<CMPT353> Project Report: OSM,PHOTOS,AND TOURS

Group ‘Betabase Gurus’
Jin Hyun (Sam) Kim: jhk29@sfu.ca

Zhi Xuan (Otto) Hu: ohu@sfu.ca

Wing Chun (Justin) Siu: wcsiu@sfu.ca

Introduction

Open Street Map (OSM) is an open source maps project that collects and shares world's map data for free. For this project we acquired the processed OSM dataset that only includes places in the Metro Vancouver area. We came up with three problems to analyze with this data:

1. Can we better categorize the places in the dataset using ML methods?
2. With our OSM data, can we plan a good tour path?
3. Are there some parts of the city with more chain restaurants than non-chain restaurants? Are non-chain restaurants more spread out than chain restaurants?

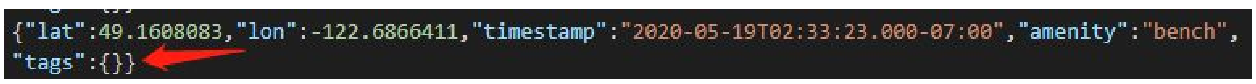
1. Categorizing Places Using ML methods

There are tens of thousands of places in the OSM dataset, ranging from restaurants, hotels to water fountains and bike stands. However, they are not categorized in a more general way. The "amenity" attribute tells the exact use of the place (e.g., post_box, restaurant, etc.). What if a user wants to search a more broad term like "food" or "health"? We can use ML methods to categorize the places into broader categories.

1.1) Data Preprocessing

1. FIrst in the process, we identified all the unique amenities that showed up in the dataset. We categorized them manually into 6 brackets.
    - **Education** - all kinds of schools
    - **Health** - 'doctors','pharmacy','dentist','Pharmacy'...
    - **Utility** - 'toilets', 'post_box', 'telephone','vending_machine'...
    - **Public Service** - 'courthouse', 'townhall', 'fire_station', 'police'...
    - **Community** - 'community_centre', 'park', 'marketplace'...
    - **Leisure** - 'cinema', 'theatre','restaurant'...
2. The data we have has a lot of text based columns. We had to extract numeric features from them.

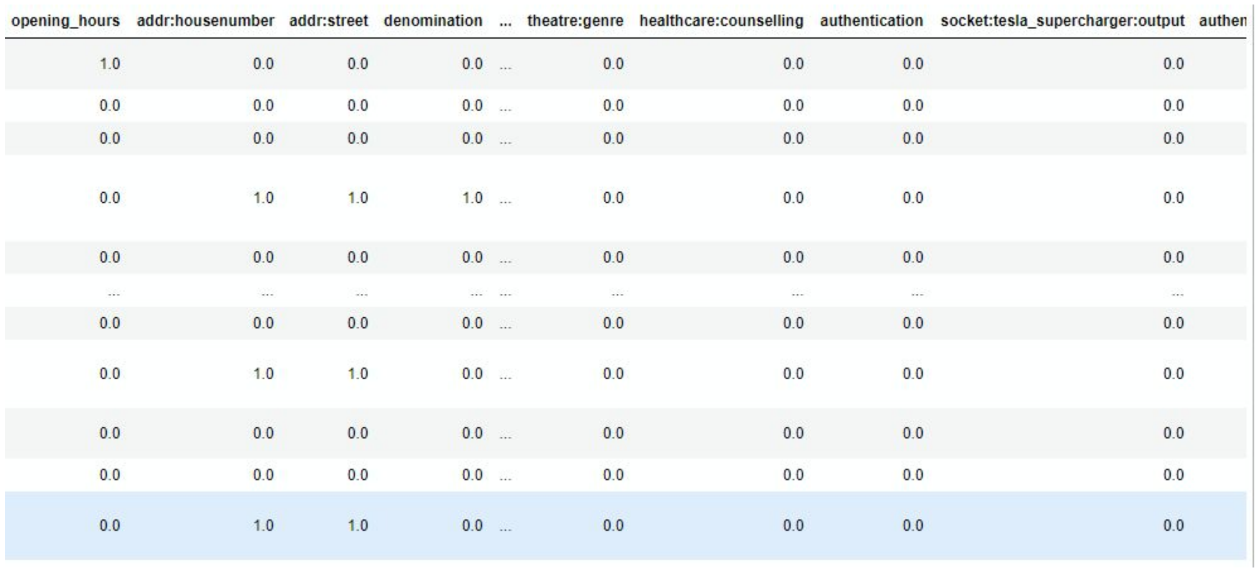
    1) We counted the number of tags attached to each place and made a new column called 'tagCount'. This can be helpful to identify utility from the other categories since utility map items usually have less or no tags. Other categories would have a lot more tags.



```
{"lat":49.1608083,"lon":-122.6866411,"timestamp":"2020-05-19T02:33:23.000-07:00","amenity":"bench",
"tags":{}}
```

    2) Then we extracted all the keys of the tags. We did not keep the values since they are too specific and too hard to convert into numeric values. We ended up with

something like the picture below, where 1 means the place has a tag and 0 means no tag. These columns will help identify between different tagged places. For example, only a “charging station” or similar would have the tag “**socket:tesla_supercharger:output”.**



opening_hours	addr:housenumber	addr:street	denomination	...	theatre:genre	healthcare:counselling	authentication	socket:tesla_supercharger:output	authen
1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
0.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	
0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	
0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
0.0	1.0	1.0	0.0	...	0.0	0.0	0.0	0.0	
0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
0.0	1.0	1.0	0.0	...	0.0	0.0	0.0	0.0	

3) Lastly we dropped the columns that are not useful in the categorization process. Things like “timestamp”, “name” and “amenity” itself. “Timestamp” is when the place is added to the dataset, which is utterly unrelated in categorizing the place. “Name” would be helpful sometimes but in most cases we found that the name is very short and does not convey category-related meanings. For example, “the reef” is a restaurant but it is hard to associate “the reef” with the leisure category. This could be a further improved task, in which we can use more advanced ML methods to associate the names. We can add these columns back after the model is trained, if needed. For the sake of using the ML methods they need to be dropped.

1.2) Data analysis

We end up with these columns for the ML algorithms to work with:
- longitude
- latitude
- tag count
- different tag availabilities…

Most of the categories tend to show up on some specific type of area. For example, leisure places are often located near city centers. Therefore, longitude and latitude can help greatly in categorization. After we categorize the dataset, we found the distribution as below. The utility items are the most common, followed by leisure places.

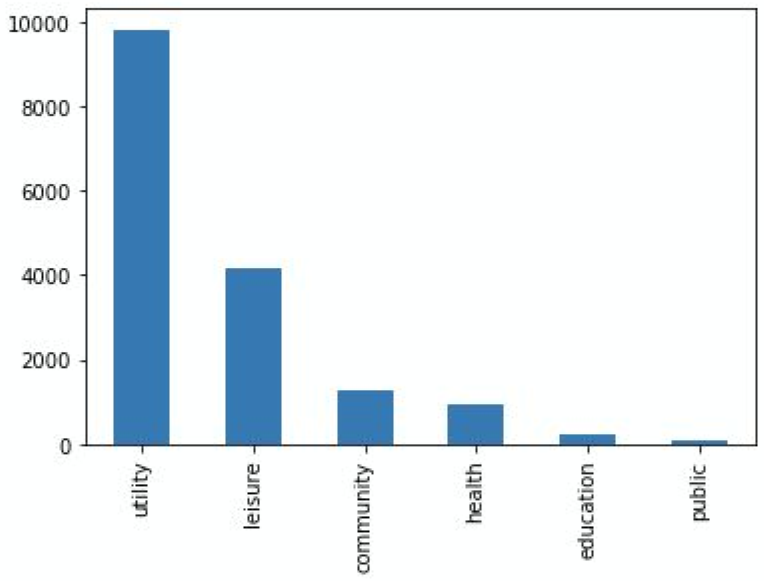


Figure 1.1. Categorized Dataset Distribution

We labeled all the places using different colors on the map. We can see blue (utility) is the most common color, followed by yellow (leisure). Blue and yellow also seem to appear quite closely together. This is reasonable since leisure places usually have a lot of utility items surrounding them. Blue and yellow also show up more in the city center areas. Some green points are nearby the blue and yellow dense areas. Community places show up around the residence area.

Utility - Blue **Leisure** - Grey **Community** - Orange **Health** - Green **Education** - Red **Public Service** - Purple

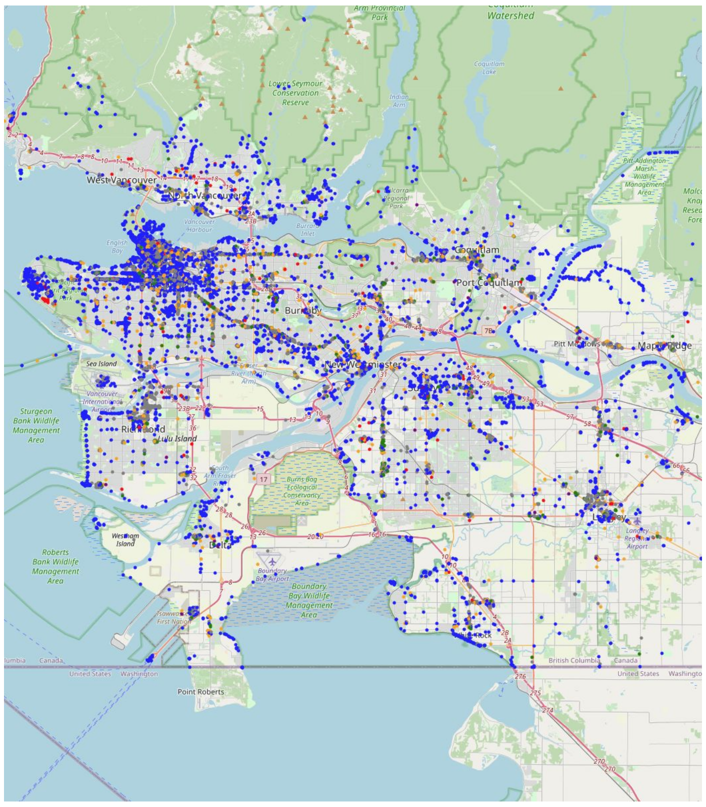


Figure 1.2. Map with labelled places

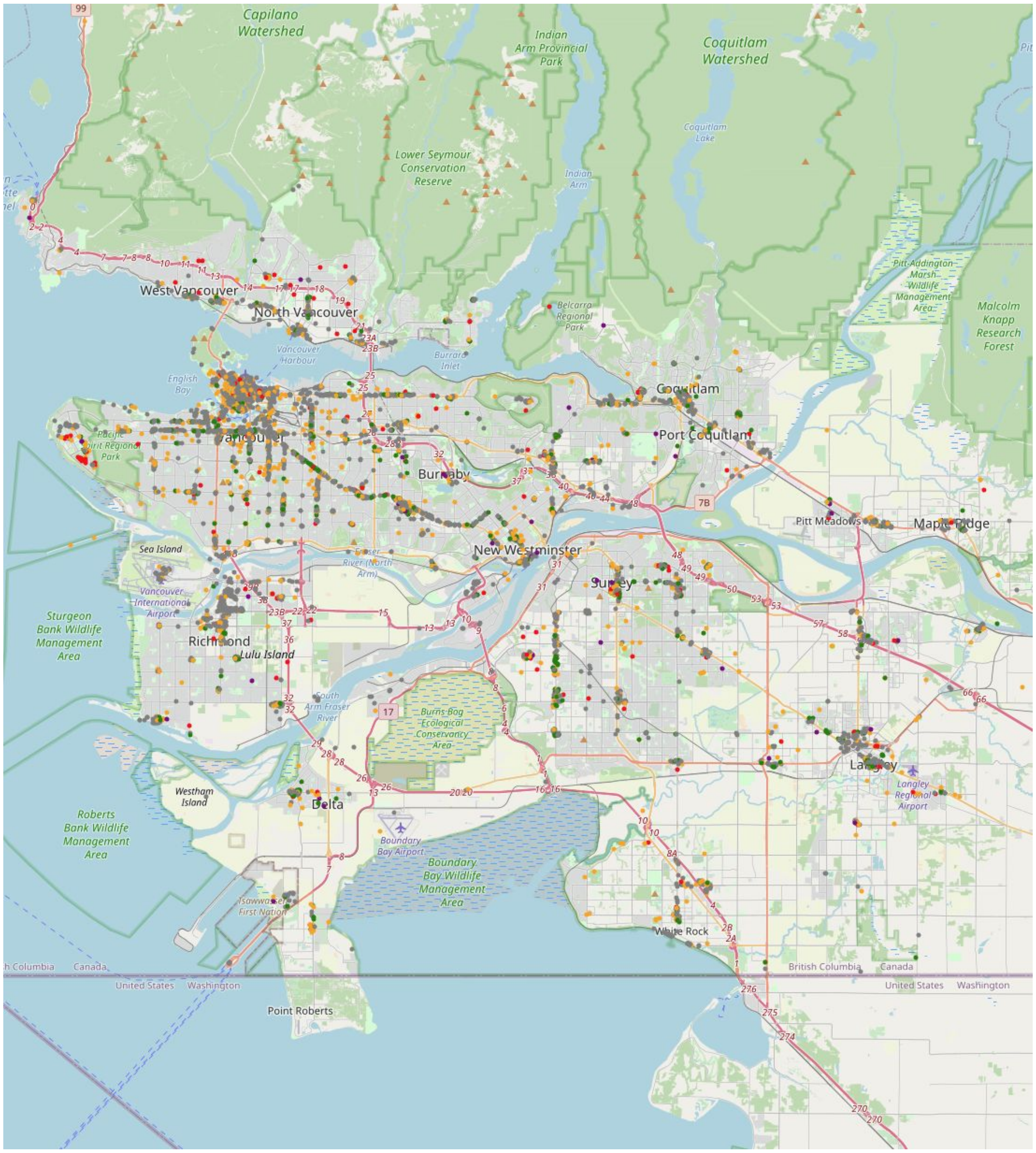


Figure 1.3. Map without utility items

To predict the category of each place, we split the dataset into x_train, y_train and x_valid, y_valid to train our models. We trained three models using three different classifier algorithms. The accuracy score is as below.

Classifier	**Gaussian Native Bayes**	**Knn nearest neighbour**	**Random forest**
Accuracy score	0.4477975176441957	0.8926746166950597	0.9123874422000486

The Gaussian Native Bayes model does the worst in our practice. We can see that Knn and random forest perform a lot better than Bayes algorithm. Knn takes the nearest places into consideration. Since the places under the same category are usually grouped together, the model can use the nearest neighbor's longitude and latitude to predict the category of this one. Random forest performs even better than Knn.

1.3) Conclusion

We can see the spread of all the places on the map. Because utility includes a wide range of everywhere items, utility items are all over the map. Leisure places, restaurants and pubs, show up in clusters, usually on the sides of the main roads. Community and Health places are less common, located more around residential areas.

With knn and random forest algorithms achieving around 90% accuracy, we can say that using the location and tags to predict the category is quite effective. We can change the categories to better fit different specific domains. For example, we can add a transportation category which includes all the bike rentals, car rentals and buses etc. Some categories, like the utility category, include perhaps too wide a range of items, it might be more helpful to further break down. Overall, we can easily use this categorization method in real-world use.

2.Recommend Tour Around Vancouver

From the provided OSM data, we get over 15000 different locations. We selected some of them to generate three tours with some interesting places that could be accessed by a bicycle, a bus, or a car.

2.1) Collect & Filter data

First, we filtered out some locations from OSM data by their amenity and arrange these locations into two groups, Food and Art Place :

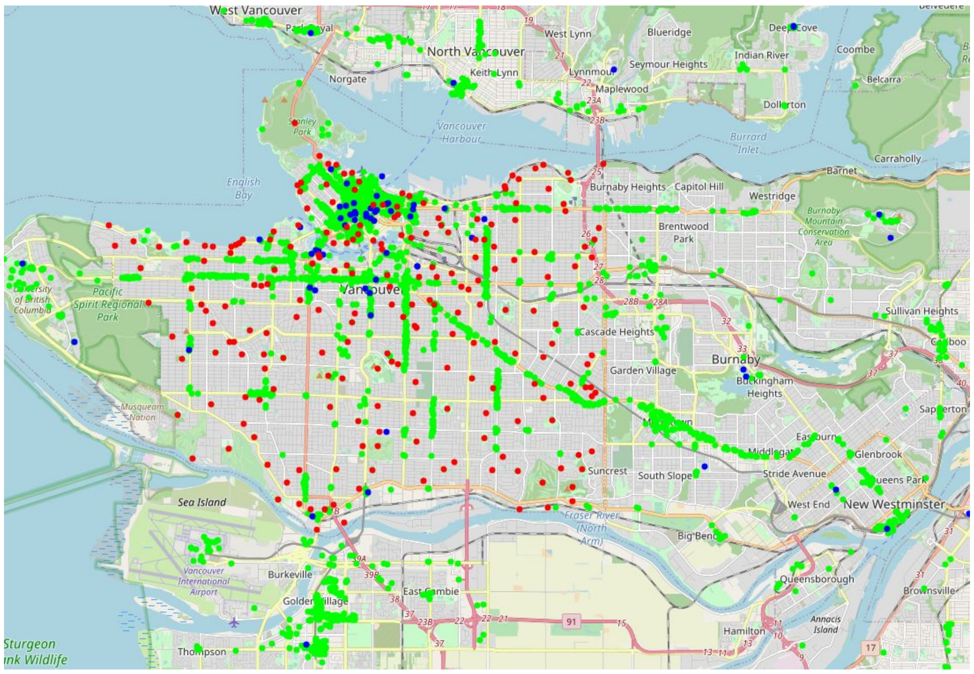
Group	Amenity
Food	Cafe , Ice cream, Restaurant
Art Place	Art centre, Cinema, College, Theatre, University

Upon finding out that there is only 1 location for amenity: ’park’ in our OSM dataset, we added more park locations from the Open Data Portal of the website – City of Vancouver [3] (parks.csv). Next, we joined these three groups together and got a total of 3923 locations which will be our “interesting place” around Vancouver to help us generate the tours later.

2.2) Analysis Data

Initially, we mapped all locations of the interesting places on the map by using Python’s folium library function where each colour represent a group of the place:

Green — Food **Blue – Art Place** **Red -- Park**



From the map, we can see that most of the interesting places are located around Vancouver Downtown. So, we can think that Downtown must be a location that the tour will most likely take place in.

However, there are too many locations for one tour. Consequently, we splitted all locations into groups that are near each other using KMeans [4] and DBSCAN [5]:

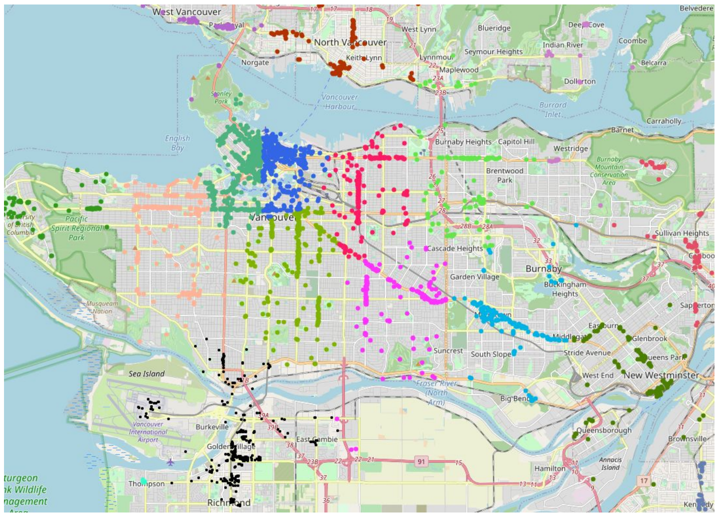


Figure 2.1: KMeans

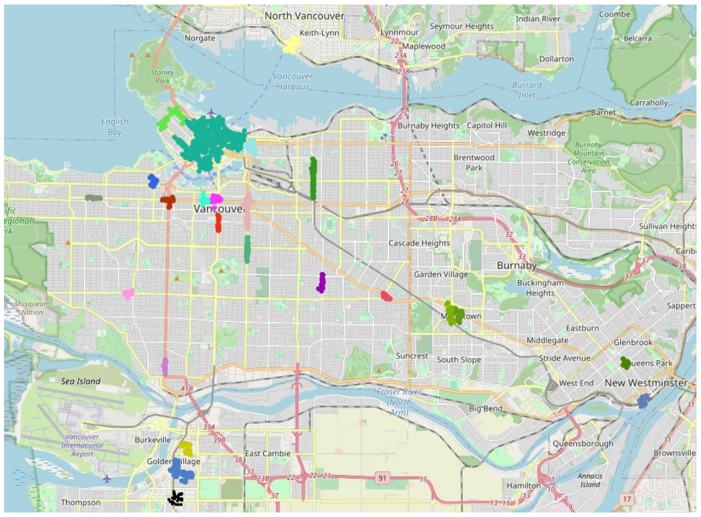


Figure 2.2: DBSCAN

From the two maps above, we can see that all locations are both well-grouped by two models. In Figure 2.1, the K-Means assigned a group to each location, helping us separate each location area. In Figure 2.2, the DBSCAN model also separated location to different area groups. It helped us find these area groups which are extremely concentrated and expands clusters from them. For our tour, we chose the area groups with DBSCAN since it filters out the noise location which is not close to together. It helped us form our tour so that we can pass more interesting places from one stop.

2.3) Creating Tour

After analyzing the data, we generated three tours around Vancouver that can be accessed by a bicycle, bus, or through driving.

2.3.1) Bicycle Tour

We found the bicycle route in Vancouver by using OpenStreetMap with Cycle Map Layer (Figure 2.3) [6]. We filtered out locations related to bicycle from OSM data by amenity: “bicycle_rental” & “bicycle_parking” and plot the map with the interesting places (Figure 2.4):

Marker Color for figure 2.4 (Red — Bicycle Rental   Blue – Bicycle Parking   Black – Interesting Place)

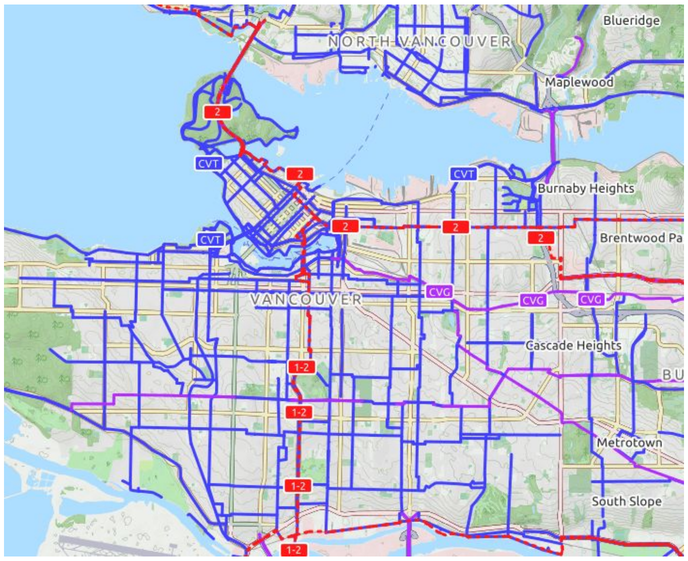


Figure 2.3

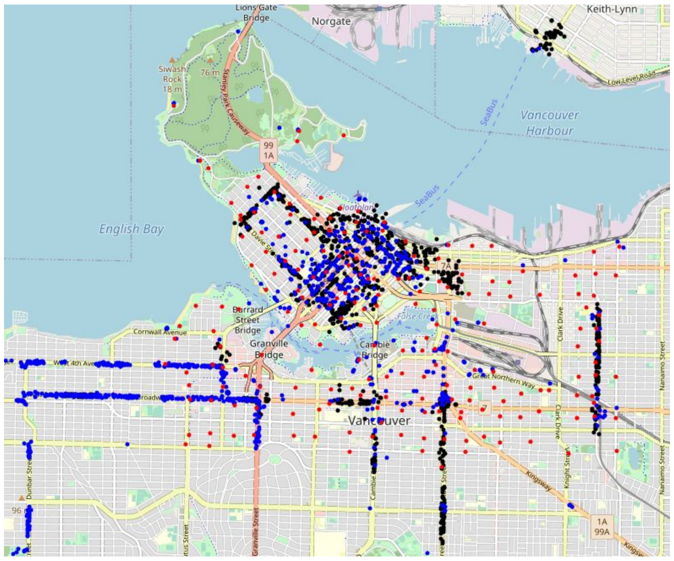


Figure 2.4

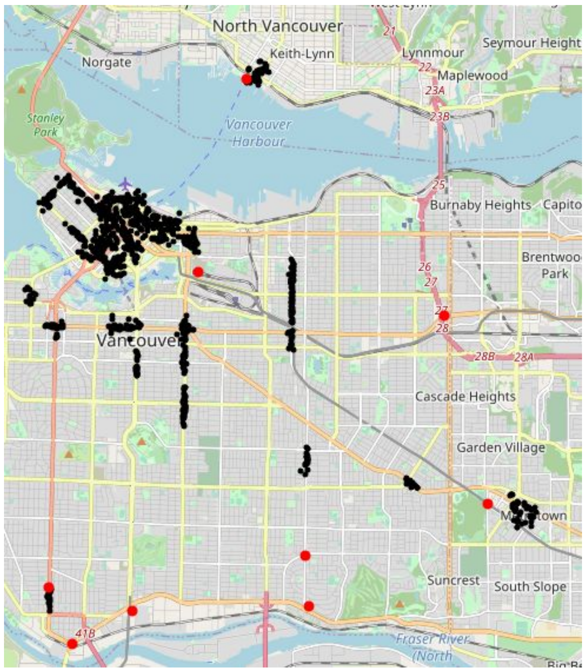
By combining information from two maps, we can see that the bicycle routes are mostly around Downtown and the bicycle rental as well, so we only took interesting places around Downtown area to generate the route for bicycle tours.

2.3.2) Bus Tour

To generate a tour that can be accessed through a bus, we filtered out locations from OSM data by amenity : “bus_station” and plotted the map with the interesting places (Figure 2.5). We found that there are three bus stations near the interesting area. So, these will be counted into suggestions to generate the bus tour.

(Red — Bus Station
Black – Interesting Place)

Figure 2.5 ⇒



2.3.3) Drive Tour

For self-drive, we filtered out locations from OSM data by amenity : "fuel" & "car_sharing" & "parking" and plotted the map with the interesting places (Figure 2.6). We found that there are also some areas where parking and interesting places are mixed up. So, these will be also counted into suggestions for generating the driving tour.

(Red — Parking Blue – Fuel

Black – Interesting Place)

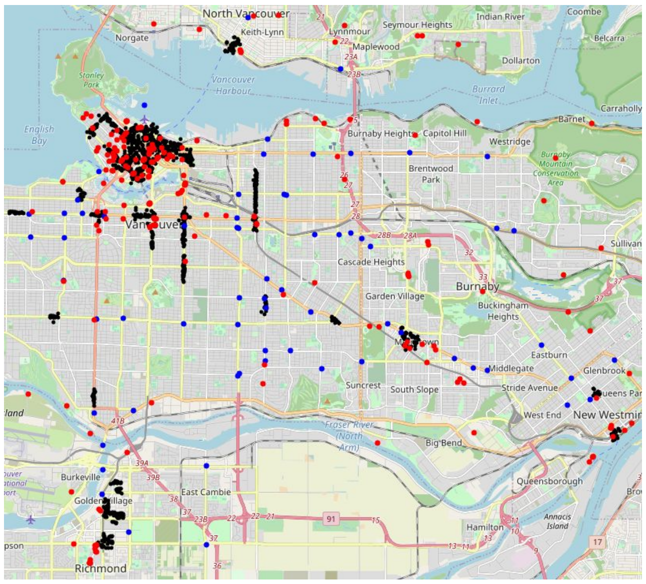


Figure 2.6 ⇒

2.4) Conclusion

We collected data from provided OSM data and Park data. Upon filtering and analyzing the data, we produced three tour routes around Vancouver, which usually set Downtown as the destination since there are relatively more interesting places in Downtown.

The three produced tours are as shown below:

For Bicycle Tour :

Downtown → Burrard Street Bridge → West Broadway → Broadway City Hall → Main Street → Science World → Downtown

*Rental bicycle in Downtown if needed

For Bus Tour :

Granville street → Metrotown (Patterson) → Downtown (Vancouver Pacific Central)

* ( ) as the bus station location

For Drive Tour :

1. Commercial Broadway → Main street → Broadway City Hall → West Broadway → Downtown

2. New Westminster → Metrotown → Richmond

3. Chain Restaurants vs. Non-chain Restaurants

3.1) Collecting & Cleaning Data:

On top of the OSM data provided on the course project page, we collected wikidata that shows all franchise restaurants, cafes, and pizzeria with the following SPARQL commands:

```
# Restaurants (chain-restaurants-wikidata.csv)
SELECT ?restaurant ?restaurantLabel
WHERE
{
  ?restaurant wdt:P31 wd:Q18534542.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
```

```
# Fast Food Restaurants (fastfood-chain-wikidata.csv)
SELECT ?item ?itemLabel
WHERE
{
  ?item wdt:P31 wd:Q18509232.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
```

```
# Cafe (cafe-chain-wikidata.csv)
SELECT ?item ?itemLabel
WHERE
{
  ?item wdt:P31 wd:Q76212517.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
```

```
# Pizzeria (pizzeria-chain.csv)
SELECT ?item ?itemLabel
WHERE
{
  ?item wdt:P31 wd:Q18654742.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
```

To clean our data, we first filtered out our OSM dataset so that it only contains data with ‘restaurant’, ‘fast_food’, and ‘cafe’ amenities. All restaurant names were capitalized so that restaurants of the same name would be marked the same even if lowercase and uppercase letters were different. We also removed restaurant data without any names since we would have no way of telling whether it is a chain restaurant or not. Timestamp of our OSM data was also removed since there was no need for it for our data analysis process.

3.2) Data Analysis & Results

We first needed to distinguish chain restaurants from non-chain restaurants. To do so, we compared the restaurant names of our OSM data restaurants and the wikidata containing the franchises. If they matched, then the restaurant in our OSM data was labelled as a chain restaurant. We also made an assumption that if there were more than two restaurants with the same name in our OSM, it was also a chain restaurant [1].

We decided that geographical visualization would be the best option to see whether some parts of the city contain more chain restaurants than non-chain restaurants, and whether chain restaurants are more spread out than non-chain restaurants. Using Python's folium library [2], we made heatmaps for each independently-owned and franchise restaurants:

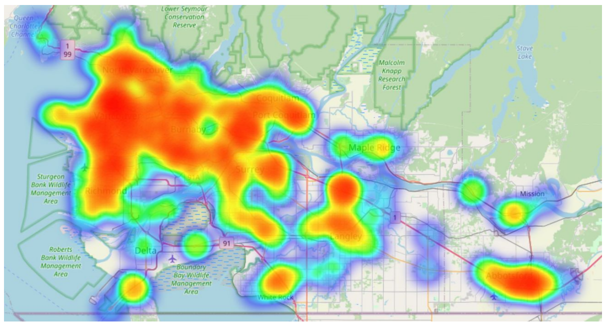


Figure 3.1 Heatmap for Chain Restaurants

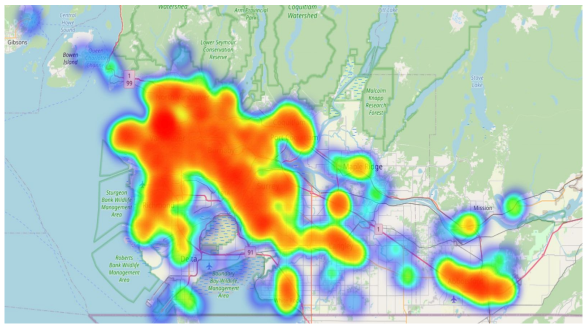


Figure 3.2. Heatmap for Non-chain Restaurants

From the two maps, we can see that non-chain restaurants tend to be more concentrated in Metro Vancouver and less spread out than chain restaurants. We also drew a cluster map to observe the exact number of chains and non-chains in parts of the city so that we can more safely conclude that there are more non-chain restaurants than chain restaurants in some parts of the city:

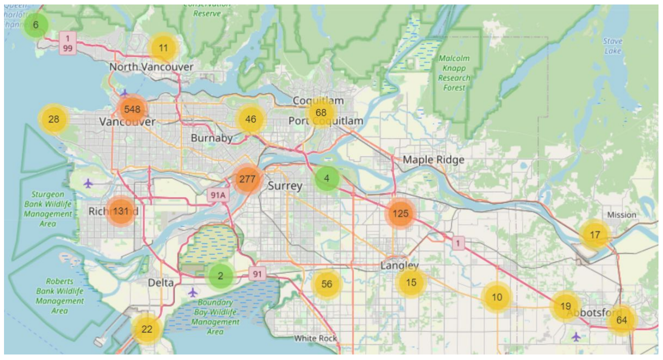


Figure 3.3 Cluster Map for Chain Restaurants

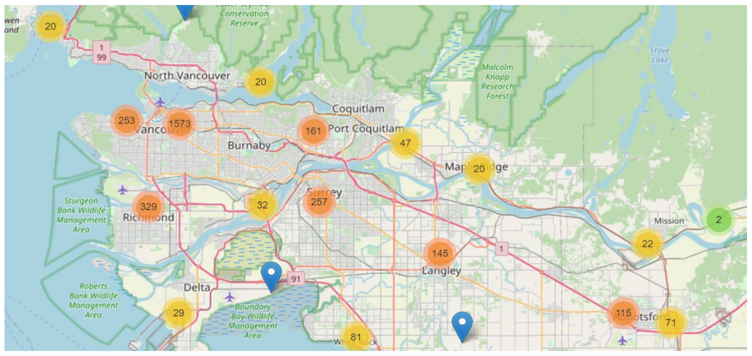


Figure 3.4. Cluster Map for Non-chain Restaurants

Based on the numbers shown for each cluster, we can also see that there are more non-chain restaurants in the Metro Vancouver area. We also calculated variances of chain and non-chain restaurants to see the spread of the data:

	Longitude Variance	Latitude Variance
Chain	~0.05745	~0.00637
Non-chain	~0.05145	~0.00585

Although the variances are very similar, we can see that chain restaurants tend to be more spread out than non-chain restaurants.

3.3) Summary

Upon analyzing our OSM data through comparing with wikidata, drawing out heat and cluster maps, and calculating variances, we can confidently conclude that chain restaurants tend to be more spread out, but less concentrated in the Metro Vancouver area.

Project Experience

Otto Hu:

- Performed preprocessing tasks like data cleaning and feature extraction using pandas.
- Created plots and maps to illustrate the data using folium and pandas.
- Used different classifier algorithms in sklearn and pandas to predict the category of map items
- Tried different settings of algorithm and features to use to improve the accuracy score

Justin Siu:

- Collect and Filter the park data From online open data resource
- Group Data using pandas and learn how to use two cluster methods, KMeans & DBSCAN, to cluster data in to group
- Draw different tour maps with folium function and add the cluster data in each map

Sam Kim:

- Successfully distinguished chain and non-chain restaurants through writing Python functions and collecting restaurant wikidata using SPARQL commands.
- Helped visualize the distribution and density of chain and non-chain restaurants through drawing heat and cluster maps using Python’s folium library.

References

[1] Unknown. Difference between Chain Restaurant and Independent Restaurant, hehehe3304.blogspot.com/2014/10/difference-between-chain-restaurant-and.html.

[2] Ivan, Anthony. “Data 101s: Spatial Visualizations and Analysis in Python with Folium.” Medium, Towards Data Science, 20 Nov. 2018, towardsdatascience.com/data-101s-spatial-visualizations-and-analysis-in-python-with-folium-39730da2adf.

[3] OpenStreetMap, www.openstreetmap.org/.
“Parks.” - City of Vancouver Open Data Portal, 9 Sept. 2019, opendata.vancouver.ca/explore/dataset/parks/information/.

[4] “Sklearn.cluster.KMeans¶.” Scikit, scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

[5] “Sklearn.cluster.DBSCAN¶.” Scikit, scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

[6] *OpenStreetMap*, https://www.openstreetmap.org/#map=12/49.2725/-123.1094&layers=C

[7] VanderPlas, Jake. “Vectorized String Operations.” Vectorized String Operations | Python Data Science Handbook, jakevdp.github.io/PythonDataScienceHandbook/03.10-working-with-strings.html.