IBM DATA SCIENCE PROFESSIONAL CERTIFICATION

CAPSTONE PROJECT ON

COMMUNITY MAPPING WITH GEOGRAPHICAL COORDINATES USING FOURSQUARE RESOURCES

BY

IDOKO JOB JOHN

April 2020

Abstract

Community mapping is essential for navigation, town planning and development, and construction projects that can alter the landscapes of the community. Foursquare Corporation provide a medium for obtaining and exploring addresses of locations and getting their geographical coordinates among other numerous, exciting functions. Using Python 3.6 with its relevant Python packages, and by employing Foursquare resources, the map of Staten Island borough of New York City was generated, and its neighbourhoods were juxtaposed on the map.

Table of Contents

1.0 Introduction/Business Problem

Community mapping involves creating pictorial representations of the geographical entities: settlements, landmarks, roads, institutions, topographies and other physical features of the area occupied by a constitutionally recognised group of people. Products of community mapping or maps are useful in a wide range of applications; from simple, individual projects like navigating a city to advanced, capital-intensive projects that involve multiple corporations like the building of stadiums and airports to rail track and trajectory design. In this study, an example is used to illustrate the methodology for community mapping with Foursquare locator resources through relevant Python libraries.

This document provides a useful tool for the aviation industry, health workers who need to create maps of locations where they need to embark on outreaches and everyone who needs to create a map of a setting knowing only the address of the place.

2.0 Method of Study

In this study, the map of Staten Island was generated in Python 3.6 using Foursquare geolocator resources and the relevant Python Libraries and Packages as described the following sections.

4.1 Importing Required Python Libraries

```
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 # converts an address into latitude and longitude values
import requests # library to handle geolocator 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


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

Figure 1: Python Syntaxes for Importing the Relevant Libraries

The Python packages used for this task are: (i) NumPy for handling data in vectorised manner; (ii) pandas for data analysis; (iii) json for handling JSON files; (iv) requests, geopy and Nominatim as APIs and for obtaining and converting addresses into their latitude and longitude values; (v) json_normalize for converting JSON files into pandas data frame; (vi) matplotlib and associated plotting modules and; (vii) folium for map creation. A screenshot of the Syntax for this section is shown in Figure 1.

4.2 Data Acquisition and Processing

```
!wget -q -O 'newyork_data.json' https://cocl.us/new_york_dataset
with open('newyork_data.json') as json_data:
    newyork_data = json.load(json_data)
neighborhoods_data = newyork_data['features']
column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude']
neighborhoods = pd.DataFrame(columns=column_names)
for data in neighborhoods_data:
    borough = neighborhood_name = data['properties']['borough']
    neighborhood_name = data['properties']['name']
    neighborhood_latlon = data['geometry']['coordinates']
    neighborhood_lat = neighborhood_latlon[1]
    neighborhood_lon = neighborhood_latlon[0]
    neighborhoods = neighborhoods.append({'Borough': borough,
                                          'Neighborhood': neighborhood_name,
                                          'Latitude': neighborhood_lat,
                                          'Longitude': neighborhood_lon}, ignore_index=True)
staten_island = neighborhoods[neighborhoods['Borough'] == 'Staten Island'].reset_index(drop=True)
staten_island.head()
```

Figure 2: Python Syntax for obtaining and processing the needed data

This exercise focus on the neighbourhoods of Staten Island borough in the City of New York as provided by the NYU Spatial Data Repository. Data scraping and storage into JSON data file were done by Alex Aklson and Polong Lin. The python Syntax for extracting the needed data, cleaning and turning it into a Pandas data frame is shown in Figure 2. While the first ten rows of the data frame are shown in Figure 3.

	Borough	Neighborhood	Latitude	Longitude
0	Staten Island	St. George	40.644982	-74.079353
1	Staten Island	New Brighton	40.640615	-74.087017
2	Staten Island	Stapleton	40.626928	-74.077902
3	Staten Island	Rosebank	40.615305	-74.069805
4	Staten Island	West Brighton	40.631879	-74.107182
5	Staten Island	Grymes Hill	40.624185	-74.087248
6	Staten Island	Todt Hill	40.597069	-74.111329
7	Staten Island	South Beach	40.580247	-74.079553
8	Staten Island	Port Richmond	40.633669	-74.129434
9	Staten Island	Mariner's Harbor	40.632546	-74.150085

Figure 3: Snapshot of Pandas data frame showing the first 10 rows of the data (Neighbourhoods of Staten Island with their Latitude and Longitude)

4.3 Employing the Foursquare Resources and Python Libraries to Generate a Community Map of Staten Island

4.3.1 Defining an instance of the geocoder and a user agent to get the geographical coordinates of a location

Figure 4 shows the function for defining an instance of the geocoder created with a user agent named 'ny_explorer' and generating the geographical coordinates of Staten Island Borough.

```
address = 'Staten Island, NY'

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

```
The geograpical coordinate of Staten Island are 40.5834557, -74.1496048.
```

Figure 4: Function for Generating an Instance of the Geocoder Locator

4.3.2 Creating the Community Map

The Foursquare geolocator resources were used to obtain the latitudes and longitudes of the neighbourhoods in Staten Island and Python folium packages were used to generate map of Staten Island with the neighbourhoods superimposed on it using the function in Figure 5.

```
map_staten_island = folium.Map(location=[latitude, longitude], zoom_start=11)


for lat, lng, label in zip(staten_island['Latitude'], staten_island['Longitude'], staten_island['Neighborhood']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_staten_island)

map_staten_island
```

Figure 5: Python syntax for generating community map using Foursquare resources

5.0 Result Presentation, Analysis and Discussion

Figure 6 displays the map of Staten Island borough showing neighbourhoods juxtaposed as blue dots. This map will be useful for outreaches in Staten Island such as city-to-city community sensitisation and intra-city and inter-city developmental projects like road and railway planning and construction.

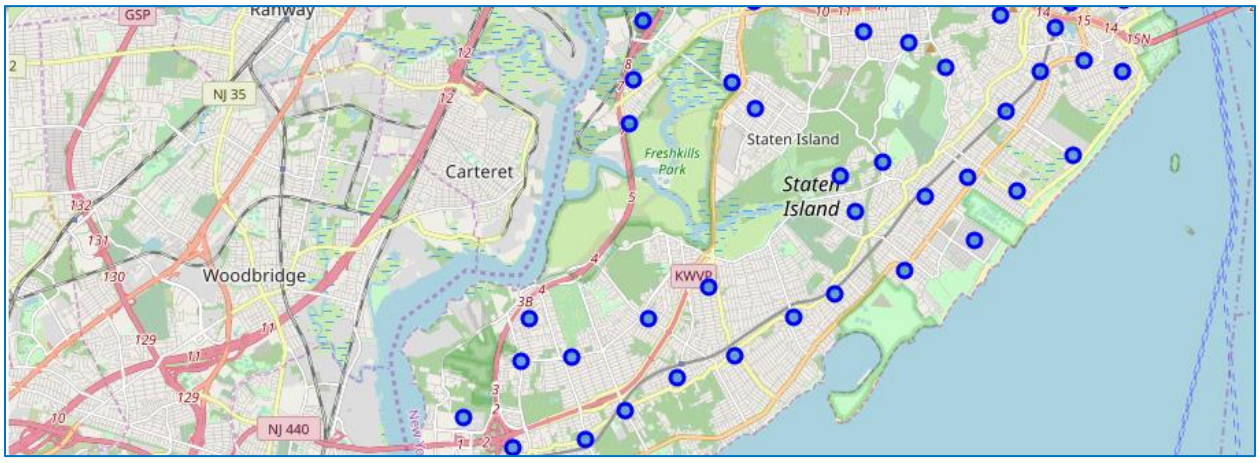

Figure 6: Map of Staten Island with Neighbourhoods indicated in blue.