Capstone project

For

IBM Data Science Specialization

Urban Happiness

Ayushi Sharma

sharmayu20@gmail.com

Introduction/Business Problem

Relocating from one place to another is in itself a very difficult task for a person, he/she has to decide whether the neighborhood has the required venues that he/she likes and on top of that whether the neighborhood is safe.

What if a family man wants to move and is unaware of the environment around his selected neighborhood? This is where the project Urban Happiness comes in.

Urban Happiness is a project written in Python that can help a user decide his/her favorable neighborhood based on the venue that he requires and keeping in mind the crime rate of the neighborhood in city of San Francisco.

Urban Happiness presents a map to the user with the crime ranges of the locality and the clustered neighborhood based on the venues. It also outputs the closest neighborhood from the venue given by the user.

The map marks two neighborhoods that are close to the venue given by the user.

The project clusters the neighborhoods based on all the venues and the crime rate in that neighborhood into three clusters and shows it to the user via a map.

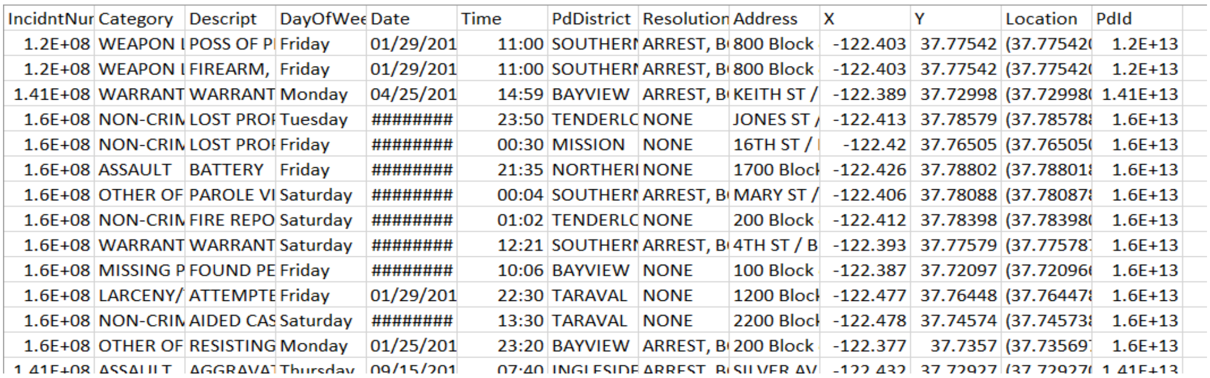
Data

The data that will be required for the project would be the crimes rate in San Francisco based on the neighborhoods, the json file that contains the coordinates of the neighborhoods in the form of a polygon which can then be used by folium to mark the neighborhoods, then the postal code or the pincode of the neighborhoods which is then used to gather the exact latitude and the longitude of the neighborhood, then the venue list that is present in the given locality which can be fetched using the foursquare API.

The following section has a detailed description of all the data that will be used for the completion of the project.

1. Crime rate in San Francisco
   This data file is a csv file that contains the arrests or crimes that were committed in different neighborhood of San Francisco.
   The important columns that are considered for the project are:
   - District
   - Count of crimes in each district

   Snapshot of the csv file used.

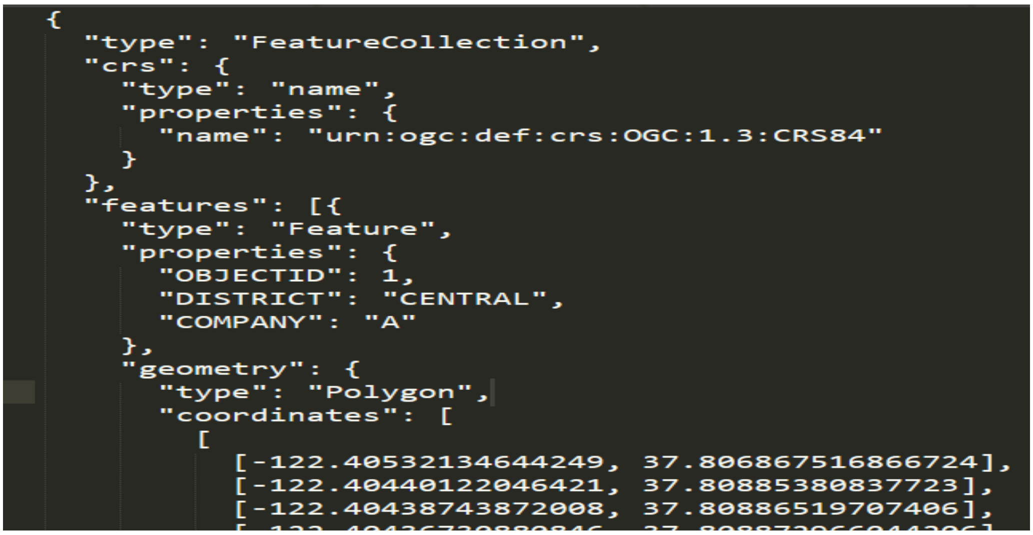


IncidntNur	Category	Descript	DayOfWee	Date	Time	PdDistrict	Resolution	Address	X	Y	Location	PdId
1.2E+08	WEAPON L	POSS OF P	Friday	01/29/201	11:00	SOUTHERN	ARREST, B	800 Block	-122.403	37.77542	(37.77542	1.2E+13
1.2E+08	WEAPON L	FIREARM,	Friday	01/29/201	11:00	SOUTHERN	ARREST, B	800 Block	-122.403	37.77542	(37.77542	1.2E+13
1.41E+08	WARRANT	WARRANT	Monday	04/25/201	14:59	BAYVIEW	ARREST, B	KEITH ST /	-122.389	37.72998	(37.72998	1.41E+13
1.6E+08	NON-CRIM	LOST PROP	Tuesday	########	23:50	TENDERLO	NONE	JONES ST /	-122.413	37.78579	(37.78578	1.6E+13
1.6E+08	NON-CRIM	LOST PROP	Friday	########	00:30	MISSION	NONE	16TH ST /	-122.42	37.76505	(37.76505	1.6E+13
1.6E+08	ASSAULT	BATTERY	Friday	########	21:35	NORTHERN	NONE	1700 Block	-122.426	37.78802	(37.78801	1.6E+13
1.6E+08	OTHER OF	PAROLE VI	Saturday	########	00:04	SOUTHERN	ARREST, B	MARY ST /	-122.406	37.78088	(37.78087	1.6E+13
1.6E+08	NON-CRIM	FIRE REPO	Saturday	########	01:02	TENDERLO	NONE	200 Block	-122.412	37.78398	(37.78398	1.6E+13
1.6E+08	WARRANT	WARRANT	Saturday	########	12:21	SOUTHERN	ARREST, B	4TH ST / B	-122.393	37.77579	(37.77578	1.6E+13
1.6E+08	MISSING P	FOUND PE	Friday	########	10:06	BAYVIEW	NONE	100 Block	-122.387	37.72097	(37.72096	1.6E+13
1.6E+08	LARCENY/	ATTEMPTE	Friday	01/29/201	22:30	TARAVAL	NONE	1200 Block	-122.477	37.76448	(37.76447	1.6E+13
1.6E+08	NON-CRIM	AIDED CAS	Saturday	########	13:30	TARAVAL	NONE	2200 Block	-122.478	37.74574	(37.74573	1.6E+13
1.6E+08	OTHER OF	RESISTING	Monday	01/25/201	23:20	BAYVIEW	ARREST, B	200 Block	-122.377	37.7357	(37.73569	1.6E+13
1.41E+08	ASSAULT	AGGRAVAT	Thursday	09/15/201	07:40	INGLESIDE	ARREST, B	SILVER AV	-122.432	37.72927	(37.72927	1.41E+13

2. GeoJson file that contains information on San Francisco

   This GeoJson file contains the information on San Francisco in the form of key value pairs. The file contains the coordinates of different neighborhoods in San Francisco. These coordinates are given to the folium's geo_data attribute of the Chloropleth class.
   This geo_data is responsible for marking the districts and the colors for the folium map.

Snapshot of the geoJson file

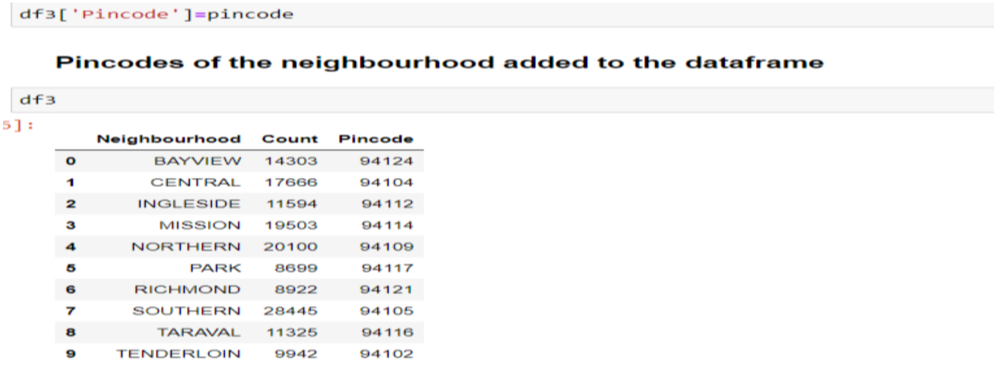


```
{
  "type": "FeatureCollection",
  "crs": {
    "type": "name",
    "properties": {
      "name": "urn:ogc:def:crs:OGC:1.3:CRS84"
    }
  },
  "features": [{
    "type": "Feature",
    "properties": {
      "OBJECTID": 1,
      "DISTRICT": "CENTRAL",
      "COMPANY": "A"
    },
    "geometry": {
      "type": "Polygon",
      "coordinates": [
        [
          [-122.40532134644249, 37.806867516866724],
          [-122.40440122046421, 37.80885380837723],
          [-122.40438743872008, 37.80886519707406],
```

3. Postal codes

Since there was no website that provided the postal codes of the neighborhood, the postal codes were manually added to the data frame by adding another column to the data frame. Postal codes were manually searched from the in internet and added to the data frame.

Snapshot of the modified data frame.



```
df3['Pincode']=pincode
```

Pincodes of the neighbourhood added to the dataframe

```
df3
```

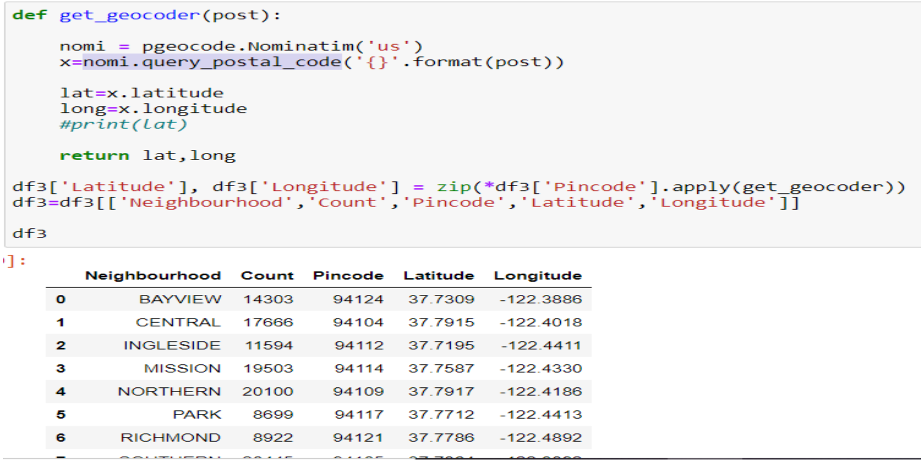
5]:

	Neighbourhood	Count	Pincode
0	BAYVIEW	14303	94124
1	CENTRAL	17666	94104
2	INGLESIDE	11594	94112
3	MISSION	19503	94114
4	NORTHERN	20100	94109
5	PARK	8699	94117
6	RICHMOND	8922	94121
7	SOUTHERN	28445	94105
8	TARAVAL	11325	94116
9	TENDERLOIN	9942	94102

4. GeoCoder:

Geocoder package is used to fetch the latitudes and the longitudes of place by passing the postal code to it.
This package has a function nomi.query_postal_code() which takes postal code as a input and outputs the longitude and the latitude of the place

Snapshot of the function used to access the longitude and the latitude and the result.



```
def get_geocoder(post):

    nomi = pgeocode.Nominatim('us')
    x=nomi.query_postal_code('{}'.format(post))

    lat=x.latitude
    long=x.longitude
    #print(lat)

    return lat,long

df3['Latitude'], df3['Longitude'] = zip(*df3['Pincode'].apply(get_geocoder))
df3=df3[['Neighbourhood','Count','Pincode','Latitude','Longitude']]

df3
```

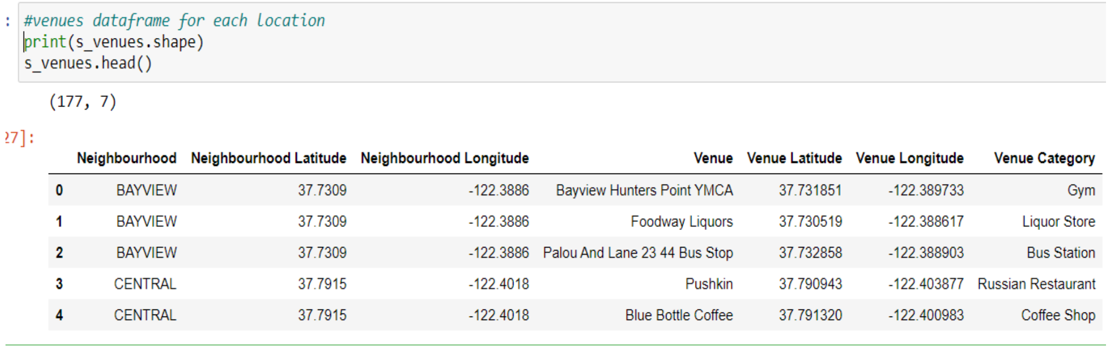
]:

	Neighbourhood	Count	Pincode	Latitude	Longitude
0	BAYVIEW	14303	94124	37.7309	-122.3886
1	CENTRAL	17666	94104	37.7915	-122.4018
2	INGLESIDE	11594	94112	37.7195	-122.4411
3	MISSION	19503	94114	37.7587	-122.4330
4	NORTHERN	20100	94109	37.7917	-122.4186
5	PARK	8699	94117	37.7712	-122.4413
6	RICHMOND	8922	94121	37.7786	-122.4892

5. <u>FourSquare API:</u>

The foursquare API is used to fetch the list of venues that are close to the given latitude and the longitude. The API uses the client ID and the Client Secret to fetch the details.
The url is then used on a get request method to the API, the url contains the client id, client secret, version of the Foursquare, latitude and longitude of the location, radius to be considered around the location and the limit as to fetch how many venues around the location.
The response is then stored in the form of json object. The response can contain details of the venue such as name, latitude and longitude of the venue, category of the venue, or rating or tip of the venues.
The response can then be converted to a pandas data frame and then be used for further operations.

Snapshot of the result after converting to pandas data frame:



```
: #venues dataframe for each location
  print(s_venues.shape)
  s_venues.head()

    (177, 7)
```

27]:

	Neighbourhood	Neighbourhood Latitude	Neighbourhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	BAYVIEW	37.7309	-122.3886	Bayview Hunters Point YMCA	37.731851	-122.389733	Gym
1	BAYVIEW	37.7309	-122.3886	Foodway Liquors	37.730519	-122.388617	Liquor Store
2	BAYVIEW	37.7309	-122.3886	Palou And Lane 23 44 Bus Stop	37.732858	-122.388903	Bus Station
3	CENTRAL	37.7915	-122.4018	Pushkin	37.790943	-122.403877	Russian Restaurant
4	CENTRAL	37.7915	-122.4018	Blue Bottle Coffee	37.791320	-122.400983	Coffee Shop