Capstone Project - The Battle of the Neighborhoods (Week 2)

Applied Data Science Capstone by IBM/Coursera

Table of contents

- Introduction: Business Problem
- Data
- Methodology
- Analysis
- Results and Discussion
- Conclusion

Introduction: Business Problem

TAIPEI (Taiwan News) – According to statistics from 2017, Taiwan possesses the second highest ratio of convenience stores per population density of any country in the world. Taiwan's ratio of convenience stores is second only to South Korea, and barely ahead of Japan in third. There are 4 major convenience stores in Taiwan -- 7-11, Fami-mart, Hi-life, and OK-mart.

In this project we will try to find an optimal location for the biggest convenience store in Taiwan-- 7-11. Specifically, this report will be targeted to stakeholders interested in opening a new 7-11 in Taipei, Taiwan. The other convenience stores will not be concerned due to the high density. Having a different convenience store in neighborhood is very nature.

Since there are lots of 7-11 in Taipei we will try to detect **locations that are not already crowded with 7-11**. We would also prefer **locations as close to Taipei Main Station as possible**, assuming that first two conditions are met.

We will use our data science powers to generate a few most promising neighborhoods based on this criteria. Advantages of each area will then be clearly expressed so that best possible final location can be chosen by stakeholders.

Data

Based on definition of our problem, factors that will influence our decission are:

- number of existing 7-11 in the neighborhood
- number of and distance to 7-11 in the neighborhood, if any
- distance of neighborhood from Taipei Main Station

We decided to use *regularly spaced grid of locations, centered around Taipei Main Station, to define our neighborhoods.

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

- centers of candidate areas will be obtained using **Geopy reverse geocoding**
- number of 7-11 and their location in every neighborhood will be obtained using **Foursquare API**
- coordinate of Taipei Main Station will be obtained using **Geopy geocoding**

*reference to "Opening an Italian restaurant in Berlin, Germany."

Neighborhood Candidates

Let's create latitude & longitude coordinates for centroids of our candidate neighborhoods. We will create a grid of cells covering our area of interest which is aprox. 2x2 killometers centered around Taipei Main Station.

```
import requests
import pandas as pd
import numpy as np
import random
from geopy.geocoders import Nominatim
from pandas.io.json import json_normalize
import folium
#!pip install shapely
import shapely.geometry
#!pip install pyproj
import pyproj
import math
from geopy import distance

print('Lib loaded!')
```

```
Lib loaded!
```

Define Foursquare Credentials and Version

```
CLIENT_ID = '' # your Foursquare ID
CLIENT_SECRET = '' # your Foursquare Secret
VERSION = '20180604'
LIMIT = 100
print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

Let's first find the latitude & longitude of Taipei Main Station, using Geopy.

```
address= 'Taipei main station'
geolocator = Nominatim(user_agent="foursquare_agent")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print(latitude, longitude)
```

```
25.0462432 121.5174745
```

Find the latitude, longitude for 7-11 aprox. 2x2 killometers centered around Taipei Main Station. And then place them into a Pandas dataframe.

```
search_query='seven eleven'
radius=1000
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT
results = requests.get(url).json()
venues = results['response']['venues']
# tranform venues into a dataframe
dataframe = json_normalize(venues)
```

Take a look.

```
dataframe.head()
```

	id	name	categories	referralId	hasPerk	location.address	location.lat	location.lng	location.lab
0	4e14b7071f6e33e7180d5374	7-Eleven	[{'id': '4d954b0ea243a5684a65b473', 'name': 'C...	v-1584409152	False	重慶南路一段1之1號	25.046514	121.513430	[{'label': 25.0465
1	4bde74b0e75c0f47ddf6c703	7-Eleven	[{'id': '4d954b0ea243a5684a65b473', 'name': 'C...	v-1584409152	False	NaN	25.046194	121.517259	[{'label': 25.0461
2	530199b0498e580edfc772ba	7-Eleven	[{'id': '4d954b0ea243a5684a65b473', 'name': 'C...	v-1584409152	False	忠孝西路一段35號	25.045790	121.518670	[{'label': 25.0457
3	4e8c732c754abcc98da3e483	7-Eleven	[{'id': '4d954b0ea243a5684a65b473', 'name': 'C...	v-1584409152	False	公園路13號	25.045522	121.517577	[{'label': 25.0455
4	4e9c41634fc602a58b380102	7-Eleven	[{'id': '4d954b0ea243a5684a65b473', 'name': 'C...	v-1584409152	False	許昌街17號	25.045637	121.517007	[{'label': 25.0456

Clean the dataframe, only keep the needed information.

```
df=dataframe.rename(columns={'location.lat':'lat','location.lng':'lon','location.distance':'distance'})
df=df[['name','lat','lon']]
df.head()
```

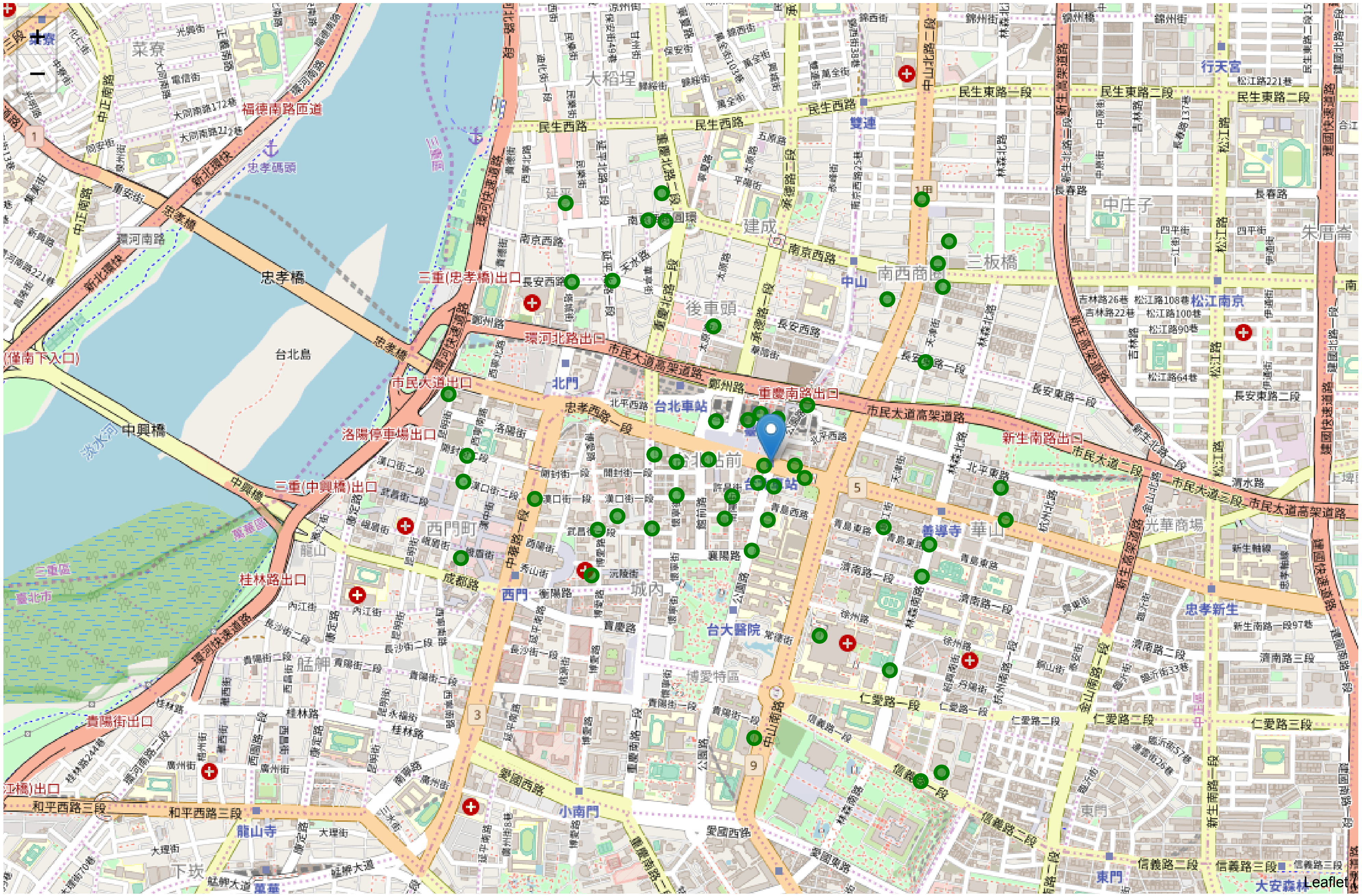
	name	lat	lon
0	7-Eleven	25.046514	121.513430
1	7-Eleven	25.046194	121.517259
2	7-Eleven	25.045790	121.518670
3	7-Eleven	25.045522	121.517577
4	7-Eleven	25.045637	121.517007

Let's visualize the data: **Taipei Main Station location and current 7-11 centers**

```
def draw_map(df,color,map_res,rad=5,fill=False):

  for lat, lng, label in zip(df.lat, df.lon, df.name):
      folium.CircleMarker(
          [lat, lng],
          radius=rad,
          color=color,
          popup=label,
          fill = fill,
          #fill_color=color,
          fill_opacity=0.6
      ).add_to(map_res)


map_se = folium.Map(location=[latitude, longitude], zoom_start=15)
folium.Marker(location=[latitude, longitude], popup=address).add_to(map_se)
draw_map(df,'green',map_se,fill=True)
map_se
```



+
−
大稻埕
民生西路
民生東路一段
民生東路二段
雙連
行天宮
中山
南西商圈
建成
後車頭
三板橋
中庄子
朱厝崙
松江南京
忠孝橋
台北島
中興橋
淡水河
北門
台北車站
台北站前
台大醫院
市民大道高架道路
忠孝西路一段
西門町
西門
城內
博愛特區
艋舺
龍山寺
小南門
善導寺
華山
光華商場
忠孝新生
東門
大安森林
仁愛路一段
信義路二段
和平西路三段
Leaflet

Now let's create a grid of area candidates, equaly spaced, centered around city center and within ~1km from Taipei Main Station. Our neighborhoods will be defined as circular areas with a radius of 100 meters, so our neighborhood centers will be 200 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'll project those coordinates back to latitude/longitude degrees to be shown on Folium map. So let's 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=51, 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=51, 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)
```

Get the latitude, longitude of the candidate neighborhood centers:

```
center_x, center_y = lonlat_to_xy(longitude, latitude) # City center in Cartesian coordinates

k = math.sqrt(3) / 2 # Vertical offset for hexagonal grid cells
x_min = center_x - 1000
x_step = 200
y_min = center_y - 1000 - (int(11/k)*k*200 - 2000)/2
y_step = 200 * k

latitudes = []
longitudes = []
distances_from_center = []
```

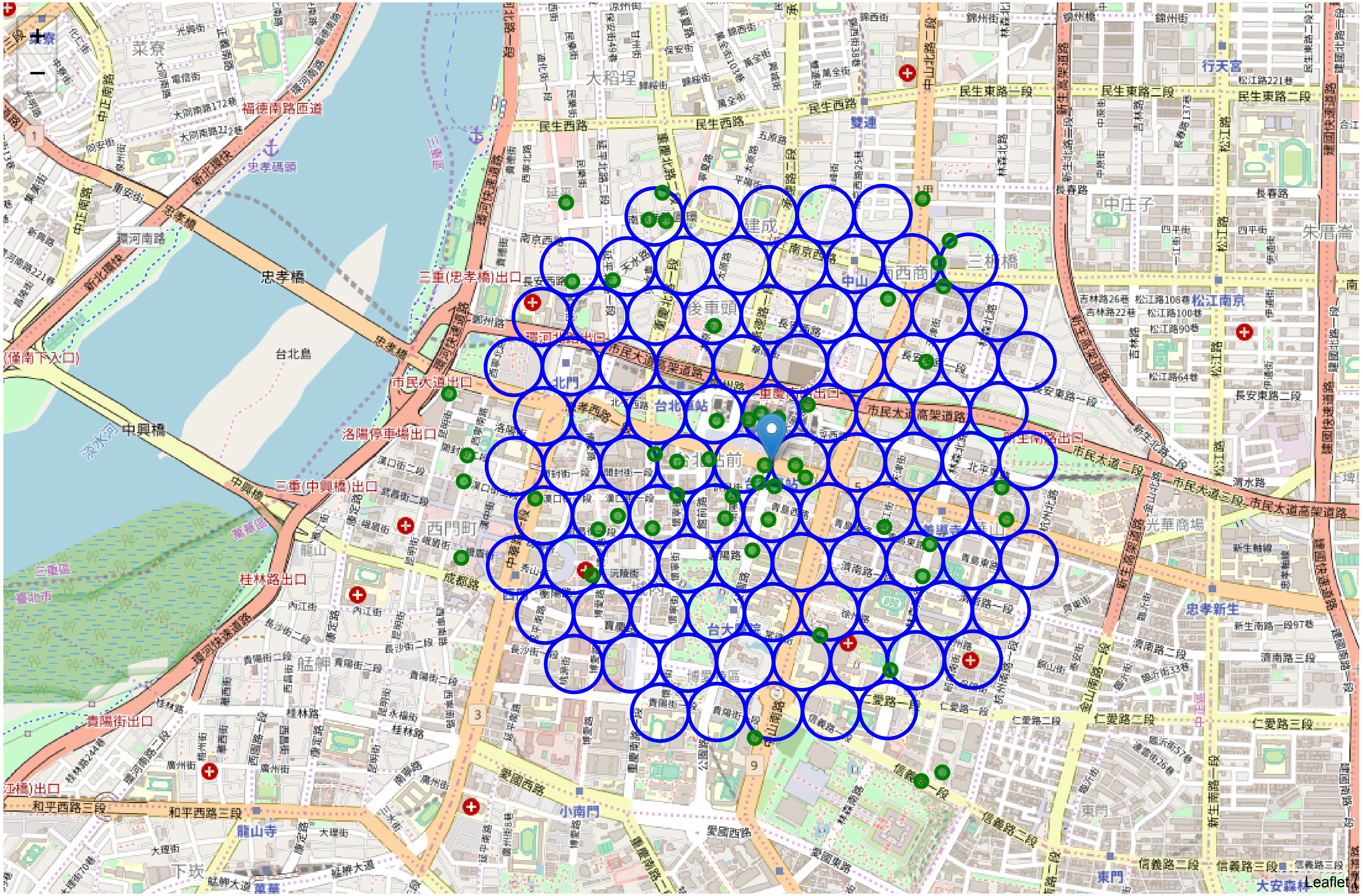
```
xs = []
ys = []
for i in range(0, int(21/k)):
    y = y_min + i * y_step
    x_offset = 100 if i%2==0 else 0
    for j in range(0, 11):
        x = x_min + j * x_step + x_offset
        distance_from_center = calc_xy_distance(center_x, center_y, x, y)
        if (distance_from_center <= 1001):
            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)
```

Let's visualize the data: **Taipei Main Station center location and candidate neighborhood centers**

```
map_res = folium.Map(location=[latitude, longitude], zoom_start=15)
folium.Marker(location=[latitude, longitude], popup=address).add_to(map_res)

for lat, lon in zip(latitudes, longitudes):
    folium.Circle([lat, lon], radius=100, color='blue', fill=False).add_to(map_res)

draw_map(df,'green',map_res, fill=True)
map_res
```


+
−
Leaflet
台北車站
北門
西門町
中山
雙連
善導寺
台大醫院
小南門
龍山寺
忠孝橋
中興橋
台北島
大稻埕
後車頭
建成
三重區
臺北市

Methodology

In this project we will direct our efforts on detecting areas of Taipei that have low 7-11 density. We will limit our analysis to area ~1km around Taipei Main Station.

In first step we have collected the required **data: location of every 7-11 within 1km from Taipei Main Station**.

Second step in our analysis will be calculation and exploration of '**7-11 density**' across different areas of Taipei - we will identify a few promising areas close to Taipei Main Station with low number of 7-11 in general and focus our attention on those areas.

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 will take into consideration locations with **no more than two restaurants in radius of 100 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

Let's perform some basic explanatory data analysis and derive some additional info from our raw data. First let's count the **number of 7-11 in every area candidate**:

```
def count_near_data(search_query, lat, lon, radius=100, LIMIT=100):
  global CLIENT_ID, CLIENT_SECRET, VERSION
  url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIE
  results = requests.get(url).json()
  venues = results['response']['venues']
  # tranform venues into a dataframe
  dataframe = json_normalize(venues)
  return len(dataframe)


count_store=[]
for lat, lon in zip(latitudes, longitudes):
    count_store.append(count_near_data(search_query,lat,lon))
```

```
print('There are '+str(count_store.count(0))+' candidate neighborhood centers without other 7-11 in 100 Meters.')
```

```
There are 42 candidate neighborhood centers without other 7-11 in 100 Meters.
```

Place them into a Pandas dataframe.

```
df_count=pd.DataFrame({'lat':latitudes, 'lon':longitudes, 'count':count_store})
df_count.head()
```

	lat	lon	count
0	25.038385	121.513605	0
1	25.038405	121.515587	0
2	25.038425	121.517569	1
3	25.038444	121.519550	0
4	25.038464	121.521532	0

Remove the data with other 7-11 within 100 Meters.

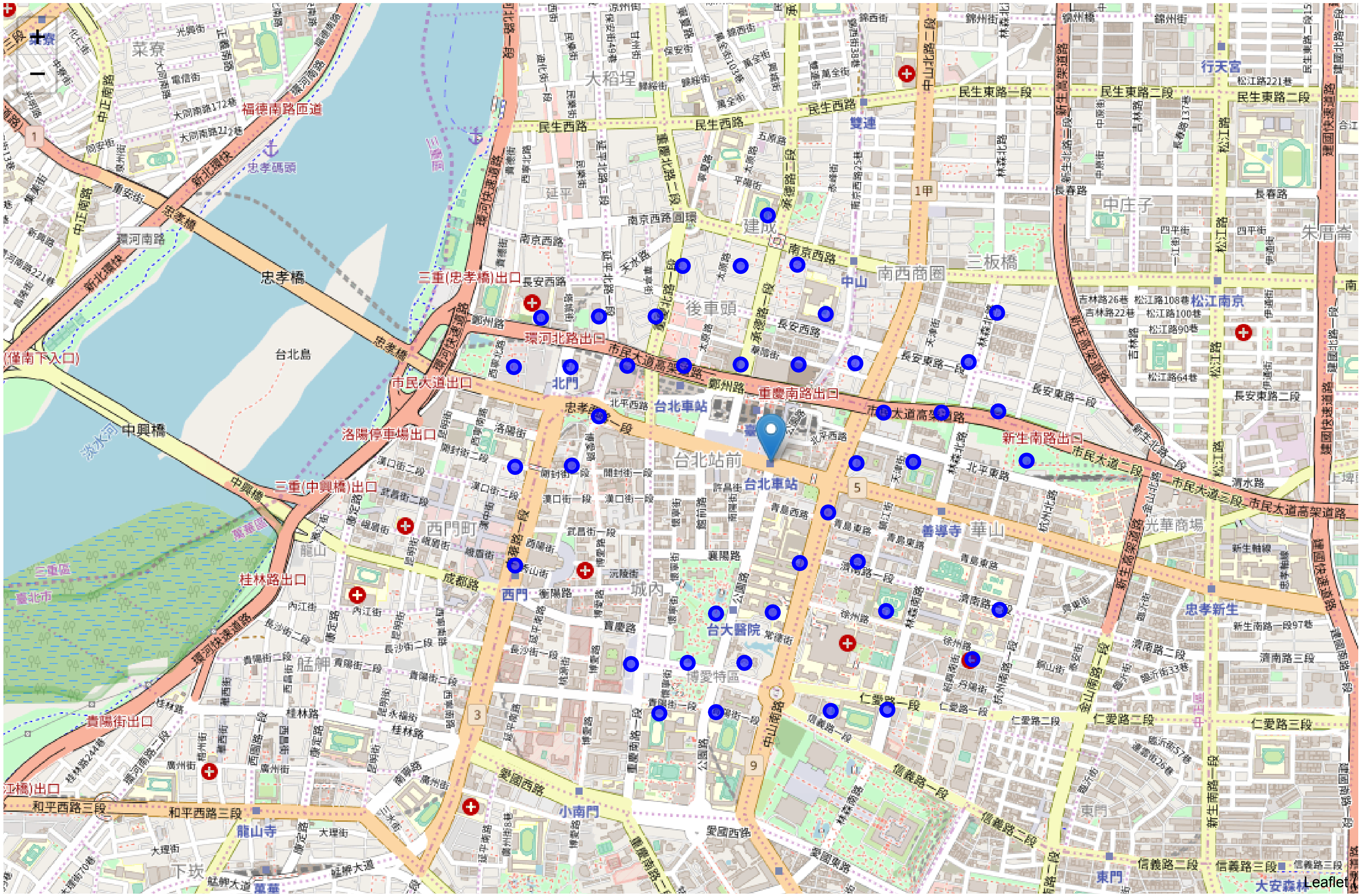
```
df_count=df_count.query('count==0')
df_count['name']='candidate'
df_count.head()
```

	lat	lon	count	name
0	25.038385	121.513605	0	candidate
1	25.038405	121.515587	0	candidate
3	25.038444	121.519550	0	candidate
4	25.038464	121.521532	0	candidate
6	25.039939	121.512595	0	candidate

Let's visualize the data: **the centers without other 7-11 within 100 Meters**

```
map_cad = folium.Map(location=[latitude, longitude], zoom_start=15)
folium.Marker(location=[latitude, longitude], popup=address).add_to(map_cad)

#draw_map(df,'green',map_res)
draw_map(df_count, 'blue', map_cad, fill=True)
map_cad
```


台北車站
北門
西門
台大醫院
中山
雙連
善導寺
忠孝新生
小南門
龍山寺
東門
大安森林
松江南京
行天宮
Leaflet

Let us now **cluster** those locations by **Kmeans** to create **centers of zones containing candidate locations**. Those zones, their centers and addresses will be the final result of our analysis.

```
from sklearn.cluster import KMeans

number_of_clusters = 5
clf = KMeans(n_clusters=number_of_clusters, random_state=0).fit(df_count[['lat','lon']].values)
df_count['label']=clf.labels_
df_count.head()
```

	lat	lon	count	name	label
0	25.038385	121.513605	0	candidate	3
1	25.038405	121.515587	0	candidate	3
3	25.038444	121.519550	0	candidate	1
4	25.038464	121.521532	0	candidate	1
6	25.039939	121.512595	0	candidate	3

Get the latitude, longitude for the cluster centers:

```
df_rec=pd.DataFrame(clf.cluster_centers_)
df_rec.columns=['lat','lon']
df_rec['name']='recommended'
df_rec
df_rec.head()
```

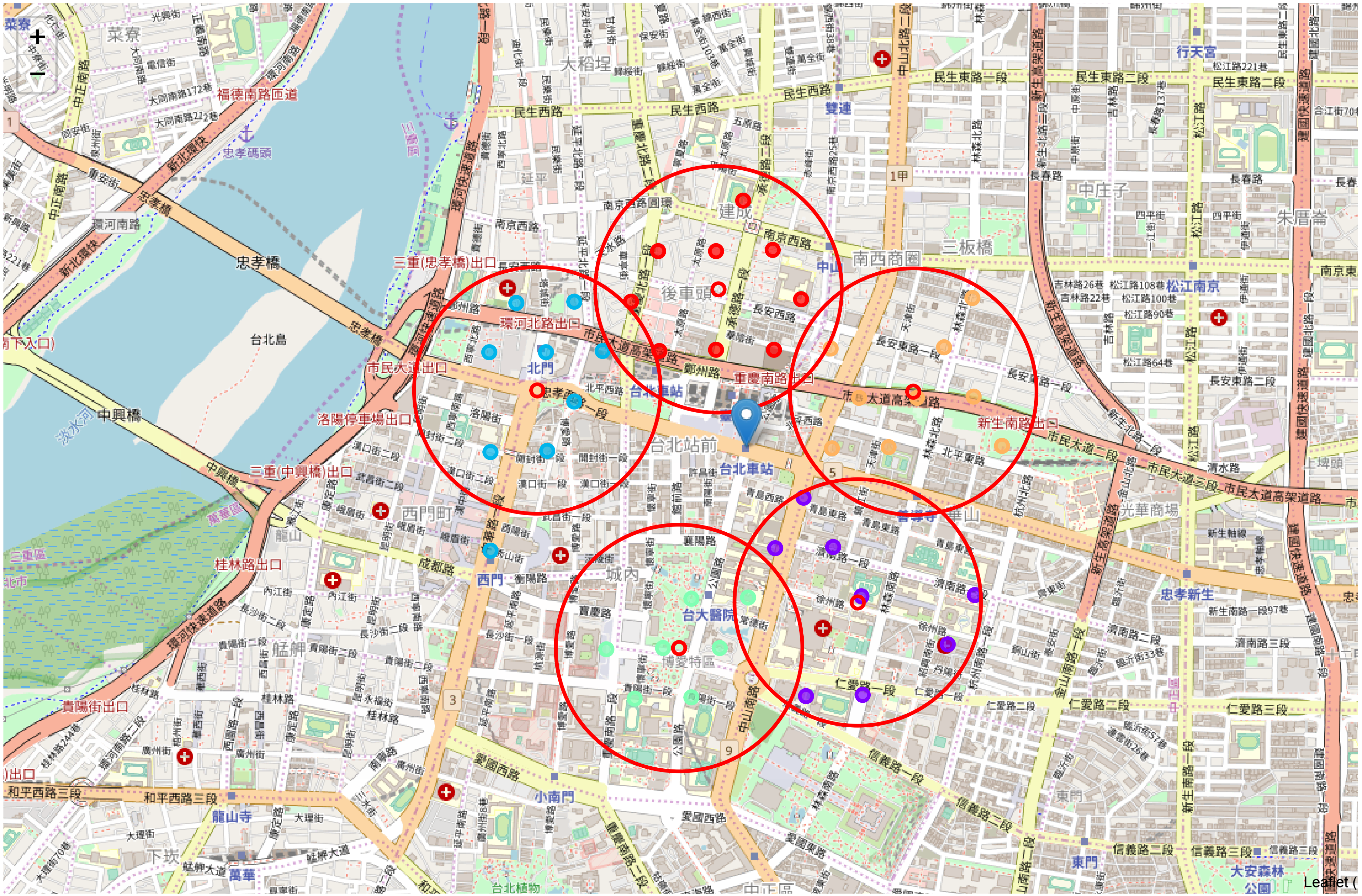
	lat	lon	name
0	25.051273	121.516533	recommended
1	25.041395	121.521373	recommended
2	25.048082	121.510184	recommended
3	25.039964	121.515143	recommended
4	25.048039	121.523290	recommended

Let's visualize the data: **candidate neighborhood centers and the cluster centers**

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

map_clf = folium.Map(location=[latitude, longitude], zoom_start=15)

# set color scheme for the clusters
x = np.arange(number_of_clusters)
ys = [i + x + (i*x)**2 for i in range(number_of_clusters)]
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, cluster in zip(df_count['lat'], df_count['lon'], df_count['label']):
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup='Label: '+str(cluster),
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clf)
folium.Marker(location=[latitude, longitude], popup=address).add_to(map_clf)
draw_map(df_rec,'red',map_clf,rad=5)
draw_map(df_rec,'red',map_clf,rad=100)
map_clf
```


Leaflet (

Finaly, let's reverse geocode those candidate area centers to get the addresses which can be presented to stakeholders.

```
addresses = []
dist=[]
for lat, lon in zip(df_rec.lat, df_rec.lon):
    location=geolocator.reverse(str(lat)+","+str(lon))
    address=location.address
    dist.append(round(distance.distance((lat, lon),(latitude, longitude)).km,2))
    #print('{}{} => {:.2f}km from Taipei Main Station'.format(address, ' '*(50-len(address)), dist))
    if address is None:
        address = 'NO ADDRESS'

    addresses.append(address)
df_col=pd.DataFrame.from_dict({'Address':addresses,'Distance(km)':dist})
print('Below is the recommended address and the distance from Taipei Main Station list:')
df_col
```

```
Below is the recommended address and the distance from Taipei Main Station list:
```

	Address	Distance(km)
0	新逸旅店, 長安西路, 建明里, 後車頭, 大同區, 臺北市, 10444, Taiwan	0.57
1	活動中心, 徐州路, 東門里, 華山, 中正區, 臺北市, 10055, Taiwan	0.67
2	臺北築城材料及構造特色, 忠孝西路一段, 光復里, 台北站前, 中正區, 臺北市, 1001...	0.76
3	二二八和平公園, 懷寧街, 黎明里, 城內, 中正區, 臺北市, 10045, Taiwan	0.73
4	市民大道二段, 正得里, 南西商圈, 中山區, 臺北市, 10441, Taiwan	0.62

Results and Discussion

The analysis shows that although there is a great number of 7-11 in Taipei (~1000 Meters in the initial area of interest which was 2x2km around Taipei Main Station), there are pockets of low 7-11 density fairly close to Taipei Main Station.

We first created a dense grid of location candidates (spaced 100m appart); those locations were then filtered so that those with more than two 7-11 closer than 200m were removed.

Those location candidates 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 5 zones containing largest number of potential new 7-11 locations based on number of and distance to existing venues. This, of course, does not imply that those zones are actually optimal locations for a new 7-11! Purpose of this analysis was to only provide info on areas close to Taipei Main Station but not crowded with existing 7-11 - it is entirely possible that there is a very good reason for small number of 7-11 in any of those areas, reasons which would make them unsuitable for a new 7-11 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 Taipei Main Station with low number of 7-11 in order to aid stakeholders in narrowing down the search for optimal location for a new 7-11. By calculating 7-11 density distribution from Foursquare data we have first identified general boroughs that justify further analysis, and then generated extensive collection of locations which satisfy some basic requirements regarding existing nearby 7-11. Clustering of those locations was then performed in order to create major zones of interest and addresses of those zone centers were created to be used as starting points for final exploration by stakeholders.

Final decission on optimal 7-11 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 major roads, real estate availability, prices, social and economic dynamics of every neighborhood etc.

PS. BIG THANKS to the author of "Opening an Italian restaurant in Berlin, Germany", there is no need to recode/retype this notebook from scratch.