Atlan | Data Science Intern Challenge Documentation

Aim: Identify commercial centers using Points of Interest (POI) data

Objective:

1. Get Points of Interest from open data sources like open street maps (OSM).
2. Understand how spatial location data works
    1. Understand spatial vector data types and how to manipulate it using your language of choice.
    2. Understand necessary GIS concepts like projections, spatial clustering, etc.
3. Figure out a way of clustering these points into commercial centers/markets. You can use standard size polygons also to cluster the points.
4. Find and label the most significant clusters, statistically and intuitively.
5. Visualize the resultant commercial centres/markets.

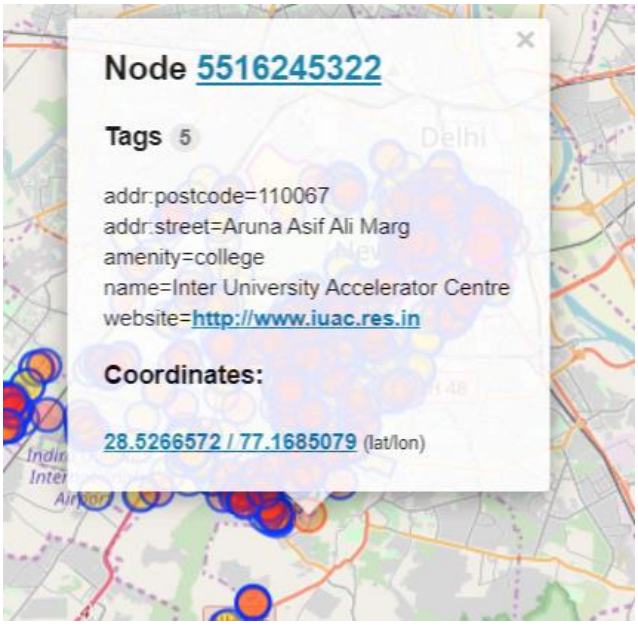
1. Getting point of interest(POI)data from OpenStreetMap

A point of interest (POI for short) is a term used in cartography (and therefore in reference to maps or geo datasets) for the choice to represent a particular feature using an icon that occupies a particular point. The idea is that, as opposed to linear features like roads or areas of landuse, some features might be suited to being indicated as a point in a particular context; for example, if one wanted to send a letter, it would be relevant to see all the post offices and mailboxes nearby, and if they are all represented by an envelope icon, it is easy to see.

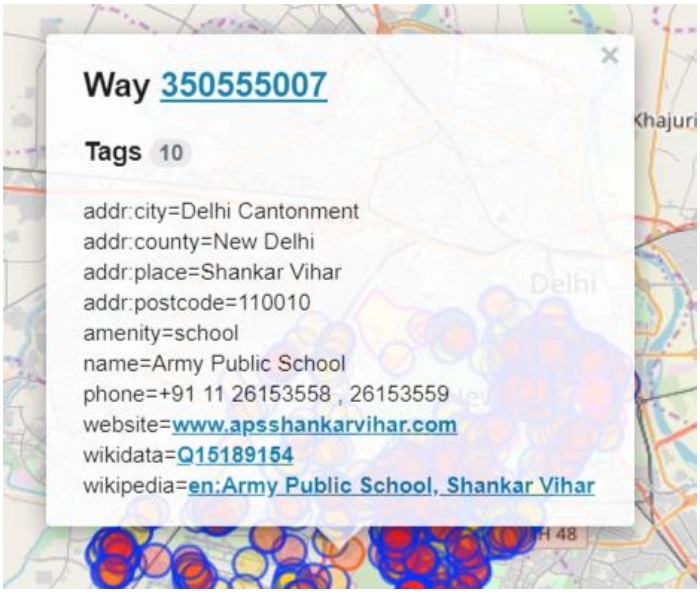
Some example of types of POI:

- Churches, schools, town halls, distinctive buildings
- Post offices, shops, post boxes, telephone boxes
- Pubs (pub names are useful when navigating by map)
- Car parks and lay-bys
- Speed cameras
- Tourist attractions

Before we start, we have to take a look at how OSM is structured. We have three basic components in the OSM data model, which are nodes, ways and relations which all come with an id. Many of the elements come with tags which describe specific features represented as key-value pairs. In simple terms, **nodes** are points on the maps (in latitude and longitude) as in the next image of a well documented college in New Delhi.



A **way** on the other hand is a ordered list of nodes, which could correspond to a street or the outline of a house. Here is an example of school in New Delhi which can be found as a way in OSM.



The final data element is a **relation** which is also an ordered list containing either nodes, ways or even other relations. It is used to model logical or geographic relationships between objects.

This can be used for example for large structures as in the restaurant which contains multiple polygons to describe the amenity.

Loading Data from OpenStreetMap with the Overpass API

Now we'll take a look how to load data from OSM. The Overpass API uses a custom query language to define the queries. It takes some time getting used to, but luckily there is Overpass Turbo by Martin Raifer which comes in handy to interactively evaluate our queries directly in the browser. Let's say we want to query nodes, ways and relations for all amenities in New Delhi, then our query looks like this:

```
  /*This has been generated by the overpass-turbo wizard.
The original search was:
“amenity= * in "New Delhi"” */
[out:json][timeout:25];
// fetch area “New Delhi” to search in
{{geocodeArea:New Delhi}}->.searchArea;
// gather results
(// query part for: “amenity=*”
  node["amenity"](area.searchArea);
  way["amenity"](area.searchArea);
  relation["amenity"](area.searchArea););
// print results
out body;
>;
out skel qt;
```

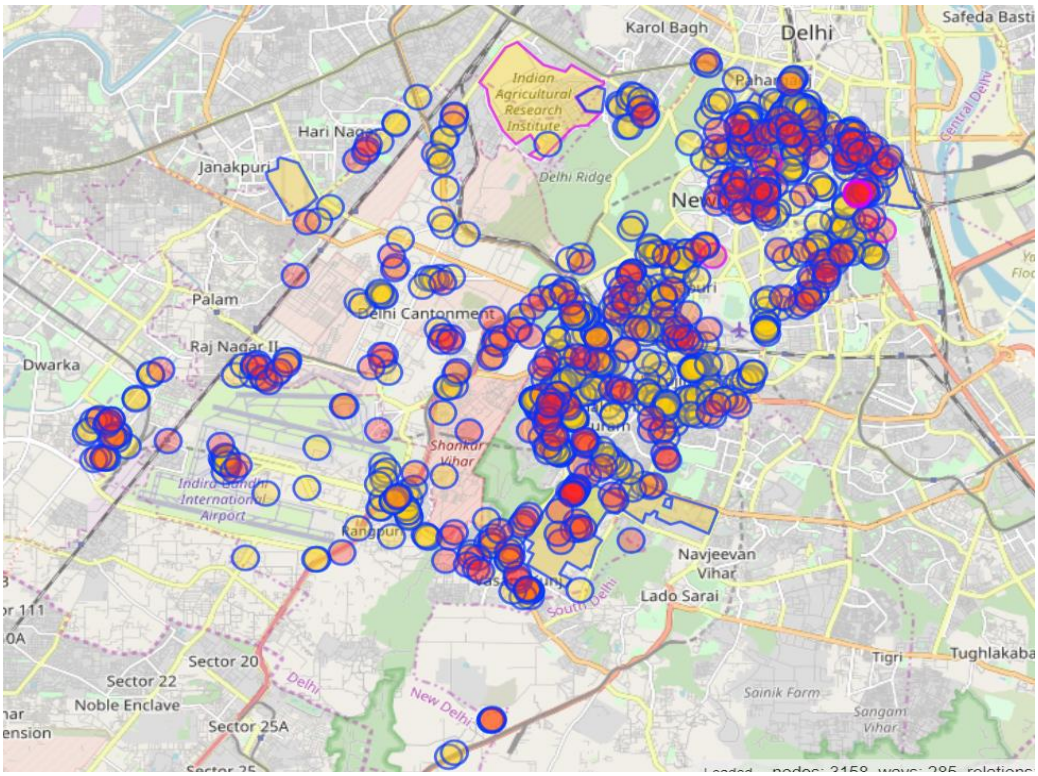
where each statement in the query source code ends with a semicolon. This query starts by specifying the component we want to query. There are different options to filter by tag. There

is a variety of tags to choose from, one common key is amenity which covers various community facilities like cafe, restaurant or just a bench. To have an overview of most of the other possible tags in OSM take a look at the OSM Map Features .

Here our ‘key’ is amenity. And the ‘values’ related to the ‘key ” amenity are as below:

- **Values**
  - Sustenance(bar,bbq,cafe,fast_food,food_court,ice_cream,pub,restaurant etc)
  - Education(college, library, university, school etc)
  - Transportation(parking, fuel, taxi etc)
  - Financial(atm, bank etc)
  - Healthcare(hospital,pharmacy,clinic,nursing_home etc)
  - Entertainment, Arts & Culture(casino,cinema,theatre,fountain etc)
  - Others(bench,toilets,police,clock etc)

Below is the map obtained after writing query in overpass turbo:



Now let’s export the data and download it as a GeoJSON file .And later we will read this GeoJSON file with the help of geopandas library in jupyter notebook for further clustering process. Later for visualization I have also converted POI data geoJSON file to shapefile (ESRI) with the help of QGIS software.

2. Understand how spatial location data works

Geographic Information System(GIS) concepts

- **Spatial vector data types**

The two primary types of spatial data are vector and raster data in GIS:

i. Vector data

The three basic symbol types for vector data are points, lines and polygons (areas). Because cartographers use these symbols to represent real-world features in maps, they often have to decide based on the level of detail in the map.

POINTS

Vector points are simply XY coordinates. Generally, they are a latitude and longitude with a spatial reference frame.When features are too small to be represented as polygons, points are used. For example, you can't see city boundary lines at a global scale. In this case, maps often use points to display cities.

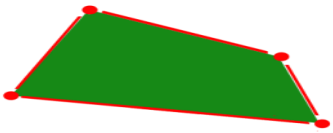
LINES

Vector lines connect each vertex with paths. Basically, you're connecting the dots in a set order and it becomes a vector line with each dot representing a vertex. Lines usually represent features that are linear in nature. For example, maps show rivers, roads and pipelines as vector lines. Often, busier highways have thicker lines than an abandoned road.

On the other hand, networks are line data sets but they are often considered to be different. This is because linear networks are topologically connected elements. They consist of junctions and turns with connectivity.If you were to find an optimal route using a traffic line network, it would follow set rules. For example, it can restrict turns and movement on one-way streets.

POLYGONS

When you join a set of vertices in a particular order and close it, this is now a vector polygon feature. In order to create a polygon, the first and last coordinate pair are the same. Cartographers use polygons to show boundaries and they all have an area.



ii. Raster data

Raster data is made up of pixels (also referred to as grid cells). They are usually regularly-spaced and square but they don't have to be. Rasters often look pixelated because each pixel has its own value or class.

Vector vs Raster: Spatial Data Types

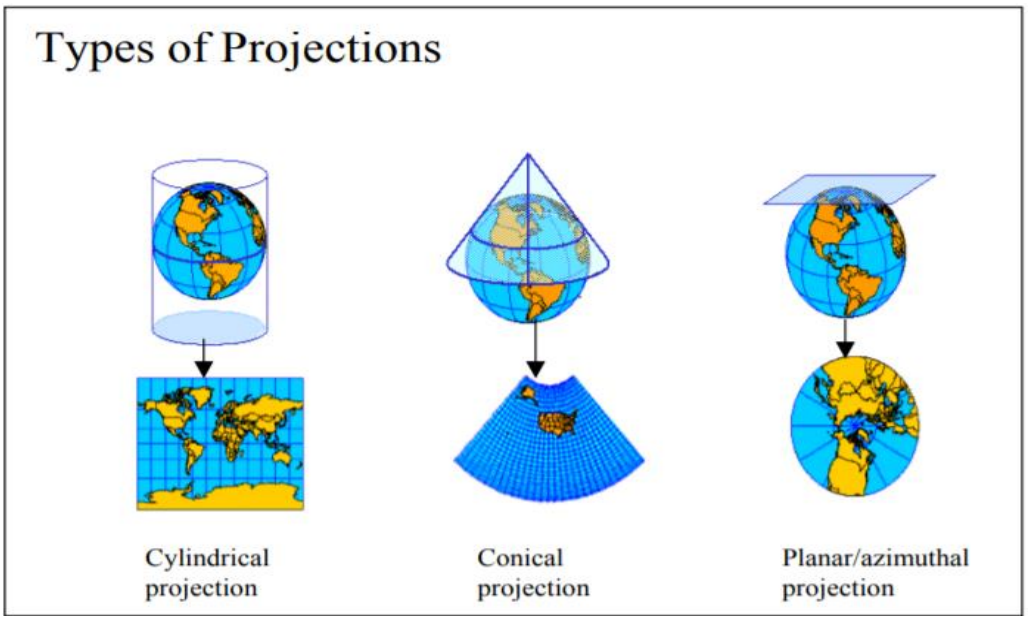
It's not always straight-forward which spatial data type you should use for your maps.

In the end, it really comes down to the way in which the cartographer conceptualizes the feature in their map.

- Raster data works with pixels. Vector data consists of coordinates.
- Vectors can scale objects up to the size of a billboard. But you don't get that type of flexibility with raster data
- Raster file size can result larger in comparison with vector data sets with the same phenomenon and area.

- **GIS projections**

There are many methods of map projections, since there are an infinite number of ways to project the 3-dimensional earth's surface on to a 2-dimensional planar surface. The 3-d to 2- d projections can be done to a plane or to the surface of a cone or cylinder leading to azimuthal, , conic or cylindrical projections respectively with many variations.



Depending on the scale and the agreeable tradeoffs with respect to distortions, a specific projection form is chosen. Different countries have adopted different standard projections at different map scales. In India, the polyconic projection is commonly used by Survey of India (SOI). All SOI toposheets are in the polyconic projection.

- **Vector GIS File Formats:**

Vector data is not made up of grids of pixels. Instead, vector graphics are comprised of vertices and paths.

The three basic symbol types for vector data are points, lines and polygons (areas). These GIS file formats vector data.

Extension	File Type	Description
Esri Shapefile	.SHP, .DBF, .SHX	The shapefile is BY FAR the most common geospatial file type you'll encounter. All commercial and open source accept shapefile as a GIS format. It's so ubiquitous that it's become the industry standard. But you'll need a complete set of three files that are mandatory to make up a shapefile. The three required files are: ▪ SHP is the feature geometry. ▪ SHX is the shape index position. ▪ DBF is the attribute data. You can optionally include these files but are not completely necessary. ▪ PRJ is the projection system metadata. ▪ XML is the associated metadata. ▪ SBN is the spatial index for optimizing queries. ▪ SBX optimizes loading times.
Geographic JavaScript Object Notation (GeoJSON)	.GEOJSON .JSON	The GeoJSON format is mostly for web-based mapping. GeoJSON stores coordinates as text in JavaScript Object Notation (JSON) form. This includes vector points, lines and polygons as well as tabular information. GeoJSON store objects within curly braces {} and in general have less markup overhead (compared to GML). GeoJSON has straightforward syntax that you can modify in any text editor. Webmaps browsers understand JavaScript so by default GeoJSON is a common web format. But JavaScript only understands binary objects. Fortunately, JavaScript can convert JSON to binary.

- **Open Source GIS software:**

a) QGIS (previously: Quantum GIS)
b) GRASS GIS
c) gvSIG Desktop
d) OpenJUMP
e) MapWindow GIS
f) udig
g) SAGA GIS
h) Spatial Manager Desktop

- **Spatial Clustering**

The core idea of statistical clustering is to summarize the information contained in several variables by creating a relatively small number of categories. Each observation in the dataset is then assigned to one, and only one, category depending on its values for the variables originally considered in the classification. If done correctly, the exercise reduces the complexity of a multi-dimensional problem while retaining all the meaningful information contained in the original dataset. This is because, once classified, the analyst only needs to look at in which category every observation falls into, instead of considering the multiple values associated with each of the variables and trying to figure out how to put them together in a coherent sense. When the clustering is performed on observations that represent areas, the technique is often called geodemographic analysis.

3. Figuring out a way of clustering POI data into commercial centers/markets

- **Geodemographic analysis**

The main intuition behind geodemographic analysis is to group disparate areas of a city or region into a small set of classes that capture several characteristics shared by those in the same group. By doing this, we can get a new perspective not only on the types of areas in a city, but on how they are distributed over space. In the context of our spatial data analysis, the idea is that we can group different amenities of New Delhi based on the type of amenity listed on the website. This will give us a hint into the geography of amenity(commercial_center) in the New Delhi.

Although there exist many techniques to statistically group observations in a dataset, all of them are based on the premise of using a set of attributes to define classes or categories of observations that are similar *within* each of them, but differ *between* groups. How similarity within groups and dissimilarity between them is defined and how the classification algorithm is operationalized is what makes techniques differ and also what makes each of them particularly well suited for specific problems or types of data. As an illustration, we will only dip our toes into one of these methods, K-modes, which is probably the most commonly used technique for statistical clustering.

- **K-modes clustering**

Introduced in 1998 by Zhehue Huang, k-modes provides a much-needed alternative to k-means when the data at hand are categorical rather than numeric.

Categorical here means that we can make a finite list all possible categories as column headings and then, for each row, indicate whether the category indicated is observed. For example, in clustering different commercial centers, POI data found around 73 unique amenities listed among around 1488 rows. Most rows listed at least one amenity, many listing more than that. If all the amenities are shown as as columns, the amenity can be marked with a "1" in the columns for amenity that the row has and "0" for those that does not. The below shows the sparseness of the resulting dataframe, with a "1" present in only one out of the 73 amenities:

	amenity_Ayurvedic Hospital	amenity_House	amenity_Netaji Nagar Market	amenity_Suvidha Market, Netaji Nagar	amenity_arts_centre	amenity_atm	amenity_bank	amenity_bar	amenity_bench
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0

5 rows × 73 columns

The distance metric used for K-modes is instead the Hamming distance from information theory. The Hamming distance (or dissimilarity) between two rows is simply the number of columns where the two rows differ.

$$d(X,Y)=\sum_{j=1}^{m}\delta(x_j,y_j)$$

where

$$\delta(x_j,y_j)=\begin{cases}0, & x_j=y_j\\ 1, & x_j\neq y_j.\end{cases}$$

The k-modes algorithm tries to minimize the sum of within-cluster Hamming distance from the mode of that cluster, summed over all clusters. Huang quoted, “if a dataset has m categorical attributes, the mode vector Z consists of m categorical values, each being the mode of an attribute.” In our amenity example, the mode of an attribute is either “1” or “0,” whichever is more common in the cluster.
The procedure is similar to k-means: a number of clusters (k) is chosen, and k cluster-mode vectors are chosen at random.

Step 1: Observations are assigned to the closest cluster mode by Hamming distance.
Step 2: New cluster modes are calculated, each from the observations associated with an previous cluster mode.
Step 3: Steps 1 and 2 are repeated until the cluster modes stabilize.

The Python k-modes library that I used is called kmodes and can be installed with pip install kmodes . It works analogously to scikit-learn’s k-means construct. The number of clusters to find is specified along with a heuristic for starting the clusters (‘Huang’) and n_init, which is the number of initial clusters to try in an attempt to find the lowest local minimum for the cost function.
After KModes clustering ,the clusters obtained is as below:

	amenity	clusters	geometry
0	restaurant	7	POLYGON ((77.19960 28.60221, 77.19960 28.60212...
1	courthouse	0	POLYGON ((77.23391 28.61606, 77.23344 28.61535...
2	post_office	0	POLYGON ((77.21243 28.62242, 77.21304 28.62207...
3	place_of_worship	0	POLYGON ((77.23927 28.60768, 77.23927 28.60730...
4	courthouse	0	POLYGON ((77.24011 28.62287, 77.24011 28.62302...
...	...	...	...
1483	fountain	0	POINT (77.23111 28.61253)
1484	fast_food	9	POINT (77.21877 28.62889)
1485	cafe	6	POINT (77.21867 28.62889)
1486	toilets	0	POINT (77.21896 28.63579)
1487	bank	0	POINT (77.22180 28.63466)

1488 rows × 3 columns

4. **Finding and labelling the most significant clusters, statistically and intuitively**.

These are the sample of clusters obtained using K-mode clustering:

	amenity	clusters
0	restaurant	7
1	courthouse	0
2	post_office	0
3	place_of_worship	0
4	courthouse	0
5	university	0
6	school	4
7	parking	2
11	school	4
12	school	4
14	parking	2
15	school	4
18	parking	2
24	parking	2
27	school	4
29	parking	2
35	parking	3
38	bus_station	8
40	fast_food	9
59	restaurant	7
90	bus_station	8
108	bus_station	8
129	cafe	6
130	cafe	6

K-modes clustering gave very good result. However further clustering can be done logically to minimize the number of clusters.
We know that there are 7 ‘values’ of ‘key’ amenity. Let’s further cluster these amenities to those ‘values’

- **Values**
  - Sustenance(bar,bbq,cafe,fast_food,food_court,ice_cream,pub,restaurant etc)
  - Education(college, library, university, school etc)
  - Transportation(parking, fuel, taxi etc)
  - Financial(atm, bank etc)
  - Healthcare(hospital,pharmacy,clinic,nursing_home etc)
  - Entertainment, Arts & Culture(casino,cinema,theatre,fountain etc)
  - Others(bench,toilets,police,clock etc)

Below is the visualization after clustering commercial centers intuitively to these 'values':

	amenity	clusters	geometry
0	restaurant	Sustenance	POLYGON ((77.19960 28.60221, 77.19960 28.60212...
1	courthouse	Others	POLYGON ((77.23391 28.61606, 77.23344 28.61535...
2	post_office	Others	POLYGON ((77.21243 28.62242, 77.21304 28.62207...
3	place_of_worship	Others	POLYGON ((77.23927 28.60768, 77.23927 28.60730...
4	courthouse	Others	POLYGON ((77.24011 28.62287, 77.24011 28.62302...
...	...	...	...
95	school	Education	POLYGON ((77.20637 28.62832, 77.20641 28.62905...
96	school	Education	POLYGON ((77.05558 28.56442, 77.05644 28.56546...
97	school	Education	POLYGON ((77.05999 28.56152, 77.06013 28.56169...
98	school	Education	POLYGON ((77.05663 28.56369, 77.05731 28.56325...
99	school	Education	POLYGON ((77.05977 28.56123, 77.05949 28.56141...

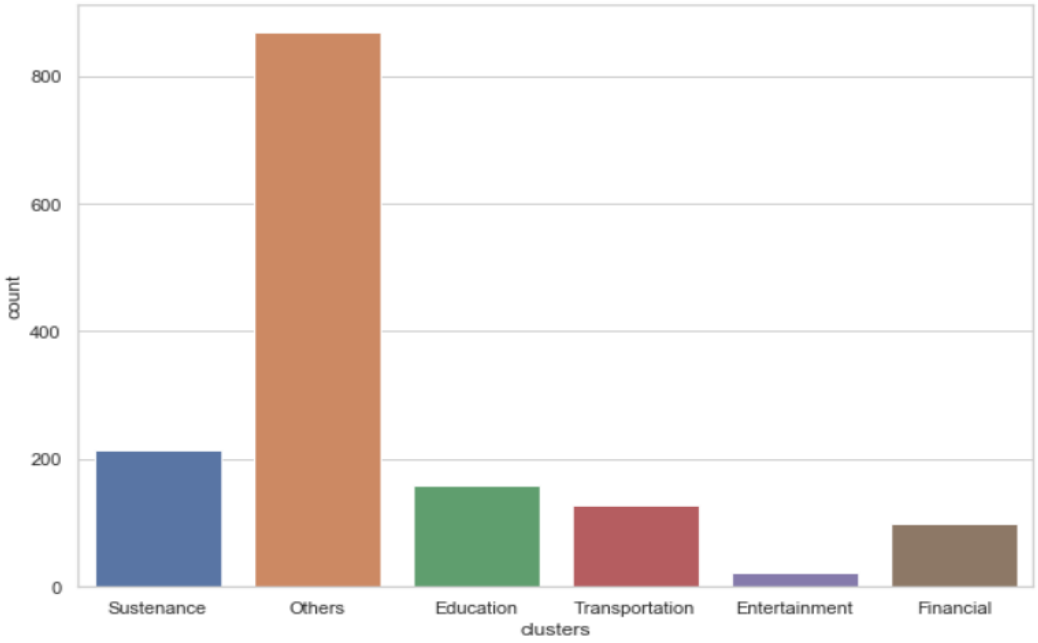


Fig: Countplot of different commercial centers

5. Visualize the resultant commercial centres/markets.

Below are some visualization obtained in Jupyter notebook

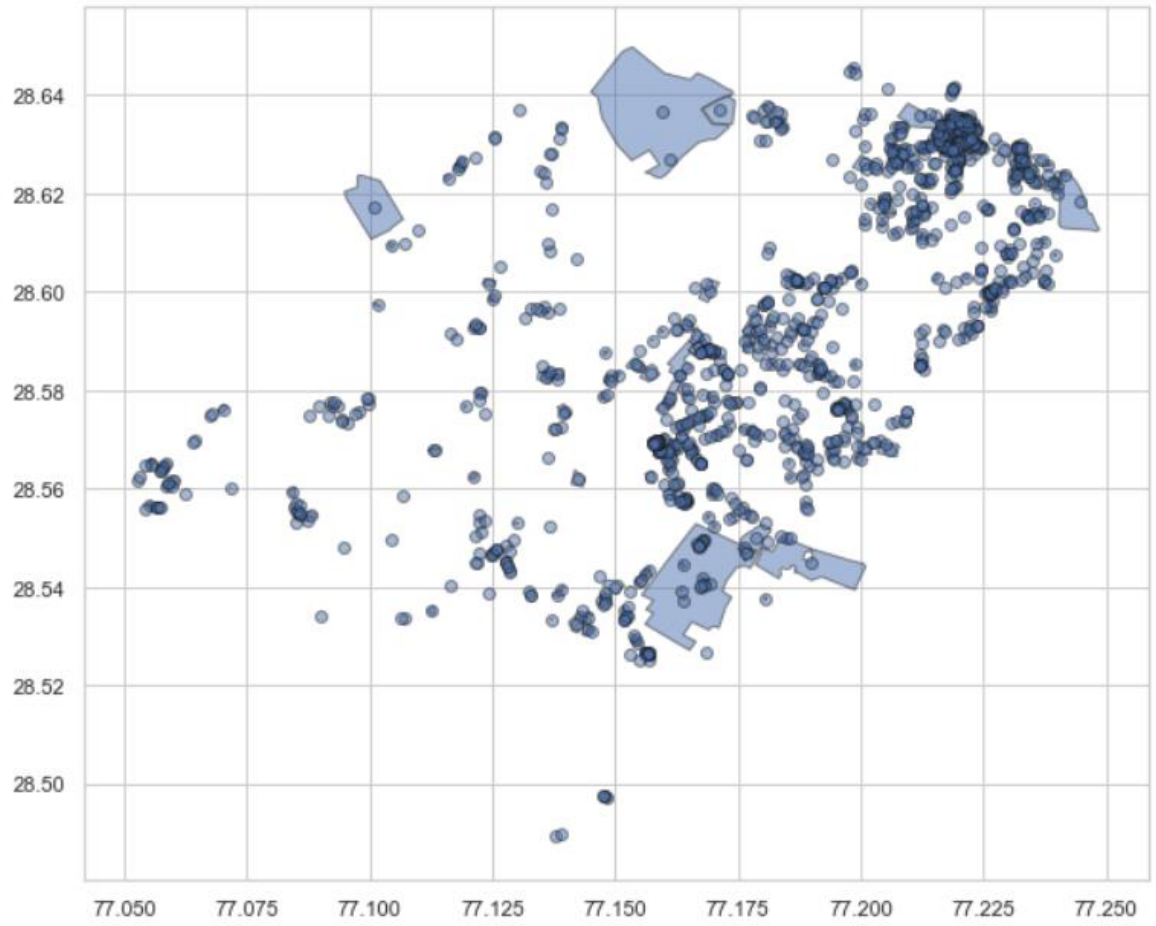
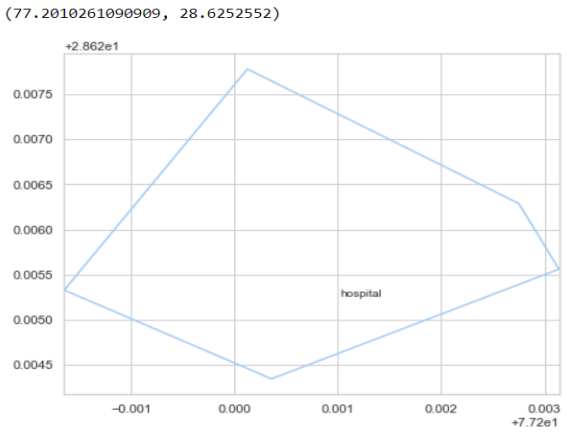
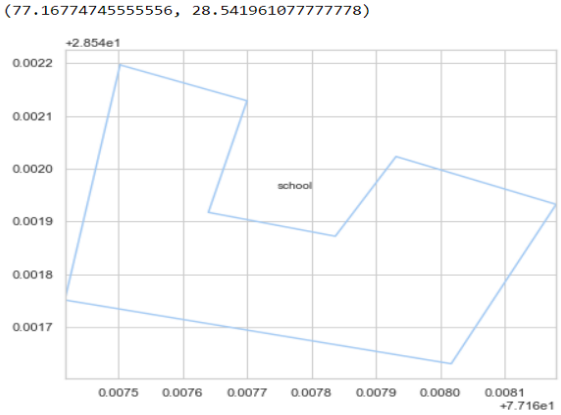


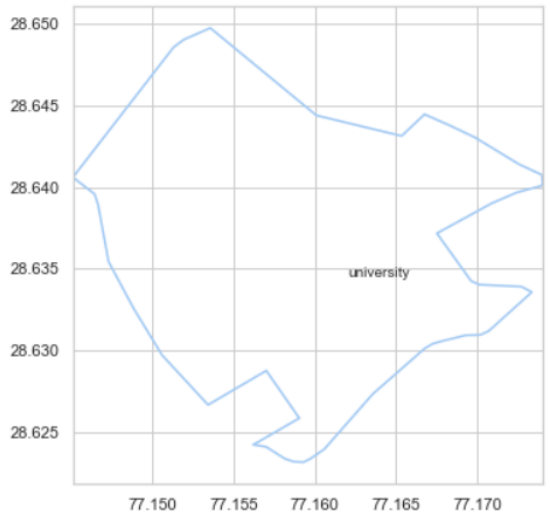
Fig: Point and Polygon plot with the help of geopandas library

Shape files for different amenities:

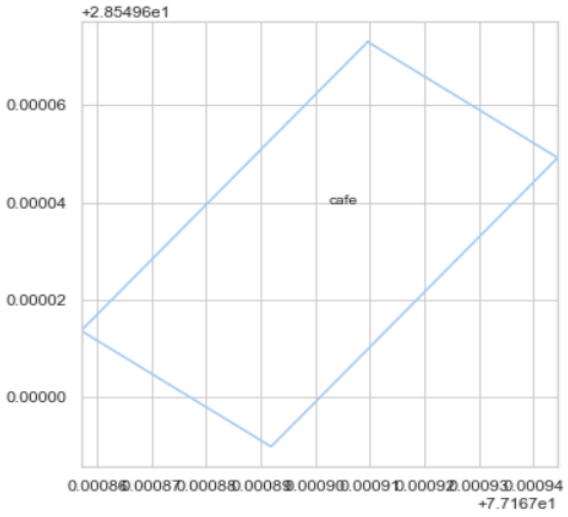




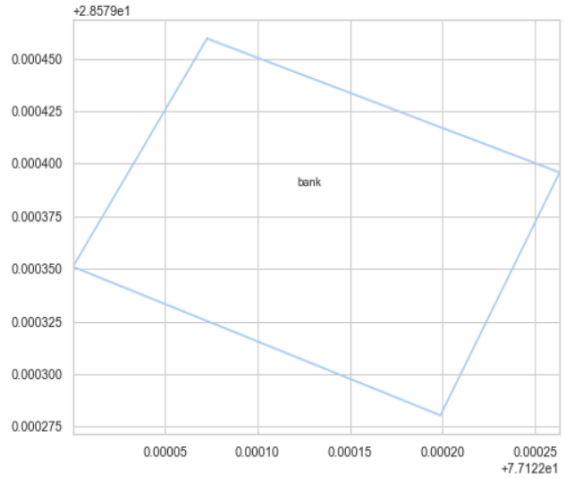
(77.16200191914893, 28.63451091489362)



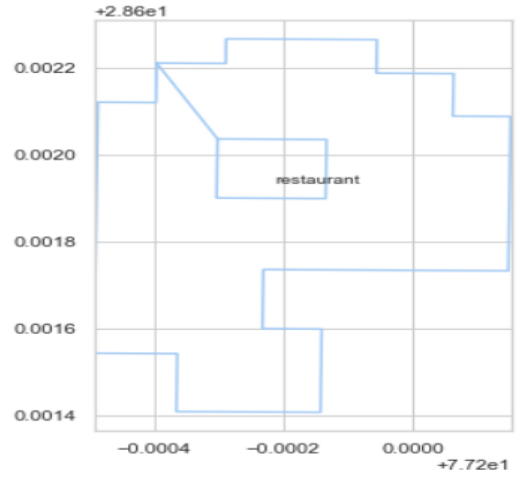
(77.16790252000001, 28.54963976)



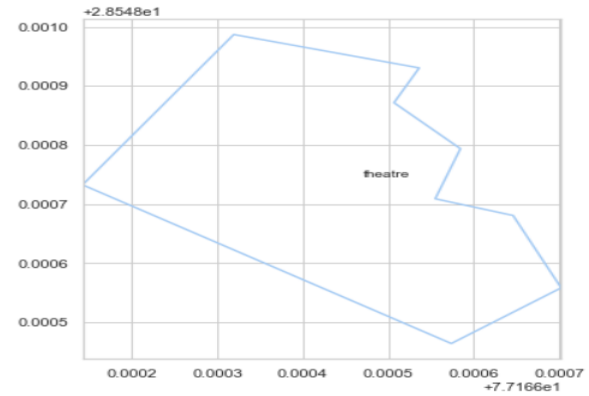
(77.1221216, 28.5793892)



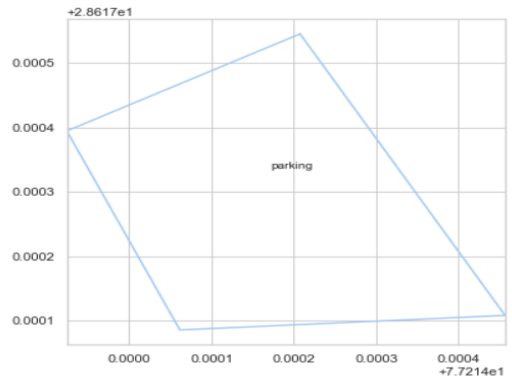
(77.19978806249999, 28.601934583333335)



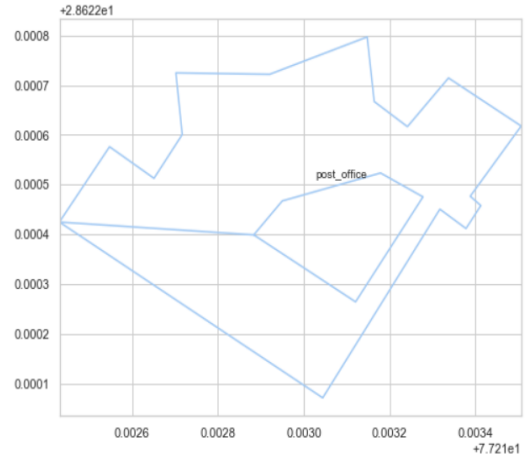
(77.16647087, 28.5487462)



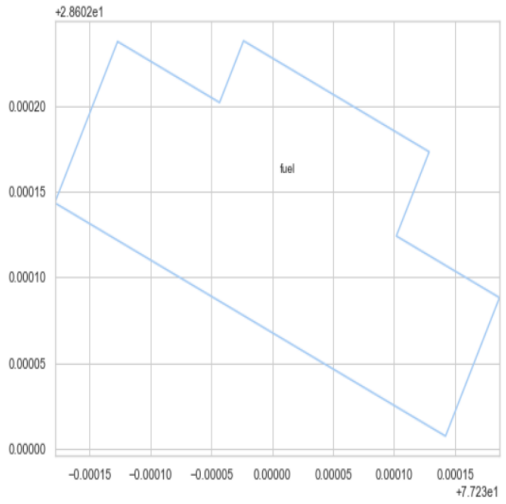
(77.21417146, 28.617335739999998)



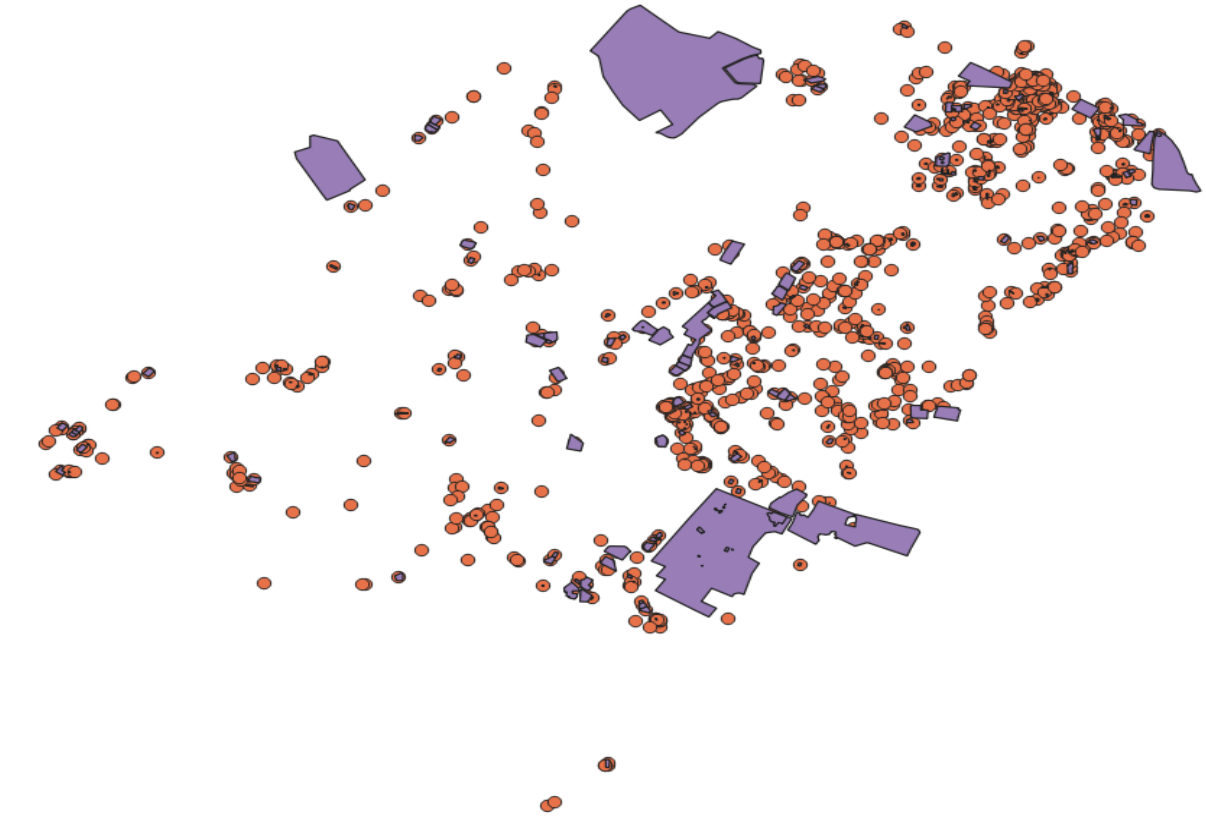
(77.21302747826087, 28.622512943478263)



(77.23000674444444, 28.602161166666665)



Visualisation in QGIS software



Visualization in Tableau