighborhood candidate center. This concludes the data gathering phase - we're now ready to use this data for analysis to produce the report on optimal locations for a new coffeeshop.

Methodology

In this project we will direct our efforts on detecting areas in Jeddah that have low coffeeshop density, particularly those with low number of coffeeshops. We will limit our analysis to area ~6km around city center.

1. First step we have collected the required data location and type (category) of every coffeeshop within 6km from Jeddah center, and identified coffeeshops (according to Foursquare categorization).
2. Second step in our analysis will be calculation and exploration of 'coffeeshop density' across different areas of Jeddah - we will use heatmaps to identify a few promising areas close to center with low number of coffeeshops in general (coffeeshops in vicinity) and focus our attention on those areas.
3. In third and final step we will focus on most promising areas and within those create clusters of locations that meet some basic requirements established in discussion with stakeholders:

we have to consider locations with no more than two coffeeshops in radius of 250 meters, and we want locations without coffeeshops in radius of 400 meters. We will present map of all such locations but also create clusters (using k-means clustering) of those locations to identify general zones / neighborhoods / addresses which should be a starting point for final 'street level' exploration and search for optimal venue location by stakeholders.

Analysis

Performing basic explanatory data analysis and derive some additional info from our raw data, and count the number of coffeeshops in every area candidate

```
location_coffeeshops_count = [len(res) for res in location_coffeeshops]

df_locations['Coffeeshops in area'] = location_coffeeshops_count

print('Average number of coffeeshops in every area with radius=300m:', np.array(location_coffeeshops_count).mean())

df_locations.head(10)
```

```
Average number of coffeeshops in every area with radius=300m: 1.804945054945055
```

	Address	Latitude	Longitude	X	Y	Distance from center	Coffeeshops in area	Distance to coffeeshop
0	Unnamed Road, Al Andalus, Jeddah 23311	21.526354	39.111886	3.052336e+06	2.585954e+06	5992.495307	0	10000
1	Unnamed Road, Al Andalus, Jeddah Saudi Arabia	21.525541	39.117171	3.052936e+06	2.585954e+06	5840.376700	0	10000
2	Unnamed Road, Al Andalus, Jeddah 23311	21.524727	39.122457	3.053536e+06	2.585954e+06	5747.173218	0	10000
3	Unnamed Road, Al Andalus, Jeddah 23311	21.523914	39.127742	3.054136e+06	2.585954e+06	5715.767665	0	10000
4	Unnamed Road, Al Andalus, Jeddah 23311	21.523101	39.133026	3.054736e+06	2.585954e+06	5747.173218	0	10000
5	Unnamed Road, Al Andalus, Jeddah Saudi Arabia	21.522287	39.138311	3.055336e+06	2.585954e+06	5840.376700	0	10000
6	Unnamed Road, Al Andalus, Jeddah Saudi Arabia	21.521473	39.143595	3.055936e+06	2.585954e+06	5992.495307	0	10000
7	Unnamed Road, Al Andalus, Jeddah 23311	21.531856	39.104710	3.051436e+06	2.586474e+06	5855.766389	0	10000
8	Unnamed Road, Al Andalus, Jeddah 23311	21.531043	39.109996	3.052036e+06	2.586474e+06	5604.462508	0	10000
9	Unnamed Road, Al Andalus, Jeddah 23311	21.530230	39.115281	3.052636e+06	2.586474e+06	5408.326913	0	10000

Calculating the distance to nearest Italian coffeeshop from every area candidate center, including those within 300m, and distance to closest one, regardless of how distant it is.

```
distances_to_coffeeshop = []

for area_x, area_y in zip(xs, ys):
    min_distance = 10000
    for res in coffee_shops.values():
        res_x = res[5]
        res_y = res[6]
        d = calc_xy_distance(area_x, area_y, res_x, res_y)
        if d<min_distance:
            min_distance = d
    distances_to_coffeeshop.append(min_distance)

df_locations['Distance to coffeeshop'] = distances_to_coffeeshop
df_locations.head(10)
```

	Address	Latitude	Longitude	X	Y	Distance from center	Coffeeshops in area	Distance to coffeeshop
0	Unnamed Road, Al Andalus, Jeddah 23311	21.526354	39.111886	3.052336e+06	2.585954e+06	5992.495307	0	10000
1	Unnamed Road, Al Andalus, Jeddah Saudi Arabia	21.525541	39.117171	3.052936e+06	2.585954e+06	5840.376700	0	10000
2	Unnamed Road, Al Andalus, Jeddah 23311	21.524727	39.122457	3.053536e+06	2.585954e+06	5747.173218	0	10000
3	Unnamed Road, Al Andalus, Jeddah 23311	21.523914	39.127742	3.054136e+06	2.585954e+06	5715.767665	0	10000
4	Unnamed Road, Al Andalus, Jeddah 23311	21.523101	39.133026	3.054736e+06	2.585954e+06	5747.173218	0	10000
5	Unnamed Road, Al Andalus, Jeddah Saudi Arabia	21.522287	39.138311	3.055336e+06	2.585954e+06	5840.376700	0	10000
6	Unnamed Road, Al Andalus, Jeddah Saudi Arabia	21.521473	39.143595	3.055936e+06	2.585954e+06	5992.495307	0	10000
7	Unnamed Road, Al Andalus, Jeddah 23311	21.531856	39.104710	3.051436e+06	2.586474e+06	5855.766389	0	10000
8	Unnamed Road, Al Andalus, Jeddah 23311	21.531043	39.109996	3.052036e+06	2.586474e+06	5604.462508	0	10000
9	Unnamed Road, Al Andalus, Jeddah 23311	21.530230	39.115281	3.052636e+06	2.586474e+06	5408.326913	0	10000

```
print('Average distance to closest coffeeshop from each area center:', df_locations['Distance to coffeeshop'].mean())
```

```
Average distance to closest coffeeshop from each area center: 1000.0
```

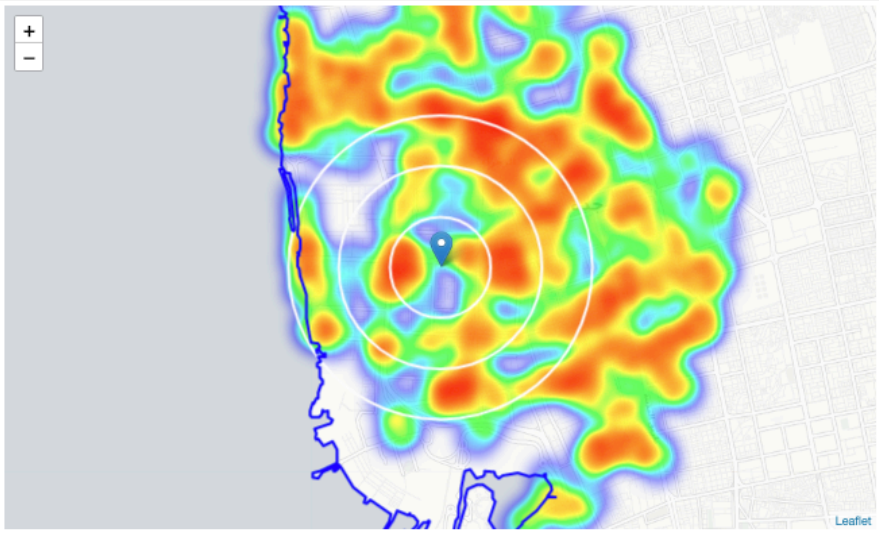
On average coffeeshop can be found within ~500m from every candidate area center. That's fairly close, therefore; we need to filter our areas carefully.

Creating a map showing heatmap / density of coffeeshops and try to extract some meaningful info from that, and showing borders of Jeddah boroughs on our map and a few circles indicating distance of 1km, 2km and 3km from Salamah.

```
with open('Makkah.geojson','r') as f:
    jeddah_boroughs = json.load(f)

def boroughs_style(feature):
    return { 'color': 'blue', 'fill': False }
```

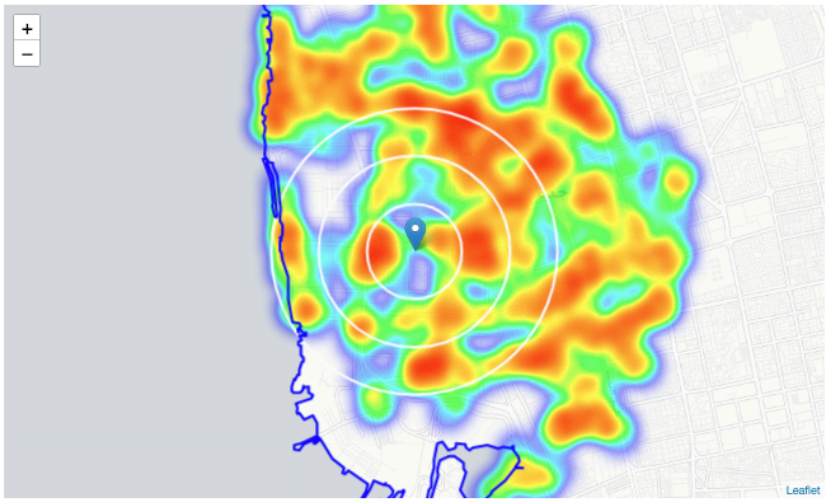
```
coffeeshop_latlons = [[res[2], res[3]] for res in coffeeshops.values()]

coffee_shops_latlons = [[res[2], res[3]] for res in coffee_shops.values()]
```

```
jeddah_map = folium.Map(location=jeddah_center, zoom_start=13)
folium.TileLayer('cartodbpositron').add_to(jeddah_map) #cartodbpositron cartodbdark_matter
HeatMap(coffeeshop_latlons).add_to(jeddah_map)
folium.Marker(jeddah_center).add_to(jeddah_map)
folium.Circle(jeddah_center, radius=1000, fill=False, color='white').add_to(jeddah_map)
folium.Circle(jeddah_center, radius=2000, fill=False, color='white').add_to(jeddah_map)
folium.Circle(jeddah_center, radius=3000, fill=False, color='white').add_to(jeddah_map)
folium.GeoJson(jeddah_boroughs, style_function=boroughs_style, name='geojson').add_to(jeddah_map)
jeddah_map
```



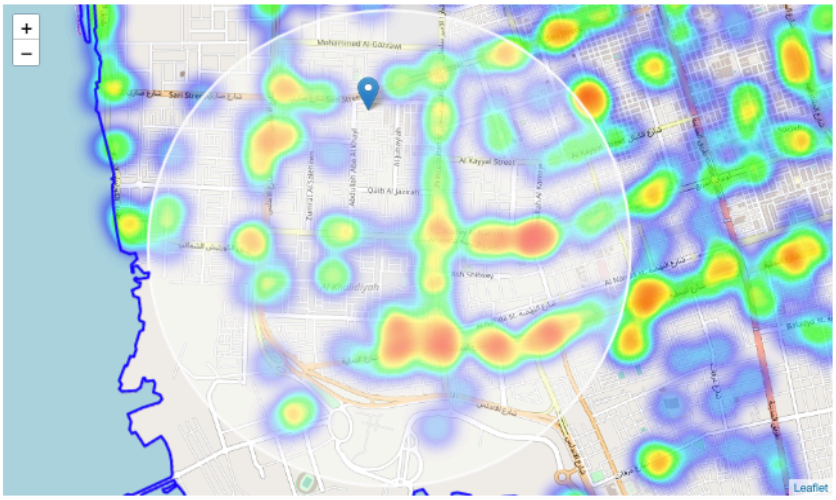
Looks like a few pockets of low coffeeshop density closest to city center can be found south, south-east and east from Salama. Will create another heatmap map showing heatmap/density of coffeeshops only.

```
jeddah_map = folium.Map(location=jeddah_center, zoom_start=13)
folium.TileLayer('cartodbpositron').add_to(jeddah_map) #cartodbpositron cartodbdark_matter
HeatMap(coffeeshop_latlons).add_to(jeddah_map)
folium.Marker(jeddah_center).add_to(jeddah_map)
folium.Circle(jeddah_center, radius=1000, fill=False, color='white').add_to(jeddah_map)
folium.Circle(jeddah_center, radius=2000, fill=False, color='white').add_to(jeddah_map)
folium.Circle(jeddah_center, radius=3000, fill=False, color='white').add_to(jeddah_map)
folium.GeoJson(jeddah_boroughs, style_function=boroughs_style, name='geojson').add_to(jeddah_map)
jeddah_map
```



Define new narrower region of interest, which will include low-coffeeshop-count parts of Salamah.

```
roi_x_min = jeddah_center_x - 2000
roi_y_max = jeddah_center_y + 1000
roi_width = 5000
roi_height = 5000
roi_center_x = roi_x_min + 2500
roi_center_y = roi_y_max - 2500
roi_center_lon, roi_center_lat = xy_to_lonlat(roi_center_x, roi_center_y)
roi_center = [roi_center_lat, roi_center_lon]

jedddah_map = folium.Map(location=roi_center, zoom_start=14)
HeatMap(coffeeshop_latlons).add_to(jedddah_map)
folium.Marker(jeddah_center).add_to(jedddah_map)
folium.Circle(roi_center, radius=2500, color='white', fill=True, fill_opacity=0.4).add_to(jedddah_map)
folium.GeoJson(jeddah_boroughs, style_function=boroughs_style, name='geojson').add_to(jedddah_map)
jedddah_map
```



This is covering all the pockets of low coffeeshop density in Salamah and Rawdah closest to Jeddah center. Therefore; will create new, denser grid of location candidates restricted to our new region of interest (let's make our location candidates 100m apart).

```
k = math.sqrt(3) / 2 # Vertical offset for hexagonal grid cells
x_step = 100
y_step = 100 * k
roi_y_min = roi_center_y - 2500

roi_latitudes = []
roi_longitudes = []
roi_xs = []
roi_ys = []
for i in range(0, int(51/k)):
    y = roi_y_min + i * y_step
    x_offset = 50 if i%2==0 else 0
    for j in range(0, 51):
        x = roi_x_min + j * x_step + x_offset
        d = calc_xy_distance(roi_center_x, roi_center_y, x, y)
        if (d <= 2501):
            lon, lat = xy_to_lonlat(x, y)
            roi_latitudes.append(lat)
            roi_longitudes.append(lon)
            roi_xs.append(x)
            roi_ys.append(y)

print(len(roi_latitudes), 'candidate neighborhood centers generated.')
```

```
2261 candidate neighborhood centers generated.
```

Calculating two most important things for each location candidate: number of coffeeshops in vicinity (we'll use radius of 250 meters) and distance to closest coffeeshop.

```
def count_coffeeshops_nearby(x, y, coffeeshops, radius=250):
    count = 0
    for res in coffeeshops.values():
        res_x = res[7]; res_y = res[8]
        d = calc_xy_distance(x, y, res_x, res_y)
        if d<=radius:
            count += 1
    return count

def find_nearest_coffeeshop(x, y, coffeeshops):
    d_min = 100000
    for res in coffeeshops.values():
        res_x = res[7]; res_y = res[8]
        d = calc_xy_distance(x, y, res_x, res_y)
        if d<=d_min:
            d_min = d
    return d_min

roi_coffeeshop_counts = []
roi_coffeeshop_distances = []

print('Generating data on location candidates... ', end='')
for x, y in zip(roi_xs, roi_ys):
    count = count_coffeeshops_nearby(x, y, coffeeshops, radius=250)
    roi_coffeeshop_counts.append(count)
    distance = find_nearest_coffeeshop(x, y, coffee_shops)
    roi_coffeeshop_distances.append(distance)
print('done.')
```

```
Generating data on location candidates... done.
```

```
# Let's put this into dataframe
df_roi_locations = pd.DataFrame({'Latitude':roi_latitudes,
                                 'Longitude':roi_longitudes,
                                 'X':roi_xs,
                                 'Y':roi_ys,
                                 'Coffeeshops nearby':roi_coffeeshop_counts,
                                 'Distance to coffeeshop':roi_coffeeshop_distances})

df_roi_locations.head(10)
```

	Latitude	Longitude	X	Y	Coffeeshops nearby	Distance to coffeeshop
0	21.537444	39.134193	3.054586e+06	2.587670e+06	0	100000
1	21.537308	39.135073	3.054686e+06	2.587670e+06	0	100000
2	21.538904	39.129473	3.054036e+06	2.587756e+06	0	100000
3	21.538768	39.130354	3.054136e+06	2.587756e+06	0	100000
4	21.538633	39.131235	3.054236e+06	2.587756e+06	0	100000
5	21.538497	39.132116	3.054336e+06	2.587756e+06	0	100000
6	21.538361	39.132997	3.054436e+06	2.587756e+06	0	100000
7	21.538226	39.133878	3.054536e+06	2.587756e+06	0	100000
8	21.538090	39.134759	3.054636e+06	2.587756e+06	0	100000
9	21.537954	39.135639	3.054736e+06	2.587756e+06	0	100000

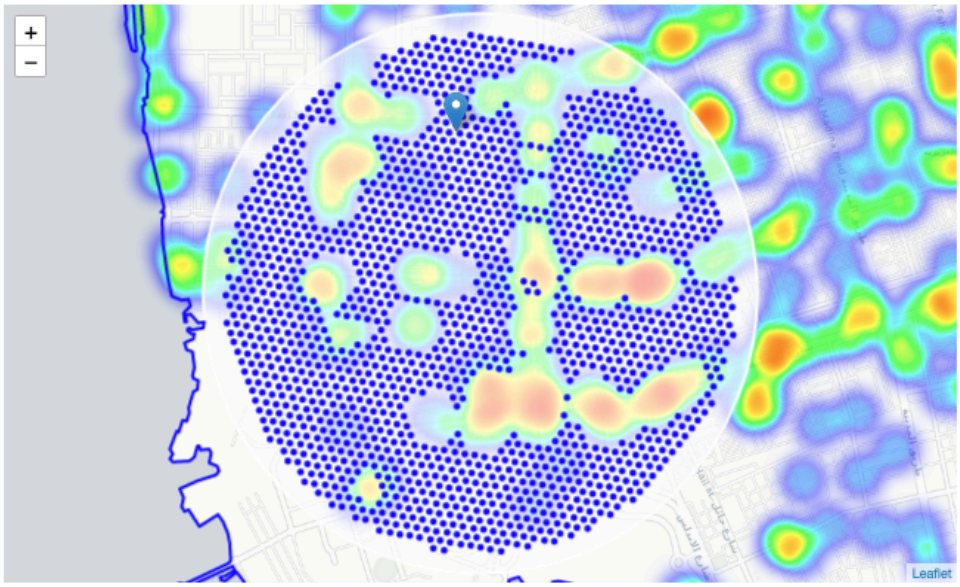
We will filter those locations: we were interested only in locations with no more than two coffeeshops in radius of 250 meters, and number of coffeeshops in radius of 400 meters.

```
good_res_count = np.array((df_roi_locations['Coffeeshops nearby']<=2))
print('Locations with no more than two coffeeshops nearby:', good_res_count.sum())

good_coffeeshop_distance = np.array(df_roi_locations['Distance to coffeeshop']>=400)
print('Locations with no coffeeshops within 400m:', good_coffeeshop_distance.sum())

good_locations = np.logical_and(good_res_count, good_coffeeshop_distance)
print('Locations with both conditions met:', good_locations.sum())

df_good_locations = df_roi_locations[good_locations]
```

```
Locations with no more than two coffeeshops nearby: 1610
Locations with no coffeeshops within 400m: 2261
Locations with both conditions met: 1610
```

Let's see how this looks on a map.

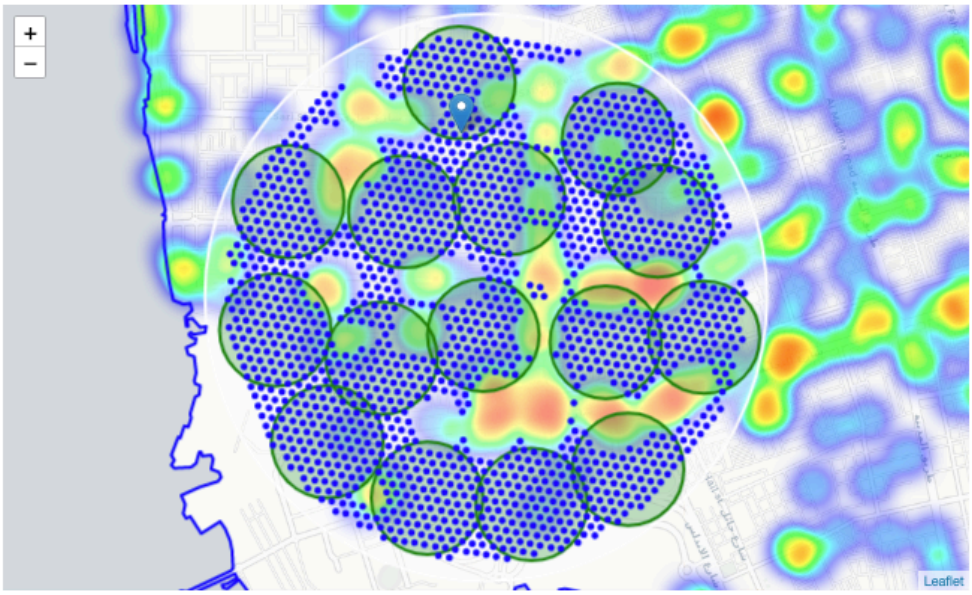
```
good_latitudes = df_good_locations['Latitude'].values
good_longitudes = df_good_locations['Longitude'].values

good_locations = [[lat, lon] for lat, lon in zip(good_latitudes, good_longitudes)]

jeddah_map = folium.Map(location=roi_center, zoom_start=14)
folium.TileLayer('cartodbpositron').add_to(jeddah_map)
HeatMap(coffeeshop_latlons).add_to(jeddah_map)
folium.Circle(roi_center, radius=2500, color='white', fill=True, fill_opacity=0.6).add_to(jeddah_map)
folium.Marker(jeddah_center).add_to(jeddah_map)
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_color='blue', fill_opacity=1).add_to(jeddah_ma
p)
folium.GeoJson(jeddah_boroughs, style_function=boroughs_style, name='geojson').add_to(jeddah_map)
jeddah_map
```



Now we have a bunch of locations fairly close to Salamah, and we know that each of those locations has no more than two coffeeshops in radius of 250m, and no coffeeshop closer than 400m. Any of those locations is a potential candidate for a new coffeeshop, at least based on nearby competition.

What we have now is a clear indication of zones with low number of coffeeshops in vicinity, and no coffeeshops at all nearby. Now we cluster locations to create centers of zones containing good locations. Those zones, their centers and addresses will be the final result of our analysis.

```
number_of_clusters = 15

good_xys = df_good_locations[['X', 'Y']].values
kmeans = KMeans(n_clusters=number_of_clusters, random_state=0).fit(good_xys)

cluster_centers = [xy_to_lonlat(cc[0], cc[1]) for cc in kmeans.cluster_centers_]

jeddah_map = folium.Map(location=roi_center, zoom_start=14)
folium.TileLayer('cartodbpositron').add_to(jeddah_map)
HeatMap(coffeeshop_latlons).add_to(jeddah_map)
folium.Circle(roi_center, radius=2500, color='white', fill=True, fill_opacity=0.4).add_to(jeddah_map)
folium.Marker(jeddah_center).add_to(jeddah_map)
for lon, lat in cluster_centers:
    folium.Circle([lat, lon], radius=500, color='green', fill=True, fill_opacity=0.25).add_to(jeddah_map)
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_color='blue', fill_opacity=1).add_to(jeddah_ma
p)
folium.GeoJson(jeddah_boroughs, style_function=boroughs_style, name='geojson').add_to(jeddah_map)
jeddah_map
```



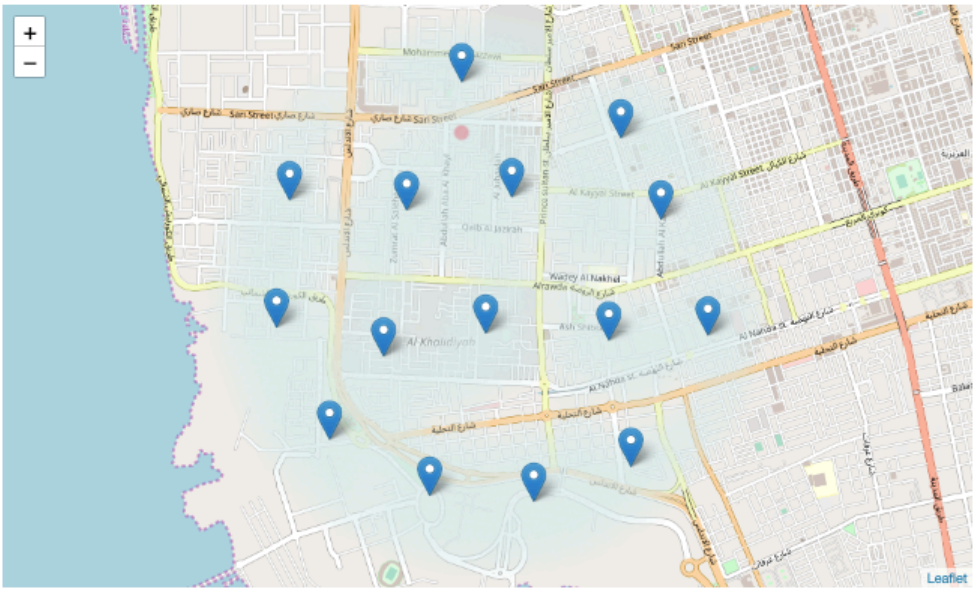
Finally, reverse geocode those candidate area centers to get the addresses which can be presented to stakeholders.

```
candidate_area_addresses = []
print('============================================================')
print('Addresses of centers of areas recommended for further analysis')
print('============================================================\n')
for lon, lat in cluster_centers:
    addr = get_address(google_api_key, lat, lon).replace(', Saudi Arabia', '')
    candidate_area_addresses.append(addr)
    x, y = lonlat_to_xy(lon, lat)
    d = calc_xy_distance(x, y, jeddah_center_x, jeddah_center_y)
    print('{}{} => {:.1f}km from Salamah'.format(addr, ' '*(50-len(addr)), d/1000))
```

```
============================================================
Addresses of centers of areas recommended for further analysis
============================================================

8040 Ma Al Ward, Al Khalidiyah District, Jeddah 23422 2635 => 0.9km from Salamah
4219 Muhammed Ibn Abdul Jabar, Al Andalus District, Jeddah 23322 6279 => 3.6km from Salamah
7715 Ibn Zaydun, AR Rawdah District, Jeddah 23433 3094 => 2.1km from Salamah
3658, Ash Shati District, Jeddah 23411 6130         => 3.2km from Salamah
3229 Muhammad Ali Balkayr, AZ Zahra District, Jeddah 23424 6301 => 0.5km from Salamah
AlKhaludiah. Al Khalidiyyah, Jeddah 23421           => 1.9km from Salamah
3203 Ahmad Al Qaisi, Ash Shati District, Jeddah 23411 7084 => 2.6km from Salamah
3575 Al Ikram, Al Khalidiyah District, Jeddah 23423 8153 => 0.8km from Salamah
3303 Ibn Surur, Ash Shati District, Jeddah 23414 8233 => 1.8km from Salamah
Al Kurnaysh Rd, Al Andalus, Jeddah 23311            => 3.5km from Salamah
8457 Al Maliki, AR Rawdah District, Jeddah 23435 2737 => 1.5km from Salamah
6709 Nahdat An Nasr, AR Rawdah District, Jeddah 23432 3538 => 3.0km from Salamah
6625 AlKhaludiah, Al Khalidiyah District, Jeddah 23421 2468 => 2.3km from Salamah
6609 Al Abud, AR Rawdah District, Jeddah 23431 2627 => 2.4km from Salamah
Unnamed Road, Al Andalus, Jeddah 23311              => 3.6km from Salamah
```

This concludes our analysis. We have created 15 addresses representing centers of zones containing locations with low number of coffeeshops and no coffeeshops nearby, all zones being fairly close to city center (all less than 4km from Salamah, and about half of those less than 2km from Salamah). Although zones are shown on map with a radius of ~500 meters (green circles), their shape is actually very irregular, and their centers/addresses should be considered only as a starting point for exploring area neighborhoods in search for potential coffeeshop locations. Most of the zones are located in Salamah, Rawdah, and Khaldia boroughs, which we have identified as interesting due to being popular with customers.

```
jeddah_map = folium.Map(location=roi_center, zoom_start=14)
folium.Circle(jeddah_center, radius=50, color='red', fill=True, fill_color='red', fill_opacity=1).add_to(jeddah_map)
for lonlat, addr in zip(cluster_centers, candidate_area_addresses):
    folium.Marker([lonlat[1], lonlat[0]], popup=addr).add_to(jeddah_map)
for lat, lon in zip(good_latitudes, good_longitudes):
    folium.Circle([lat, lon], radius=250, color='#0000ff00', fill=True, fill_color='#daf2e4', fill_opacity=0.05).add_to(jed
dah_map)
jeddah_map
```



Results and Discussion

Our analysis shows that although there is a great number of coffeeshops in Jeddah (~1000 in our initial area of interest which was 12x12km around Salamah), there are pockets of low coffeeshop density fairly close to city center. Highest concentration of coffeeshops was detected south and west from Salamah, so we focused our attention to areas south, south-east and east, corresponding to boroughs Salamah, Khaldiah. Another borough was identified as potentially interesting is Rawdah, but our attention was focused on Salamah and Rawdah which offer a combination of popularity among young customers, closeness to city center, strong socio-economic dynamics and a number of pockets of low coffeeshop density. After directing our attention to this narrower area of interest (covering approx. 5x5km south-east from Salamah) we first created a dense grid of location candidates (spaced 100m apart); those locations were then filtered so that those with more than two coffeeshops in radius of 250m and those with a coffeeshop closer than 400m were removed. The candidate's locations were then clustered to create zones of interest which contain greatest number of location candidates. Addresses of centers of those zones were also generated using reverse geocoding to be used as markers/starting points for more detailed local analysis based on other factors. Result of all this is 15 zones containing largest number of potential new coffeeshop locations based on number of and distance to existing venues, both coffeeshops in general and Specialized coffeeshops particularly. However; this is, does not imply that those zones are actually optimal locations for a new coffeeshop. The purpose of this analysis was to only provide info on areas close to Jeddah center but not crowded with existing coffeeshops, the Specialized in particular, it is entirely possible that there is a very good reason for small number of coffeeshops in any of those areas, reasons which would make them unsuitable for a new coffeeshop regardless of lack of competition in the area. Recommended zones should therefore be considered only as a starting point for more detailed analysis which could eventually result in location which has not only no nearby competition, but also other factors taken into account and all other relevant conditions met.

Conclusion

Purpose of this project was to identify Salamah areas close to center with low number of coffeeshops (particularly specialized coffeeshops) in order to aid stakeholders in narrowing down the search for optimal location for a new coffeeshop. By calculating coffeeshop density distribution from Foursquare data we have first identified general boroughs that justify further analysis (Salamah and Rawdah), and then generated extensive collection of locations which satisfy some basic requirements regarding existing nearby coffeeshops. Clustering of those locations was then performed in order to create major zones of interest (containing greatest number of potential locations) and addresses of those zone centers were created to be used as starting points for final exploration by stakeholders. Final decision on optimal coffeeshop location will be made by stakeholders based on specific characteristics of neighborhoods and locations in every recommended zone, taking into consideration additional factors like attractiveness of each location (proximity to park or water), levels of noise / proximity to major roads, real estate availability, prices, social and economic dynamics of every neighborhood etc.