Where would you open a new shopping mall in Hamburg?

- Capstone Final Project

1. Background

As ‘the gateway to the world’, Germany’s second-largest city and biggest port – Hamburg, has been continuously attracting native Germans as well as tourists. We want to discover more about the diverse, vibrant neighborhoods with multicultural eateries and centers, to explore investment opportunities in real estate industry, in this project specifically, identifying optimal shopping mall locations.

2. Business Problem and Target Audience

The objective of this project is to use machine-learning methods (e.g. clustering) in Python to analyze and identify the optimal locations to open shopping malls in the city of Hamburg, Germany. The project might be useful and interesting for real estate developers and investors looking for solid investment in European market.

3. Data Requirements

We would need the following data for our project:

- List of neighborhoods in Hamburg (e.g. Alsterdorf, Blankenese, Hamm, etc) for scope definition: it will be extracted using web scraping from Hamburg Wikipedia pages;
- Latitude and longitude coordinates data of the above-mentioned neighborhoods: Python Geocoder package is used to get latitude and longitude coordinates;
- Venue data for the neighborhoods, which particularly includes shopping mall data, (e.g. location, tips, and categories): venue data is used for clustering and will be extracted with Foursquare API.

4. Methodology

In this section, we will briefly talk about the methods we used for the project, including web scraping, Python Geocoder, Foursqaure API and k-means clustering with code examples.

4.1 Web scraping

Web Scraping is the process of downloading data from websites and extracting valuable information from that data. One of the many packages we could use while doing data scraping is Beautiful Soup. Beautiful Soup is a python package for parsing HTML and XML documents and extracting data. To perform web scraping, we also need urllib to connect the webpage.

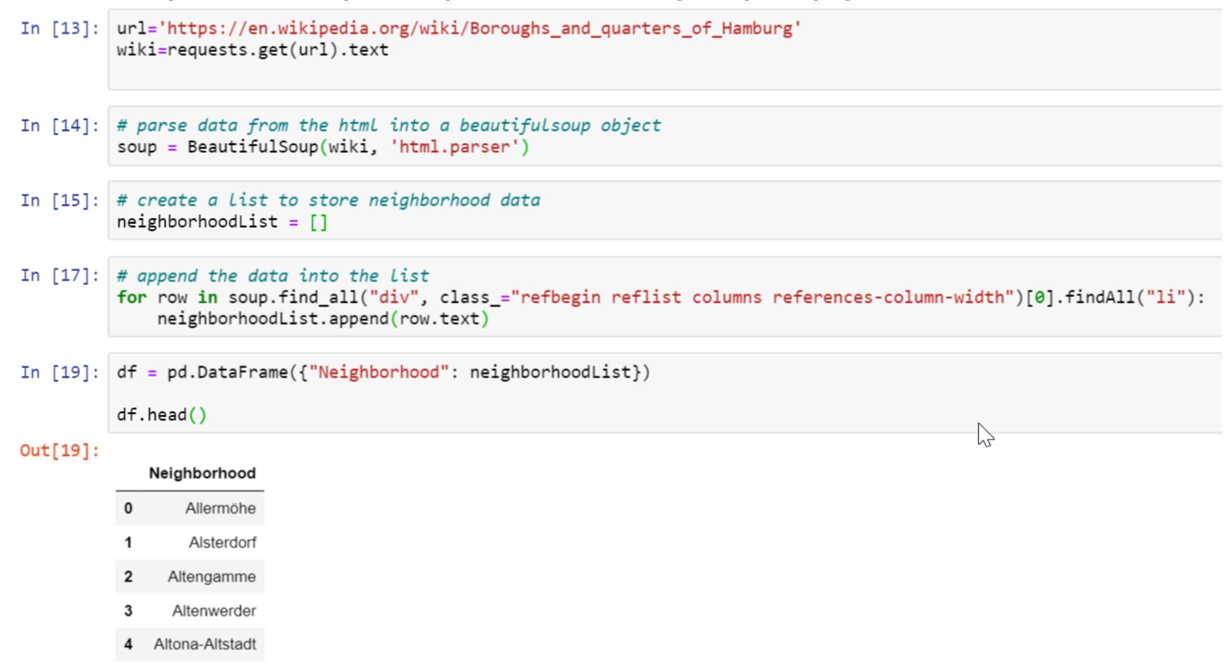
Here is a short guide to web scraping

Step 1: Identifying ranges of tags. Right-click anywhere on the webpage and go to “inspect” and select the element that we want to inspect. Each div tags has corresponding data that we want to obtain, and what we need to do is to use BeautifulSoup to go to the section, and then find all the data content

Step 2: We want to loop over our web scraper to every single div tags under this section and the corresponding data under the sub-category, which can be found in <li class = "list-item">.

Step 3: Nesting all the loops. We want to construct a nested loop for the web scraper to iterate every single list that contains our desired data.

Fig 1. Code Example: Scrap data from Hamburg Wikipedia page into a DataFrame



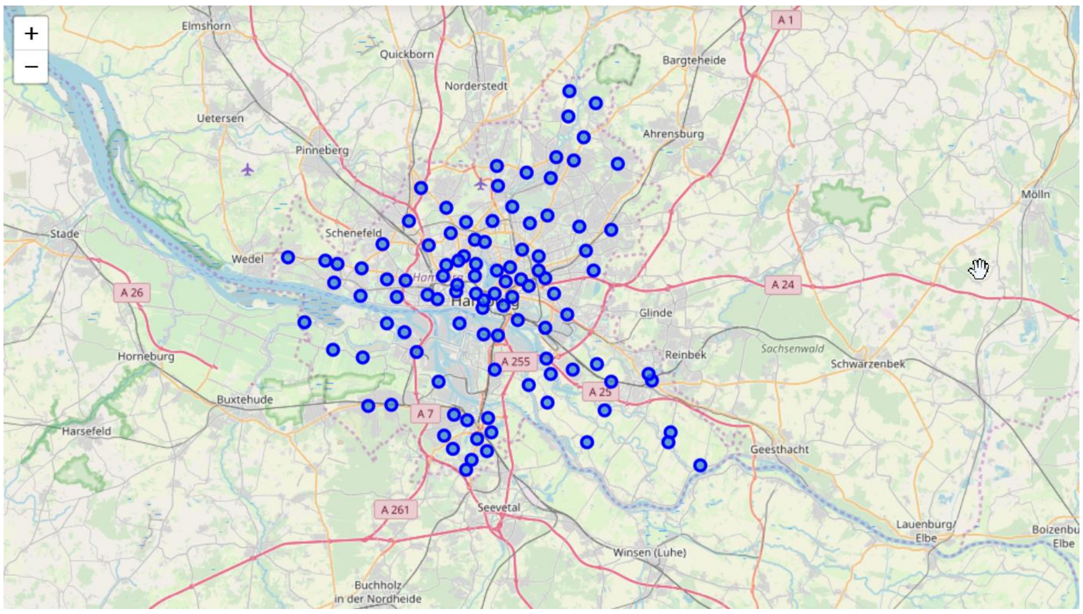
```
In [13]: url='https://en.wikipedia.org/wiki/Boroughs_and_quarters_of_Hamburg'
         wiki=requests.get(url).text

In [14]: # parse data from the html into a beautifulsoup object
         soup = BeautifulSoup(wiki, 'html.parser')

In [15]: # create a list to store neighborhood data
         neighborhoodList = []

In [17]: # append the data into the list
         for row in soup.find_all("div", class_="refbegin reflist columns references-column-width")[0].findAll("li"):
             neighborhoodList.append(row.text)

In [19]: df = pd.DataFrame({"Neighborhood": neighborhoodList})

         df.head()

Out[19]:
```

	Neighborhood
0	Allermöhe
1	Alsterdorf
2	Altengamme
3	Altenwerder
4	Altona-Altstadt

4.2 Python Geocoder

Geocoding is the computational process of transforming a physical address description to a location on the Earth’s surface (spatial representation in numerical coordinates). Luckily, Python library geopy can do the geocoding for us and make it easy to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources.

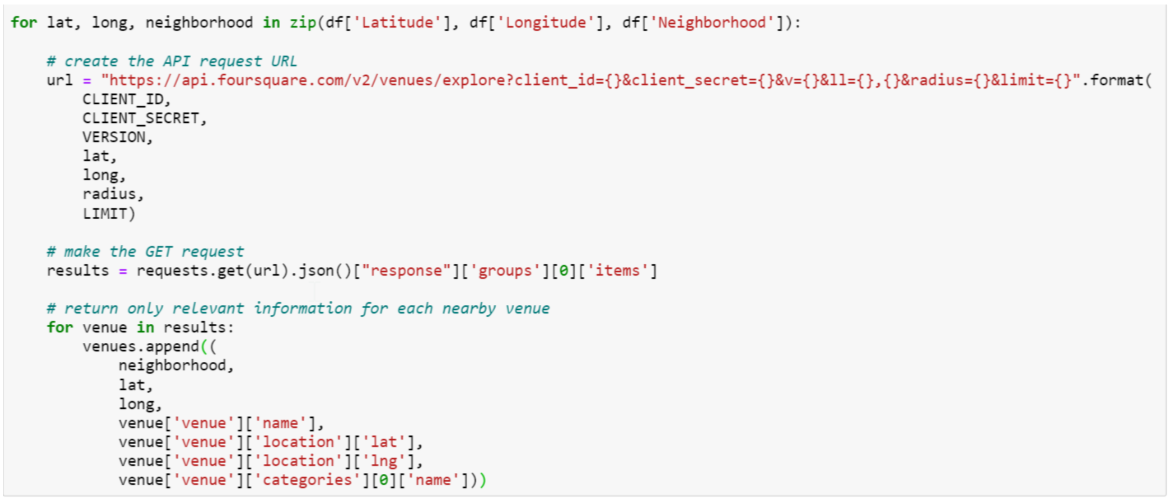
Fig 2. Creation of Hamburg map using latitude and longitude values



4.3 Foursquare API

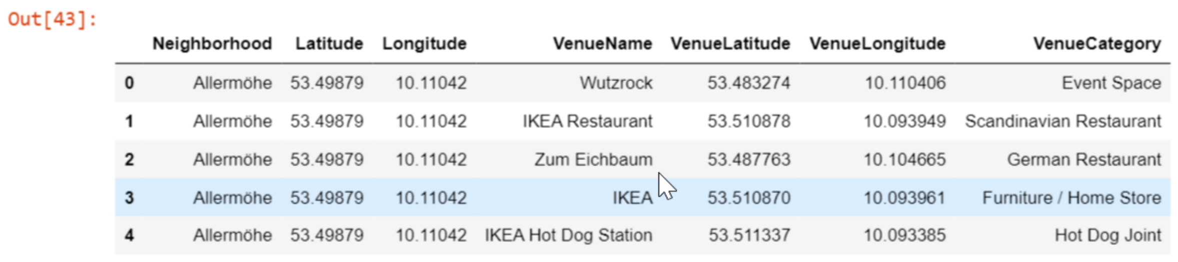
Foursquare is a social location service that allows users to explore the world around them. The Foursquare Places API provides location based experiences with diverse information about venues, users, photos, and check-ins. The API supports real time access to places, Snap-to-Place that assigns users to specific locations, and Geo-tag. JSON is the preferred response format. For example, when we make the call to the database, we will get a JSON file of the trending venues that are nearby. In the JSON file, for each trending venue, we get mostly its name, unique ID, location, and category.

Fig 3. Code Example: Create the API request URL and make the GET request



```
for lat, long, neighborhood in zip(df['Latitude'], df['Longitude'], df['Neighborhood']):

    # create the API request URL
    url = "https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}".format(
        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']))
```

Fig 4. Output Example: Venue List DataFrame



Out[43]:

	Neighborhood	Latitude	Longitude	VenueName	VenueLatitude	VenueLongitude	VenueCategory
0	Allermöhe	53.49879	10.11042	Wutzrock	53.483274	10.110406	Event Space
1	Allermöhe	53.49879	10.11042	IKEA Restaurant	53.510878	10.093949	Scandinavian Restaurant
2	Allermöhe	53.49879	10.11042	Zum Eichbaum	53.487763	10.104665	German Restaurant
3	Allermöhe	53.49879	10.11042	IKEA	53.510870	10.093961	Furniture / Home Store
4	Allermöhe	53.49879	10.11042	IKEA Hot Dog Station	53.511337	10.093385	Hot Dog Joint

4.4 K-means clustering

K-means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The k-means algorithm works as follows:

1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
   - Compute the sum of the squared distance between data points and all centroids.
   - Assign each data point to the closest cluster (centroid).
   - Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

5. Result and Discussion

Based on the frequency of occurrence for "Shopping Mall", we can see that the neighborhoods are grouped into 3 clusters:

- Cluster 0 (in red): neighborhood with moderate number of shopping malls
- Cluster 1 (in purple): neighborhood with very few number of shopping malls
- Cluster 2 (in green): neighborhood with rather high number of shopping malls

Hamburg is famous for its numerous canals and bridges, and we can clearly see that the neighborhoods near the canal have none to very few shopping malls; in the northern part of the city, where there are more and more residents living, are neighborhoods with more shopping malls.

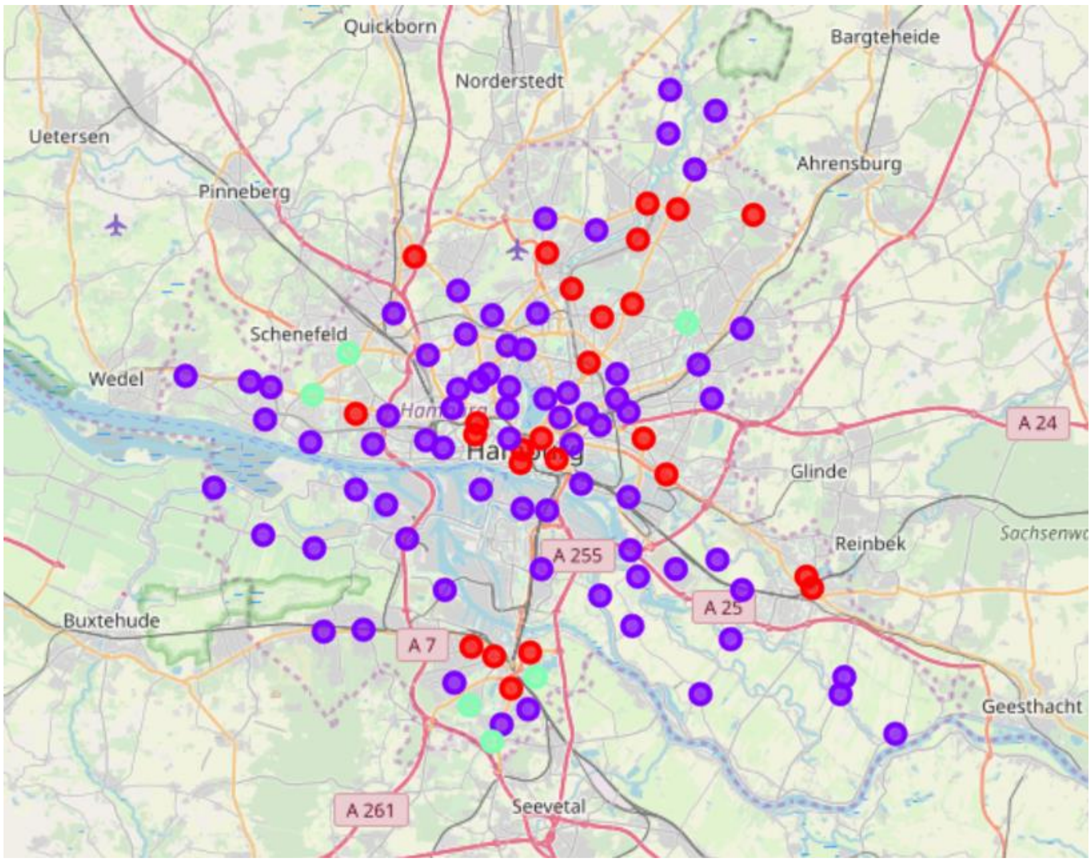
What's more interesting to note is that, cluster 2, boundary areas with other cities tend to have more shopping malls.

As the Hamburg city is getting more and more crowded and expensive, we can expect that more city residents will move further away towards the boundary areas with other cities. Thus, those border areas with less shopping malls could be an interesting option as new shopping mall locations.

Hamburg city center areas, for example Hafen City are already concentrated with a number of shopping malls; the land is expensive, rent is high and competition is fierce. Therefore, we won't recommend to invest in those "traditional" center areas in cluster 0.

Along the rivers and lakes areas as depicted in cluster 1, where there are only a few to none shopping malls, we can consider investing in high-end boutique shopping malls. As the residents in those neighborhoods are mostly high-income earners, they would have the desire, as well as the purchasing power to buy more exclusive and unique items.

Fig 5. K-means Clustering Visualization



6. Conclusion

In this project, we went through the process of identifying business problems, specifying data requirement, data preparation and cleaning, data exploration and data analysis using machine-learning methods. With the help of online open sources like Python libraries, Foursquare APIs and so on, we could perform our analysis and provide recommendations to the relevant audience. As discussed in part 5, we would recommend to invest in high-end boutique shopping malls in cluster 1 neighborhood that are along the lakes and rivers. At the same time, boundary areas to other cities could also be a sustainable and long-term investment opportunity, while the city is expanding, and more city residents are moving further away from the city center.

7. References

https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a
https://developer.foursquare.com/docs/api
https://en.wikipedia.org/wiki/Boroughs_and_quarters_of_Hamburg