CMPT353 Final Report

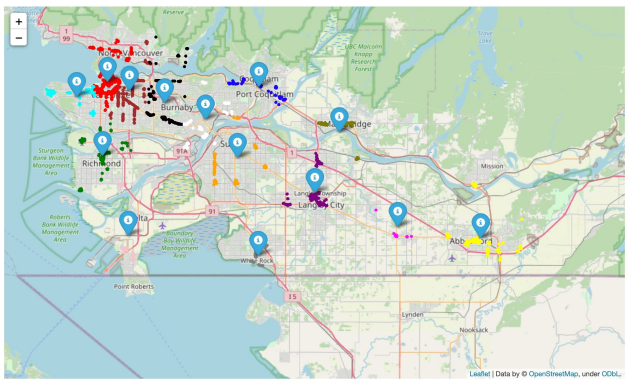
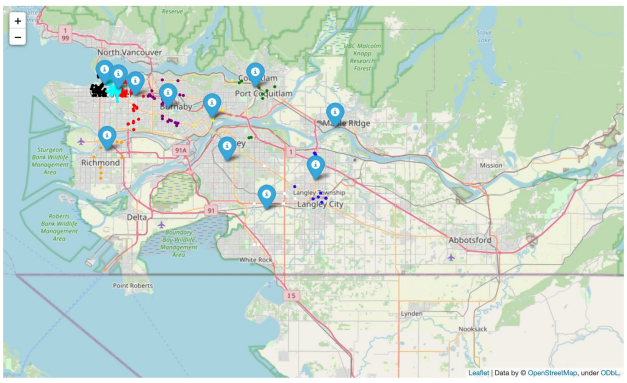
Preface

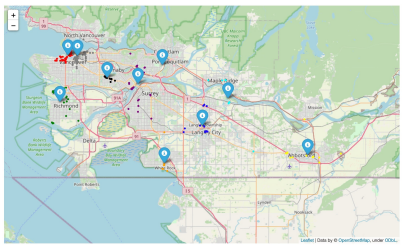
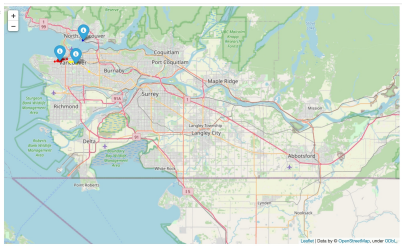
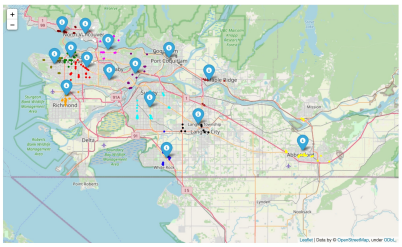
- Our team project is divide by 3 different parts. Each part is an individual analysis of problem.
- Travelling Route: Yifan Zuo
- Chain and Non-chain Restaurant Identification: Siyuan Wu
- Hotel choose: Shuang Wang

Part 1: Travelling Route

- Problem addressing:
  - `What is the most reasonable travel route around Canada Vancouver`?
  - A reasonable travel route is connected by several reasonable travel points.
  - A reasonable travel point should near by at least four most important amenities, another amenity is optional:
    - **food**: a travel point should near by lots of restaurants...tec.
    - **transportation**: a travel point should be transportation convenient and transportation relavent place.
    - **entertainment**: a travel point should be fun.
    - **shopping and banking**: a travel point should near by bank and shopping mall.
    - **living and storage** (optional): place to have rest such as hotel. luggage deposit.
- Data cleaning (data_cleaning.py):
  - Remove irrevalent amenities:
    - amenities such as fire_station, kindergarten...etc shouldn't be our consideration, those are irrelavent to our problem.
  - Group the remaining amenities and classify them as above five amenities.
    - **Food**: cafe, fast_food, bbq, vending_machine, restaurant, drinking_water, juice_bar, watering_place, water_point, food_court, ice_cream.
    - **transportation**: fuel, parking_entrance, bicycle_parking, parking, ferry_terminal, car_rental, car_sharing, bicycle_rental, seaplane terminal, car_wash, bicycle_repair_station, parking_space, taxi, bus_station, motorcycle_parking, boat_rental, loading_dock, car_rep, motorcycle_rental.
    - **entertainment**: pub, public_building, cinema, theatre, bar, arts_centre, fountain, nightclub, stripclub, gambling, bistro, playground, spa, events_venue, internet_cafe, social_centre, gym, park, biergarten, leisure.
    - **shopping and banking**: atm, bank, bureau_de_change, marketplace, atm;bank, money_transfer, shop|clothes.
    - **living and storage**: bench, shelter, luggage_locker, lounge, housing co-op, storage.

    - Remove the rows which name is Nan.
        - If the name is Nan, then traveller cannot search these places and therefore those information are useless.
    - Remove outlier:
        - In order to make later clustering center point more clear, we can use `LocalOutlierFactor()` to remove some outliers base on the latitude and longitude of the points.
- Data Written (data_cleaning.py):
    - After data cleaning process, the cleaned data will be written into cleaned_data.csv, and five separate data: food.csv, transportation.csv, entertainment.csv, living_stroage.csv, shopping_banking.csv.
    - All processed data in the directory named 'data'.
- Clustering (map_cluster.ipynb, cluster.py):
    - Use python `folium` to present the map.
    - Use `sklearn.cluster Kmean` to do the clustering.
        - the n_cluster is tuned to the best value: food: 14 clusters, transportation: 11 clusters, entertainment: 10 clusters, shopping and banking: 15 clusters, living and storage: 3 clusters.
        - The center point of each cluster(big blue point) can be calculate by mean of lon and lat of that cluster points
        - Different color represent diffierent cluster. The blue big point represent the center point of that cluster.

Food	Transportation
	

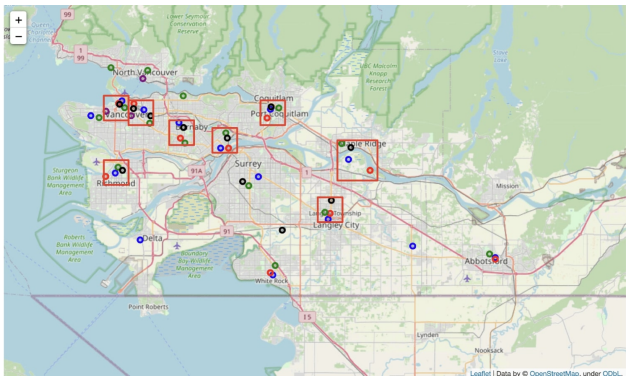
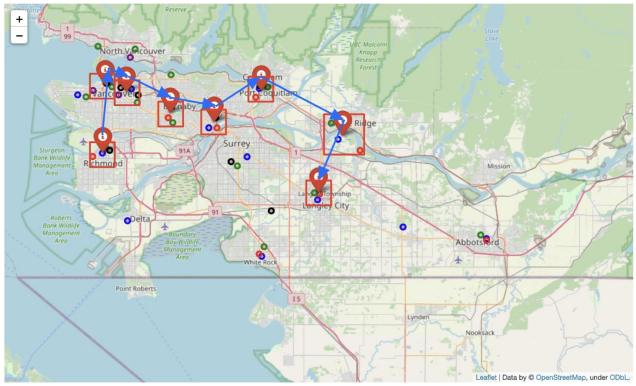
Entertainment	Living_storage	Shopping_banking
		

- Analysis and Conclusion (analysis.ipynb):
  - Amenity grouping: (groups image)
    - Base on the above five different clustering, we now present the all center points of the five amenities.
    - Blue point: food, green point: banking and shopping, red point: entertainment, black point: transportation, purple point: living and storage.
    - The red square box represent a group of those different center points. Every red square should conatin at least one of blue, red, gren black center point meaning that travel point should near by at least four most important amenities. The purple center point is optional. We mentioned early.
    - We can see the image below, there are 8 qualified groups(red box).
  - Travel route: (travel route image)
    - We can calculate the center point among those groups(red big point) shown on the below image. The big red point is actually our travel points.
    - Then the problem become TSP problem, we can find the shortest path among those points, and the path is shown as the blue arrow.
    - The latitude and longitude travel route(red big points):

      `(49.17918741164552, -123.13875940395383)` → `(49.2734536397419, -123.13272939827512)` → `(49.26395053943205, -123.0885422157221)` → `(49.2346415642415, -122.99360688849441)` → `(49.221810611933634, -122.8984494528381)` → `(49.26632186591236, -122.79747464328304)` → `(49.20270020618055, -122.620861450625)` → `(49.12279338831816, -122.67356765116651)`

    - The rough location travel route:

      `Richmond: Golden Village` → `Vancouver: Granville Bridge` → `Vancouver: Main Street-Science World` → `Burnaby: Deer Lake Park` → `New Westminister: Fraser Cemetery` → `Port Coquitlam` → `Maple Ridge: Langley Bog` → `Langley City: Langley Township`

Grouping Image	Travel Route
	

- Limitation:
    - The grouping techique(not clustering) is not supported by statistical test for roughly the same mean of longitude and latitude, we only can conclude by see the graph.
    - we need to tune the n_cluster parameter for the Kmean clustering, in some cases it's not perfect.

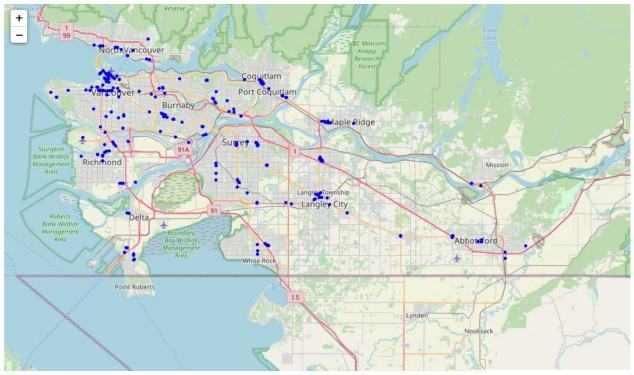
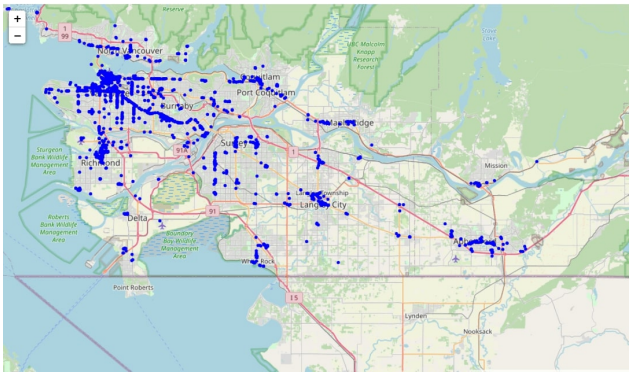
Part 2: Chain and Non-chain Restaurant Identification

Part 2: Chain and Non-chain Restaurant Identification and Analysis(Siyuan Wu)

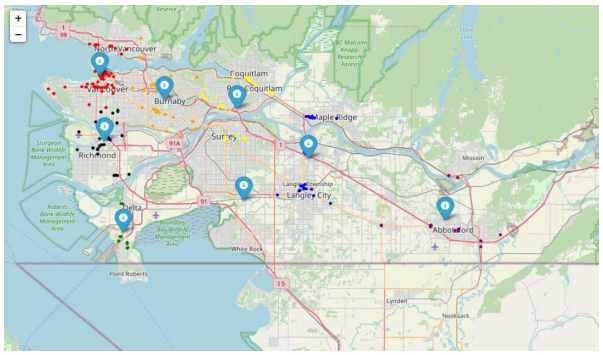
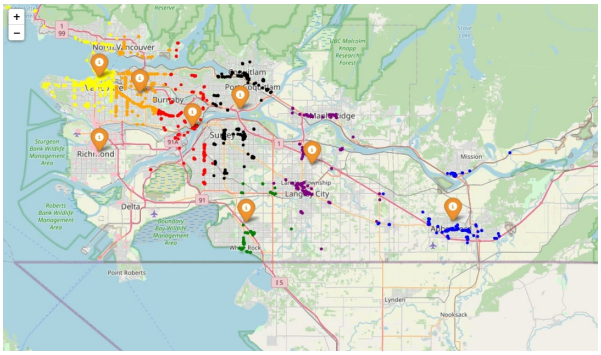
- Problem addressing:
    - `The problem: I feel like there are some parts of the city with more chain restaurants: is that true? Is there some way to find the chain places automatically and visualize their density relative to non-chains?`
    - For this topic, we are only looking into data where attribute equals to 'restaurant'.
    - A chain restaurant is defined to be having more than 3 data entries in the restaurant category or have the name listed on Wikipedia' chain restaurant' page.
- Data cleaning (data_forming.py)
    - All data cleanings are achieved through manipulating pandas libraries, so all data cleanings are performed on the spot.
    - Remove data entries where attribute 'name' is equal to Nan.
        - If the name is Nan, it would probably be a false entry. In addition, users cannot find the restaurant from the data, so it should be excluded.
    - Remove duplicate data entries.
        - Some restaurants have more than 3 data entries in the data and have their name listed on the Wikipedia page.
        - Duplicate data would reduce the accuracy of clusters, so duplicate data entries are being removed using `.drop_duplicates()`.
    - Manual cleaning
        - Some scraped data have branch names included in their names, preventing them from matching the OSM data.
        - These names don't have a general pattern that can be clean with regular expression. Thus, manual cleaning is needed.
        - Examples of manual cleaning include: 'A&W(Canada)' to 'A&W', 'Baton Rouge (restaurant)' to 'Baton Rouge', 'BeaverTails restaurant' to 'BeaverTails'
- Data gathering(data_ forming.py)
    - The definition of a chain restaurant is to have more than 3 data entries in the dataset. However, some chain restaurants are having less or equal to 3 in size, so data from Wikipedia are being used to chain restaurants from this category.
    - The first list of restaurant names is generated by retrieving `.count()` larger than 3 on the restaurant name.

- The second list of restaurant names is generated from data scraping from Wikipedia using `BeautifulSoup`. Since the OSM data is based on Vancouver, only USA and Canada chain restaurant pages are being used. The data source page is listed below.
  - The U.S chain restaurant: https://en.wikipedia.org/wiki/List_of_restaurant_chains_in_the_United_States
  - Canada chain restaurant: https://en.wikipedia.org/wiki/List_of_Canadian_restaurant_chains#Major_chains
- Two lists then concatenate to form a final list of names, then performing a join operation on the list and OSM data to obtain the chain restaurant data. All other restaurants that are not chain restaurants are classified into non-chain restaurants.
- Finally, chain restaurant data is written into 'data/chain.csv', and non-chain restaurant data is written into 'data/non_chain.csv.'

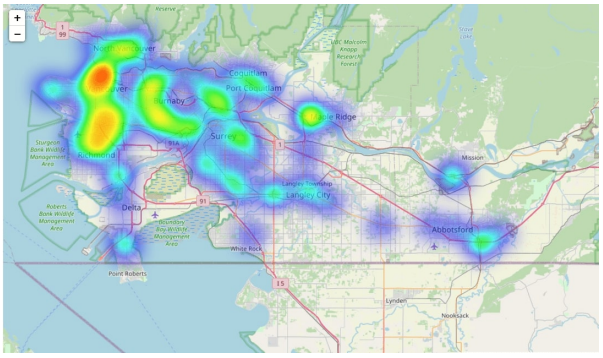
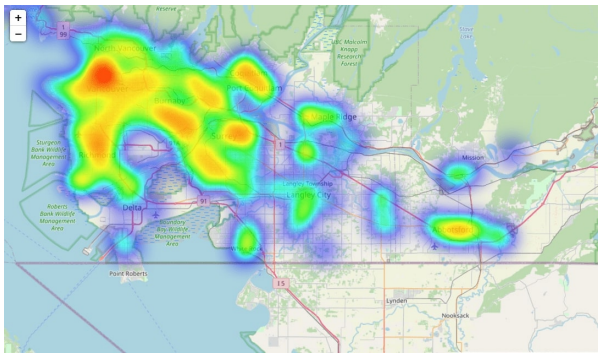
- Analysing techniques and Visualization (map_cluster.ipynb, cluster.py, stat_test.py)
  - The map_cluster.ipynb and its helper code cluster.py are adopted from the groupmate, Yifan Zuo's work. The code employed python `folium` package to present the map.
  - The visualization of chain and non-chain restaurant data points on map are below.

Chain Restaurant Distribution	Non-chain Restaurant Distribution
	

  - Cluster is done by using `sklearn.cluster Kmean`
    - The number of clusters is tuned to 8 to provide the best result.
    - The large blue and orange indicator indicates the center of each cluster and is calculated by the mean of lon and lat of that cluster.
    - The visualization of chain and non-chain restaurant data clusters and their centers on map are below.

Chain Restaurant Cluster	Non-chain Restaurant Distribution
	

- For more straightforward visualization, heatmaps on the chain and non-chain restaurants are also provided below.

Chain Restaurant Heatmap	Non-chain Restaurant Heatmap
	

- After data points are grouped into clusters, lat, lon and their designated cluster number are written into 'data/cl_data' and 'data/ncl_data' for chain and non-chain restaurants.
- In stat_test.py, data from 'cl_data' and' ncl_data' are converted to the contingency table for `scipy`'s stats package. Finally, the code will perform a chi-square test on the relationship between locations and chain/non-chain restaurants.

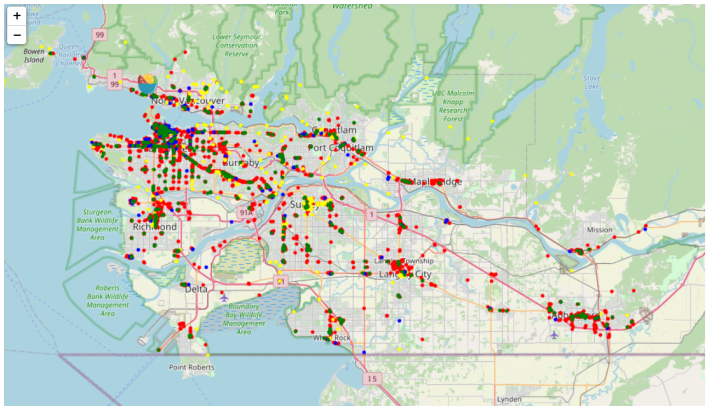
- Findings
  - The null hypothesis for the chi-square test is that chain/non-chain restaurant density has is not affected by geographic location.
  - The outcome of the p-value for chi-square is `p-value = 3.696108751238701e-34` which s less than 0.05, so we can conclude that geographic location has some effect on chain/non-chain restaurant density.
  - From the heatmap of both data, we can observe relatively more non-chain restaurants in Burnaby and Surrey area.
- Limitations
  - Some chain restaurant names are still not being classified into chain restaurants because of their slightly different name. Therefore, more comprehensive manual data cleaning is needed.
  - Drawing out the boundary for each cluster may be helpful to see which area has more or fewer chain restaurants.

- A density map instead of a cluster map may be more straightforward to show their density.

Part 3: Hotel Choose

- Problem addressing:
  - `If I was going to choose a hotel (or Airbnb), where should it be? What places have good amenities nearby? Will it effect on price of hotel?`
  - On this issue, we think hotels should be built in places with perfect facilities. First of all, these places should meet the basic food needs. The transportation should be convenient. At the same time, there should be a certain number of entertainments places around.
- Find habitable place by suing amenity:
  - Data cleaning:

    We selected some amenities. Food: cafe, restaurant, fast food Transportation: parking, bus station. Entertainment: pub, cinema. Shopping: atm, bank, marketplace
  - Analysis:
    - red: food, yellow: transportation, blue: entertainment, green: shopping



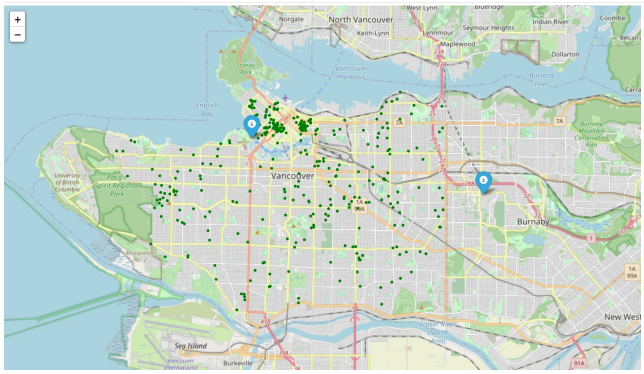
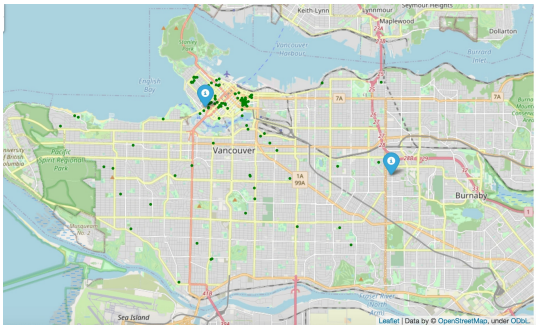
    - Conclusion:

      From the visualization of all amenities, it clearly show that most of amenities are located at the center of each city, like Downtown Vancouver, Richmond, Surry and Coquitlam.
- Airbnb dataset:

  Base on another large dataset on Airbnb from “Inside Airbnb adding to debate”, this data is summary information and metrics for listings in Vancouver and compiled at 06 July 2021. For this question, we keep name, price, minimum night, review to find best hotel for users.

  - Data cleaning:

    From the public dataset of Airbnb, including reviews and listings For Listings: Find the listings with minimum nights < 3 For Reviews: Find reviews in 2020 year houses with reviews per month > 1

data cleaning	Airbnb's price > 200
	

- Conclusion:

  As this Airbnb data was in Vancouver, we could find there are good place for people to choose hotel since there are lots of amenities in Downtown and Chinatown, most of Airbnb with price > 200 dollars per night there. But if people do not want to pay too much on hotel, they can try some place are far away from center of city.

- Limitation:

  There are lots of companies that provide service for vacation rentals and tourism activities, but we only using Airbnb dataset to analysis.

Project Experience Summary

- Yifan Zuo:

  Travelling route around Canada Vancouver: Using bigdata tool such as panda, numpy and clustering technique to determine the best travel route in Canada BC province, and using folium to visualize the results. *July 2021 – August 2021*

- Siyuan Wu:

  OSM data extraction and analysis: Using data analysis tools such as pandas and numpy, combining with web scraping tools such as beautifulsoup to obtain the desired data. Then, utilize statistical tools such as scipy to perform clustering. Finally, using folium to visualize the result. *July 2021 – August 2021*

- Shuang Wang:

  OSM data extraction and analysis: Collected Airbnb data from the Internet and used python function to extract date and name from the raw data. Then using python library folium to implement map visualization. Came up with the idea of using Airbnb data to solve hotel problem. *July 2021 – August 2021*