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

Table of Contents

1 Introduction: Business Problem

In this project we will try to find an optimal location for opening a restaurant in **Amsterdam, Netherlands** near *Central Railway Station*. Our sole sole objective is to find location that is surrounded by least number of restaurants and is nearest to *Central Railway Station* which is the famous transport center in Europe. Since there are lots of restaurants in this location we will try to detect locations that are not already crowded with restaurants. We are also particularly interested in areas with no restaurants in vicinity. We would also prefer locations as close to *Central Railway Station* of Amsterdam 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.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.

2 Data

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

- number of existing restaurants in the neighborhood (any type of restaurant)
- distance of neighborhood from *Central Railway Station of Amsterdam*

We decided to use regularly spaced grid of locations, centered around *Central Railway Station of Amsterdam*, to define our neighborhoods.

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

- 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**
- number of restaurants and their type and location in every neighborhood will be obtained using **Foursquare API**
- coordinate of *Central Railway Station of Amsterdam* will be obtained using **Google Maps API geocoding**

Steps for Foursquare API

1. Since we want to select addresses for a new restaurant, we can use Foursquare API to run a search query of food.

2. To get geographical coordinates of Amsterdam Centraal by geolocator (Nominatim).

```
address = 'Amsterdam Centraal, Netherlands'

geolocator = Nominatim(user_agent="amsterdam_agent")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print("The geographical coordinates of Amsterdam Centraal is {}, {}.".format(latitude, longitude))
```

3. Get data in the form of json file and keep only columns include venue name and anything that is associated with location.

```
results = requests.get(url).json()
results
```

```
# keep only columns that include venue name, and anything that is associated with location
filtered_columns = ['name', 'categories'] + [col for col in dataframe.columns if col.startswith('location.')] + ['id']
dataframe_filtered = dataframe.loc[:, filtered_columns]

# 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']

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

# clean column names by keeping only last term
dataframe_filtered.columns = [column.split('.')[-1] for column in dataframe_filtered.columns]

dataframe_filtered.head(10)
```

Steps for Google Maps API reverse geocoding

In [22]:

```
google_api_key='AIzaSyB8Qd0k3UbmTiC4xoS4nGnM9ZvQYX9nVFk'
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, centraal[0], centraal[1])
print('Reverse geocoding check')
print('---------------------')
print('Address of [{}, {}] is: {}'.format(centraal[0], centraal[1], addr))
```

```
Reverse geocoding check
---------------------
Address of [52.3791, 4.9003] is: Amsterdam Centraal, Stationsplein, 1012 AB Amsterdam, Netherlands
```

3 Methodology