Finding Locations to Open a Bar in Amsterdam

IBM DATA SCIENCES CAPSTONE PROJECT

Qingting.Song April 2019

Data Collection

- We need to collect names of all the neighbourhoods of Amsterdam. This can be achieved by scraping the Wiki page: https://en.wikipedia.org/wiki/Category:Neighbourhoods_of_Amsterdam

```
url = requests.get('https://en.wikipedia.org/wiki/Category:Neighbourhoods_of_Amsterdam').text
soup = BeautifulSoup(url,'html.parser')
```

- Get Geo data from https://geocoder.readthedocs.io/index.html. This will request and pull all the latitude and longitude data of Amsterdam Neighbourhoods which will be used to create map of Amsterdam.

```
def get_latlng(neighborhood):
    # initialize your variable to None
    lat_lng_coords = None
    # loop until you get the coordinates
    while(lat_lng_coords is None):
        g = geocoder.arcgis('{}, Amsterdam, Netherlands'.format(neighborhood))
        lat_lng_coords = g.latlng
    return lat_lng_coords
```

```
# call the function to get the coordinates, store in a new list using list comprehension
coords = [ get_latlng(neighborhood) for neighborhood in AMS_df["Neighborhood"].tolist() ]
# create temporary dataframe to populate the coordinates into Latitude and Longitude
df_coords = pd.DataFrame(coords, columns=['Latitude', 'Longitude'])
df_coords.to_csv("df_coords.csv", index=False)
```

```
# merge the coordinates into the AMS_df dataframe
AMS_df['Latitude'] = df_coords['Latitude']
AMS_df['Longitude'] = df_coords['Longitude']
AMS_df.head(15)
```

	Neighborhood	Latitude	Longitude
0	Admiralenbuurt	52.372728	4.856362
1	Amsteldorp	52.360420	4.905250
2	Amsterdam Oud-West	52.365390	4.870220

- Request Foursquare data of all the venues in Amsterdam. We will merge this dataset with geographic data of Amsterdam and use this combined dataset as input of clustering analysis, and find out the most suitable location for opening a small bar.

```
LIMIT = 300
radius = 3000


venues = []

for lat, long, neighborhood in zip(AMS_df['Latitude'], AMS_df['Longitude'], AMS_df['Neighborhood']):
    # create the API request URL
    url = "https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&v={}&ll={},{}&
        CLIENT_ID,
        CLIENT_SECRET,
        VERSION,
        lat,
        long,
        radius,
        LIMIT)
    # make the GET request
    results = requests.get(url).json()["response"]['groups'][0]['items']

    # return only relevant information for each nearby venue
    for venue in results:
        venues.append((
            neighborhood,
            lat,
            long,
            venue['venue']['name'],
            venue['venue']['location']['lat'],
            venue['venue']['location']['lng'],
            venue['venue']['categories'][0]['name']))
```