CMPT353 Project

By Nicolas Ong (301312417) and Colin Kwok (301300876).

The problem you are addressing, particularly how you refined the provided idea.

The problems we are addressing are the ones suggested on the OSM project page. The first one is:

"If I was going to choose a hotel (or AirBnb), where should it be? What places have good amenities nearby?"

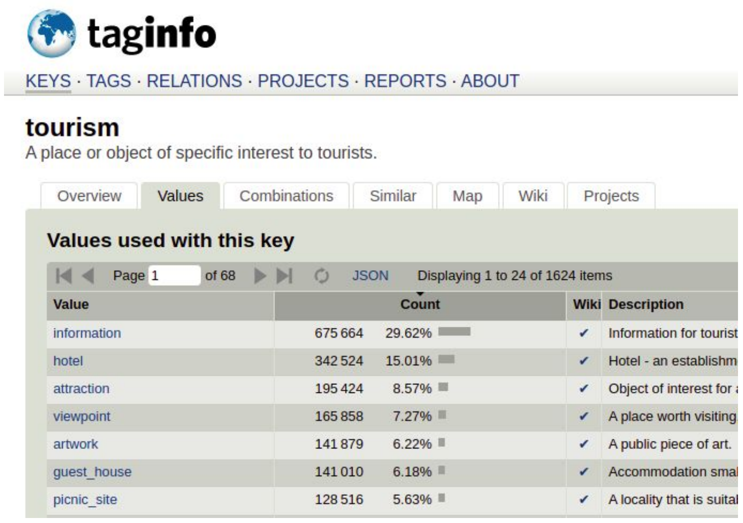
The others will be in part 2 of this report.

To refine this idea, we decided that we would give each location a score, how good the location would be for a hotel. The score would only be based on the nearby amenities, so places with good scores indicate good amenities nearby. On the other hand, it wouldn't tell you if a hotel is good based on interior features like service, room quality, or free breakfasts.

To do this, we would train a model with features like "nearest restaurant" and "number of theatres within walking distance", then tell it to predict the score of a hotel with those features.

The data that you used: how it was gathered, cleaned, etc.

Using the "0-exploring-tags.py" file, we realized we didn’t have enough hotels to train a model on. To figure out how to find hotels in the OpenStreetMap data, we used the "taginfo" tool. (https://taginfo.openstreetmap.org/search?q=hotel#values.) This showed us that we should focus on the tourism and building tags.



tourism

A place or object of specific interest to tourists.

Values used with this key

Displaying 1 to 24 of 1624 items

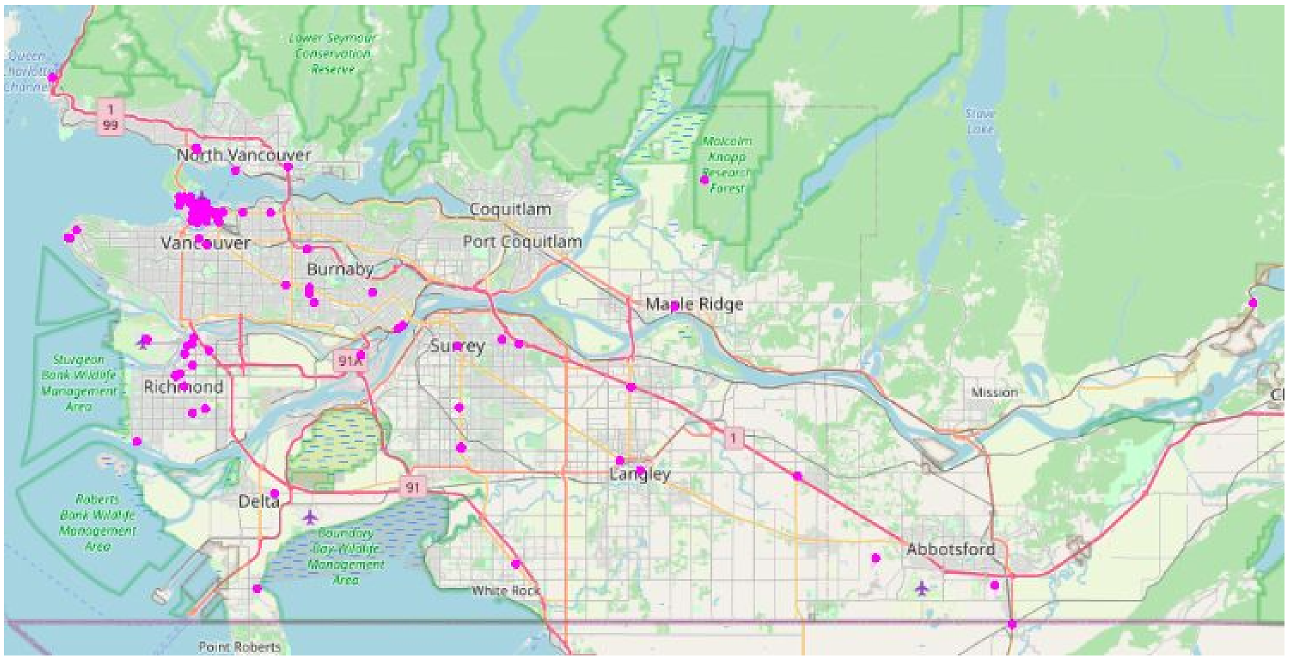
Value	Count		Wiki	Description
information	675 664	29.62%	✓	Information for tourist
hotel	342 524	15.01%	✓	Hotel - an establishm
attraction	195 424	8.57%	✓	Object of interest for
viewpoint	165 858	7.27%	✓	A place worth visiting
artwork	141 879	6.22%	✓	A public piece of art.
guest_house	141 010	6.18%	✓	Accommodation smal
picnic_site	128 516	5.63%	✓	A locality that is suital

We then modified the given files into "1-osm-tourism.py" and "2-just-vancouver.py" to get this information. After running them on the compute cluster for 2 hours, we got the 120 hotels in Vancouver out of the possible 551,817 worldwide. These can be found in the "buildings-vancouver.json.gz" and "tourism-vancouver.json.gz" datasets.

Using the coordinates of the hotels, we used the code in the "3-hotel-features.py" file to generate features that describe the state of nearby amenities. For example, two of the features may be "bench-closest" and "bench-near", which mean the distance of the closest bench and the amount of benches within walking distance. We found these features for a handful of amenities which had a significant amount (>40) of occurrences in the given OSM amenities dataset.

The last step to complete this dataset would be getting the scores of each hotel. We decided that the most efficient way to do this would be manually. This is for three reasons. First, we only had 120 hotels to retrieve the scores of, so it’d be faster to do it manually than write a script to automate it. Second, there were many hotels with no names or vague names that we'd either have to fix manually or write an exception for if we tried to automate. Third, we also wanted to keep as many of the 120 hotels as possible, since it's not a lot of data for data analysis. This process produced the "hotel_scores_manual.csv" file.

After cleaning this dataset with the "4-hotel-score-clean.py" file, we made the "hotel_scores.csv" file. By the end, we had 109 hotels to work with.



Techniques you used to analyse the data.

To analyze the data, we tried many different regression machine learning algorithms. We did this to find one that predicted well. To do this, we tried training the data on various models we learned about. For example, we tried KNeighborsRegressor, RandomForestRegressor, LinearRegression, and Ridge. We did this in the “5-hotel-regression.py” file.

We also tried some preprocessing methods we learned about, such as scaling before fitting with MinMaxScaler or StandardScaler, to not much improvement.

Model	Initial Model Score
KNeighborsRegressor	r^2=0.2

RandomForestRegressor	r^2=0.49
LinearRegression	r^2=-60
Ridge	r^2=0

After seeing the results of each model, we decided to continue with the RandomForestRegressor, and look for optimal parameters.

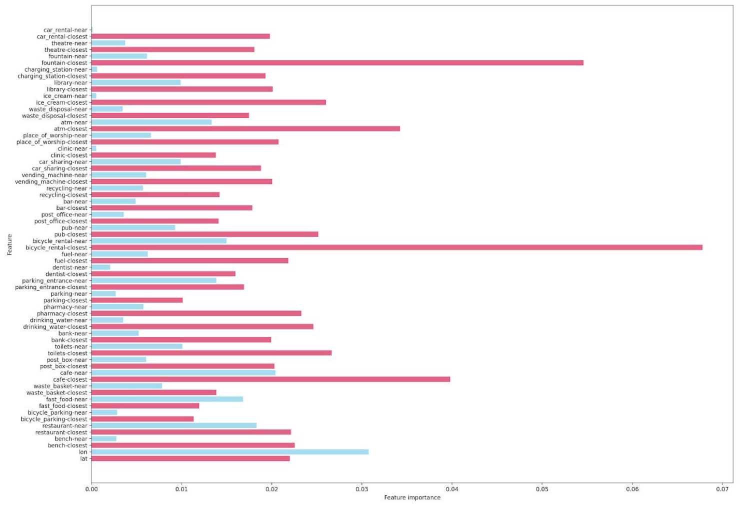
At the end of this process, we got a consistent score of r^2=0.35.

Your results/findings/conclusions. Some appropriate visualization of your data/results.

Using the RandomForestRegressor model we built, we got a consistent score of r^2 = 0.35, meaning that 35% of the variation in hotel scores could be explained by the amenities nearby.

The first thing we did with this model is look at its feature_importances_. This variable shows how important the model thinks each of the features given is when predicting a hotel's score. By doing this, we can see what features are important to look for when choosing a hotel. Perhaps there are features that not many people explicitly look for, yet looking for them will allow you to know what hotels are desirable at a glance.

Here are the features we gave the model, and their importances. You can see the full image in the repo.



The first thing you may notice is that some features are much more important than others. The four most important features to consider when choosing a hotel, according to the model, are:

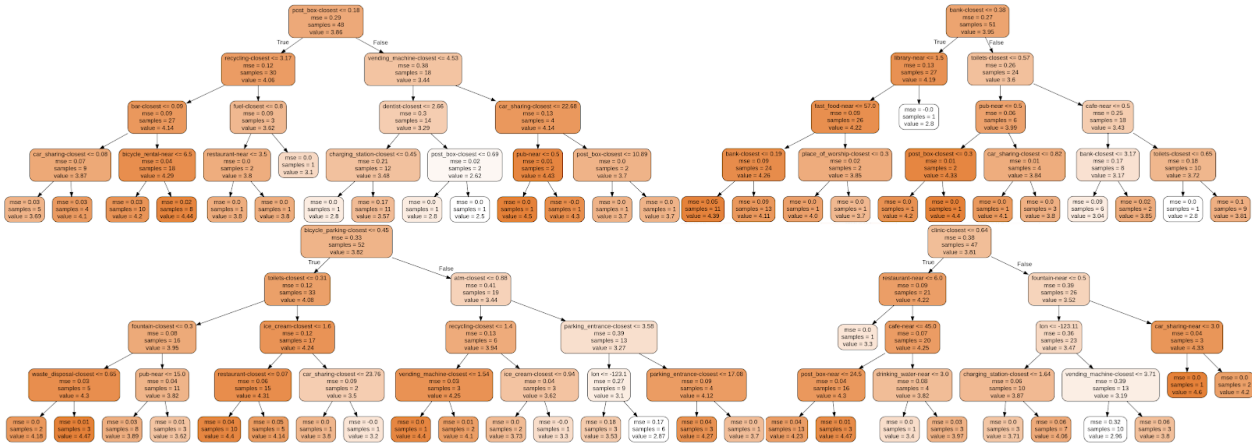
- How far the closest bicycle rental place is
- How far the closest fountain is
- How far the closest cafe is

- How far the closest ATM is

The ATM is self-explanatory - travelers often want access to ATMs when travelling. The others are a little more confusing. Why are fountains important? One conclusion is that fancy hotels have fountains in front of the building. Bicycle rental places may indicate a low-crime area, since they are prone to theft. Finally, cafes may indicate a wealthier area, where coffee is preferred over fast food.

Another thing to notice about the graph is the prevalence of red. The red bars correspond to "X-closest" features, which indicate how far away the closest "X" is. The blue bars correspond to the "X-near" features, which indicate how many "X"s are within walking distance. The graph shows that for nearly every amenity, the -closest feature is more important than the -near feature. We can conclude from this that people care more about if a wanted amenity is close to the hotel, rather than having a lot of them nearby. The only exception to this rule is the fast_food amenity. Perhaps having many fast food places nearby is better than just one very close.

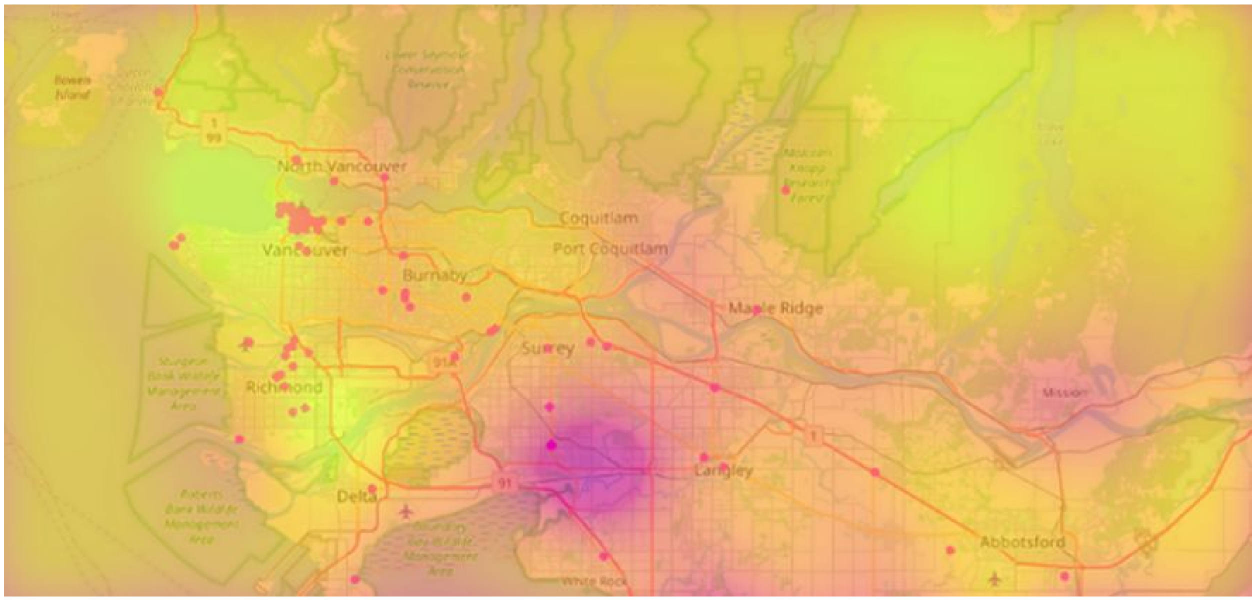
After looking at the feature importances, we decided to look at the decision paths of various trees in the model. The full images can be seen in the repo.


post_box-closest <= 0.18 mse = 0.29 samples = 48 value = 3.86
True
False
recycling-closest <= 3.17 mse = 0.12 samples = 30 value = 4.06
vending_machine-closest <= 4.53 mse = 0.38 samples = 18 value = 3.44
bar-closest <= 0.09 mse = 0.09 samples = 27 value = 4.14
fuel-closest <= 0.8 mse = 0.09 samples = 3 value = 3.62
dentist-closest <= 2.66 mse = 0.3 samples = 14 value = 3.29
car_sharing-closest <= 22.68 mse = 0.13 samples = 4 value = 4.14
car_sharing-closest <= 0.08 mse = 0.07 samples = 9 value = 3.87
bicycle_rental-near <= 6.5 mse = 0.04 samples = 18 value = 4.29
restaurant-near <= 3.5 mse = 0.0 samples = 2 value = 3.8
mse = 0.0 samples = 1 value = 3.1
charging_station-closest <= 0.45 mse = 0.21 samples = 12 value = 3.48
post_box-closest <= 0.69 mse = 0.02 samples = 2 value = 2.62
pub-near <= 0.5 mse = 0.01 samples = 2 value = 4.43
post_box-closest <= 10.89 mse = 0.0 samples = 2 value = 3.7
mse = 0.03 samples = 5 value = 3.69
mse = 0.03 samples = 4 value = 4.1
mse = 0.03 samples = 10 value = 4.2
mse = 0.02 samples = 8 value = 4.44
mse = 0.0 samples = 1 value = 3.8
mse = 0.0 samples = 1 value = 3.8
mse = 0.0 samples = 1 value = 2.8
mse = 0.17 samples = 11 value = 3.57
mse = 0.0 samples = 1 value = 2.8
mse = 0.0 samples = 1 value = 2.5
mse = 0.0 samples = 1 value = 4.5
mse = -0.0 samples = 1 value = 4.3
mse = 0.0 samples = 1 value = 3.7
mse = 0.0 samples = 1 value = 3.7
bank-closest <= 0.38 mse = 0.27 samples = 51 value = 3.95
True
False
library-near <= 1.5 mse = 0.13 samples = 27 value = 4.19
toilets-closest <= 0.57 mse = 0.26 samples = 24 value = 3.6
fast_food-near <= 57.0 mse = 0.09 samples = 26 value = 4.22
mse = -0.0 samples = 1 value = 2.8
pub-near <= 0.5 mse = 0.06 samples = 6 value = 3.99
cafe-near <= 0.5 mse = 0.25 samples = 18 value = 3.43
bank-closest <= 0.19 mse = 0.09 samples = 24 value = 4.26
place_of_worship-closest <= 0.3 mse = 0.02 samples = 2 value = 3.85
post_box-closest <= 0.3 mse = 0.01 samples = 2 value = 4.33
car_sharing-closest <= 0.82 mse = 0.01 samples = 4 value = 3.84
bank-closest <= 3.17 mse = 0.17 samples = 8 value = 3.17
toilets-closest <= 0.65 mse = 0.18 samples = 10 value = 3.72
mse = 0.05 samples = 11 value = 4.39
mse = 0.09 samples = 13 value = 4.11
mse = 0.0 samples = 1 value = 4.0
mse = 0.0 samples = 1 value = 3.7
mse = 0.0 samples = 1 value = 4.2
mse = 0.0 samples = 1 value = 4.4
mse = 0.0 samples = 1 value = 4.1
mse = 0.0 samples = 3 value = 3.8
mse = 0.09 samples = 6 value = 3.04
mse = 0.02 samples = 2 value = 3.85
mse = 0.0 samples = 1 value = 2.8
mse = 0.1 samples = 9 value = 3.81
bicycle_parking-closest <= 0.45 mse = 0.33 samples = 52 value = 3.82
True
False
toilets-closest <= 0.31 mse = 0.12 samples = 33 value = 4.08
atm-closest <= 0.88 mse = 0.41 samples = 19 value = 3.44
fountain-closest <= 0.3 mse = 0.08 samples = 16 value = 3.95
ice_cream-closest <= 1.6 mse = 0.12 samples = 17 value = 4.24
recycling-closest <= 1.4 mse = 0.13 samples = 6 value = 3.94
parking_entrance-closest <= 3.58 mse = 0.39 samples = 13 value = 3.27
waste_disposal-closest <= 0.65 mse = 0.03 samples = 5 value = 4.3
pub-near <= 15.0 mse = 0.04 samples = 11 value = 3.82
restaurant-closest <= 0.07 mse = 0.06 samples = 15 value = 4.31
car_sharing-closest <= 23.76 mse = 0.09 samples = 2 value = 3.5
vending_machine-closest <= 1.54 mse = 0.03 samples = 3 value = 4.25
ice_cream-closest <= 0.94 mse = 0.04 samples = 3 value = 3.62
lon <= -123.1 mse = 0.27 samples = 9 value = 3.1
parking_entrance-closest <= 17.08 mse = 0.09 samples = 4 value = 4.12
mse = 0.0 samples = 2 value = 4.18
mse = 0.01 samples = 3 value = 4.47
mse = 0.03 samples = 8 value = 3.89
mse = 0.01 samples = 3 value = 3.62
mse = 0.04 samples = 10 value = 4.4
mse = 0.05 samples = 5 value = 4.14
mse = 0.0 samples = 1 value = 3.8
mse = -0.0 samples = 1 value = 3.2
mse = 0.0 samples = 1 value = 4.4
mse = 0.01 samples = 2 value = 4.1
mse = 0.0 samples = 2 value = 3.73
mse = -0.0 samples = 1 value = 3.3
mse = 0.18 samples = 3 value = 3.53
mse = 0.17 samples = 6 value = 2.87
mse = 0.04 samples = 3 value = 4.27
mse = 0.0 samples = 1 value = 3.7
clinic-closest <= 0.64 mse = 0.38 samples = 47 value = 3.81
True
False
restaurant-near <= 6.0 mse = 0.09 samples = 21 value = 4.22
fountain-near <= 0.5 mse = 0.39 samples = 26 value = 3.52
mse = 0.0 samples = 1 value = 3.3
cafe-near <= 45.0 mse = 0.07 samples = 20 value = 4.25
lon <= -123.11 mse = 0.36 samples = 23 value = 3.47
car_sharing-near <= 3.0 mse = 0.04 samples = 3 value = 4.33
post_box-near <= 24.5 mse = 0.04 samples = 16 value = 4.3
drinking_water-near <= 3.0 mse = 0.08 samples = 4 value = 3.82
charging_station-closest <= 1.64 mse = 0.06 samples = 10 value = 3.87
vending_machine-closest <= 3.71 mse = 0.39 samples = 13 value = 3.19
mse = 0.0 samples = 1 value = 4.6
mse = 0.0 samples = 2 value = 4.2
mse = 0.04 samples = 13 value = 4.23
mse = 0.01 samples = 3 value = 4.47
mse = 0.0 samples = 1 value = 3.4
mse = 0.03 samples = 3 value = 3.97
mse = 0.0 samples = 3 value = 3.71
mse = 0.06 samples = 7 value = 4.06
mse = 0.32 samples = 10 value = 2.96
mse = 0.06 samples = 3 value = 3.8

One thing interesting to see when looking at these decision trees is which features are considered good, and which are considered bad.

By following the graphs, we can see that some features are consistently considered bad, like vending machines. Hotels with a very close vending machine tended to have lower scores. This can be seen in trees 0, 20, and 80.

Finally, we decided to look at where the model predicted hotels with high scores would be, and where they'd have low scores.



We can see two bright spots on the left, which correspond to Richmond and to the left of Vancouver. To the right, there is another large bright spot, in the forest. Perhaps the model only has one good hotel in the forest to go off of, which is why the forests/mountains to the North East of vancouver are so highly rated. We can also see an area to the bottom-middle of the map that is very dark. This is the area to the South of Surrey.

In conclusion, poor hotels can be found in South Surrey, while good hotels and amenities can be found in Vancouver, Richmond, and the North-Eastern forests.

Limitations: problems you encountered, things you would do if you had more time, things you should have done in retrospect, etc.

One of the biggest problems we encountered was the lack of data. There were essentially no hotels in the given amenities data, and of the hotels we found through OpenStreetMaps, only 109 were in Vancouver and usable. If we had more data, our model would be able to make better and more consistent predictions about hotel quality based on amenities nearby.

Another type of data we'd like to have more of is the amenity data. Of the amenity data given, only a few had more than 40 samples. More would be better. We'd also like more specific amenity data. For example, types of restaurants instead of just "restaurant". This is also something we would've implemented if we had more time.

If we managed to get a lot more data to improve the final model, another thing that we would've done would be to automate the hotel scoring process. The new data would have to be of higher quality, with names for each hotel.

A problem we encountered was the lack of predicting power by amenities. We always knew this would be the case, since most people rate hotels for non-amenity items, like the rooms or service. However, this is still a problem with our approach. We managed to get a model to explain ~35% of the variance in score, but sometimes the randomness of the model would create a worse model with a lower score (or one with a higher score, the highest being 50%). This is why more data is so important. We'd be able to know the true score of our model, and how much only amenities can explain.

Project Report, Part 2:

The focus of this part of the project addresses touring paths and restaurant analysis in the Lower Mainland of British Columbia based on the OpenStreetMap (OSM) data set.

In generating a touring path around the Lower Mainland, the goal is to answer: when planning a tour of the city (by walking/biking/driving), where should I go? This project approaches this question by suggesting a variety of interesting amenities or sites to visit given a start and end location. The focus of the touring path problem is further refined to suggest interesting amenities that are closest to the straight line displacement between the start and end point. In order to quantify “interest” for attractions and amenities, data of user ratings and recommendations from Yelp was incorporated into the touring path generation. Therefore, an additional refinement for touring path problem uses Yelp ratings as a heuristic for “interest”.

The restaurant analysis portion of the project focuses on answering: In different parts of the city, is there a difference in restaurant ratings, menu prices and relative densities of chain restaurants to non-chain restaurants? To refine this question, restaurant ratings, prices and chain densities are compared among restaurant clusters that were generated prior to the analysis. The data required for statistical analysis is also refined to user ratings and price categories from Yelp.

The first step of the project was to filter and clean the OSM data set. The original OSM data set, ‘amenities-vancouver.json.gz’, is filtered to only include amenity categories that are considered applicable for tourist or leisure activities. That is, only amenities related to restaurants, shopping, nightlife, or entertainment are left in the DataFrame after the filtering. For the data cleaning, OSM data entries without an amenity category, name, latitude and longitude were dropped from the DataFrame. This data cleaning step is necessary to ensure that each data entry has the sufficient parameters for making an API call to the Yelp business search endpoint.

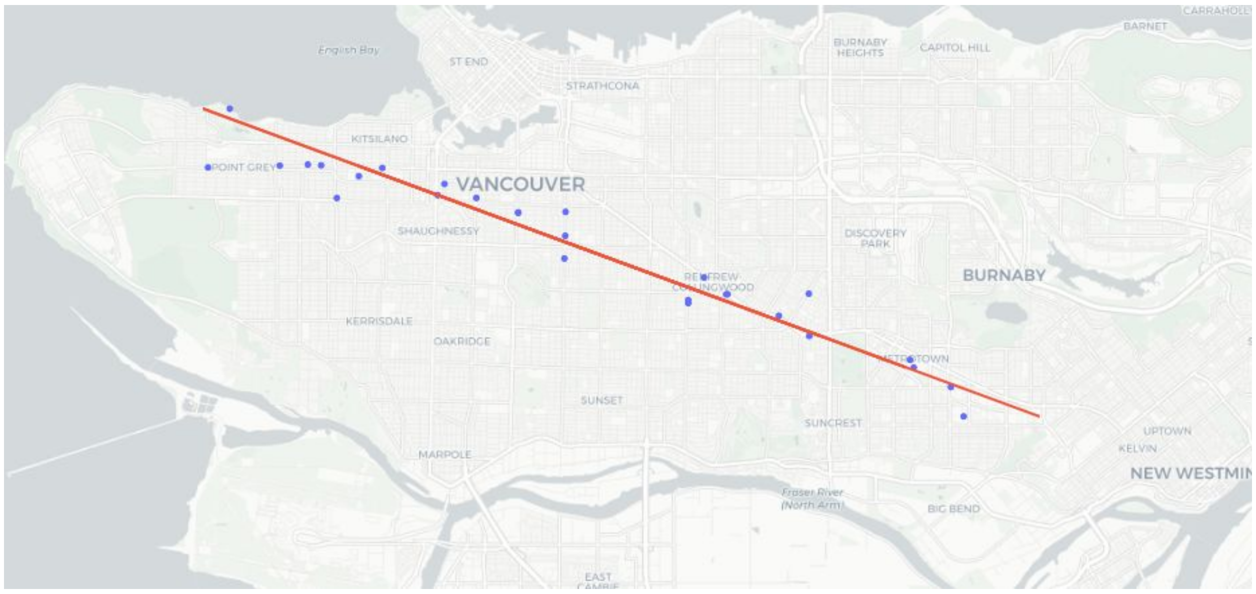
The data collection following data cleaning consists of iterating through the cleaned OSM data set and making an API call to the Yelp business endpoint for each entry. In order to use the Fusion API for the business endpoints, a registered Yelp account and an API key is necessary. The ratings for each OSM data entry is collected first. The name, latitude, longitude and amenity category for each OSM data entry are used as the search parameters in the API calls. In addition, a limit of ten businesses sorted by ‘best match’ are added as search parameters. In the case that multiple matches exist, the best match is chosen to be the rating for the OSM data entry. After the data has been collected, the resulting OSM data with ratings is stored in a compressed CSV file. Next, after the ratings have been collected the price for each OSM data entry is collected from the same Yelp endpoint. The same search parameters for the ratings collection are used for the price data collection. After the price data is collected for each OSM data entry, the result is stored in a separate compressed CSV file. In hindsight, the ratings and price data collection should have been done in one step rather than making a call to the endpoint twice. However, because the price analysis was an idea that came after the ratings, a two-step data collection is used in the project.

After data collection, in order to compare different parts of the Lower Mainland for the restaurant analysis, clusters of restaurants based on location needed to be identified. Before the clustering of restaurants is computed, additional data cleaning is necessary after data collection to drop data entries without a price. Likewise, the format of the price from the Yelp endpoint is categorical. The number of dollar signs ('$$') is converted to a numerical format.

K-Means clustering with latitude and longitude as the input data is used to determine the restaurant clusters in the Lower Mainland. K-Means Clustering is used because the metric used in the algorithm is based upon the distances between data points, which can be used to approximate relative distances between latitude and longitude data points. K-Means provides another advantage in the "number of clusters" parameter. As a result, by adjusting the number of clusters, there is the advantage of adjusting the number of restaurant clusters to match the various geographical parts of the Vancouver area.

The first part of the data analysis answers: when planning a tour of the city where should I go? OSM data was first filtered for high ratings (4.5/5.0+) to ensure only highly rated places are recommended along a touring path. Next, given the coordinates of the start and end location, ten equidistant points between the start and end coordinates are computed. For the start and end coordinates, along with each of the 10 equidistant points (12 total points), the Ball Tree K-Nearest Neighbors algorithm returns a maximum of 4 closest OSM entries with a 4.5+ rating. The Ball Tree algorithm is used because it supports the haversine distance metric, which correctly computes the distances between coordinates. The highly rated OSM data entries returned from the Ball Tree algorithm along with the start and end coordinates now form the touring path. As the last step, linear regression is used to form a best-fit line through the OSM data entries to map a straight line "tour". Figure 1 shows a map generated using Plotly to visualize a generated touring path. Each Blue point represents a highly rated attraction, and the red line is the straight line "tour" generated by linear regression.

Figure 1:



The next part of the data analysis answers: is there a difference in relative densities of chain restaurants to non-chain restaurants among the restaurant clusters? Chain restaurants are identified by a WikiData entry heuristic: an entry with a ‘brand’ Wikidata tag is regarded as a chain restaurant.

Figure 2: Map generated using Plotly visualizes the chain(red) and non-chain(blue) restaurants in the Lower Mainland

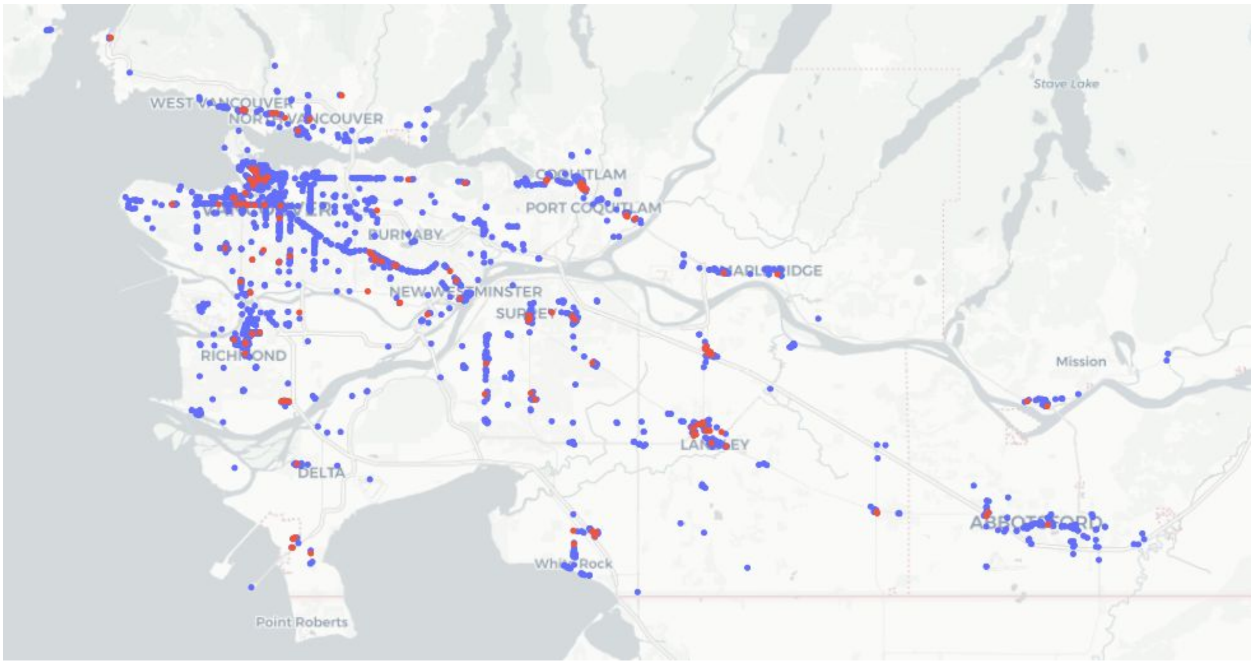
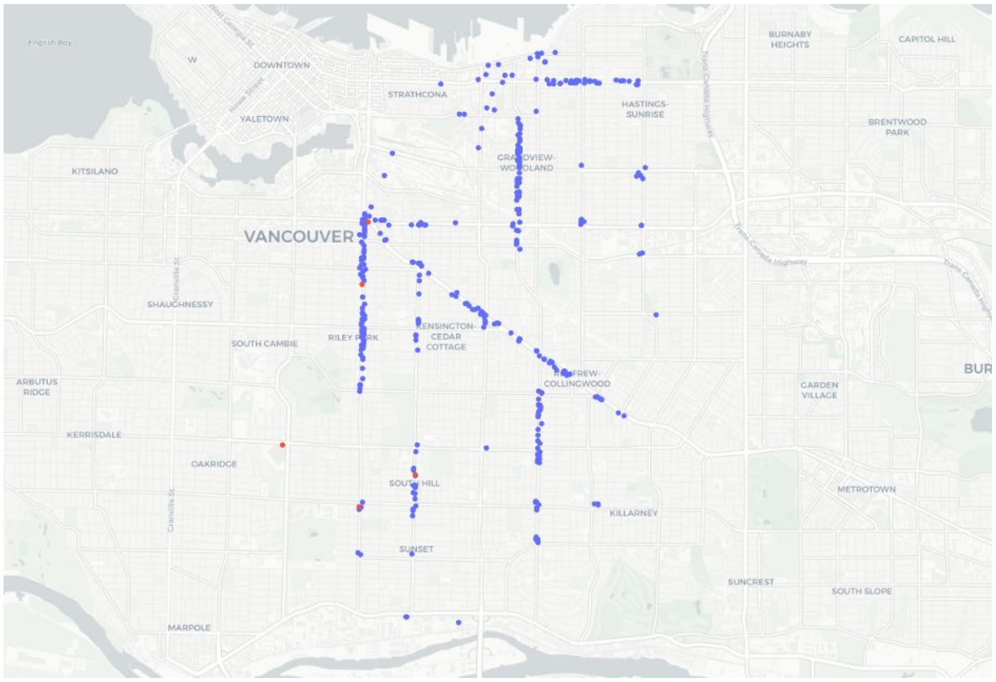


Figure 3: Map of Cluster 1 using Plotly visualizes the chain(red) and non-chain(blue) restaurants



The density of chain restaurants relative to non-chain restaurants is computed as a fraction of chain restaurants in a given cluster divided by all restaurants in the given cluster.

In conclusion, after computing the density of chain restaurants in each cluster, there is a difference in the density of chain restaurants relative to non-chain restaurants.
The full computed list of chain restaurant densities is:

Cluster 10, Chain restaurant density: 0.033112582781456956
Cluster 5, Chain restaurant density: 0.03541076487252125
Cluster 8, Chain restaurant density: 0.018518518518518517
Cluster 7, Chain restaurant density: 0.18867924528301888
Cluster 9, Chain restaurant density: 0.028901734104046242
Cluster 0, Chain restaurant density: 0.013192612137203167
Cluster 6, Chain restaurant density: 0.04779411764705882
Cluster 11, Chain restaurant density: 0.14285714285714285
Cluster 15, Chain restaurant density: 0.12149532710280374
Cluster 3, Chain restaurant density: 0.046875
Cluster 2, Chain restaurant density: 0.06285714285714286
Cluster 1, Chain restaurant density: 0.03076923076923077
Cluster 14, Chain restaurant density: 0.16666666666666666
Cluster 12, Chain restaurant density: 0.06
Cluster 13, Chain restaurant density: 0.025
Cluster 16, Chain restaurant density: 0.06666666666666667
Cluster 4, Chain restaurant density: 0.06097560975609756
Note that these values may vary due to the results of the restaurant clustering.

The last part of the data analysis answers: is there a difference in restaurant ratings and menu prices among the restaurant clusters? Since the ratings and prices data fails the normality test, a non-parametric method, the Kruskal–Wallis test is used to determine whether samples originate from the same distribution. The null hypothesis for the ratings analysis then becomes: the restaurant rating medians of all restaurant clusters are equal. Likewise, the null hypothesis for the prices analysis then becomes: the menu price medians of all restaurant clusters are equal.
For the Kruskal-Wallis test on restaurant ratings, the p-value was 1.4480832151545917e-12.
Likewise for the Kruskal-Wallis test on menu prices, the p-value was 1.2134316734167391e-05.
Note that these p-values may vary due to the results of the restaurant clustering.
Therefore, for both ratings and prices, we can reject the null hypothesis: there exists at least one median rating of one restaurant cluster that is different from the median rating of at least one other cluster. Similarly, there exists at least one median price of one restaurant cluster that is different from the median price of at least one other cluster

Since there is a significance with Kruskal-Wallis, for the post-hoc analysis, Dunn’s Test is used to determine which specific clusters have ratings or prices that tend to sort higher than the other.
The bonferroni correction was also used to adjust for the probability of incorrectly rejecting the null hypothesis when making pairwise comparisons between restaurant clusters.
The full pairwise p-values of the Dunn’s Test for the restaurant ratings and menu prices are outputted as ‘ratings_post_hoc.csv’ and ‘prices_post_hoc.csv’ respectively.

Figure 4: Restaurant Clusters from OSM ratings analysis

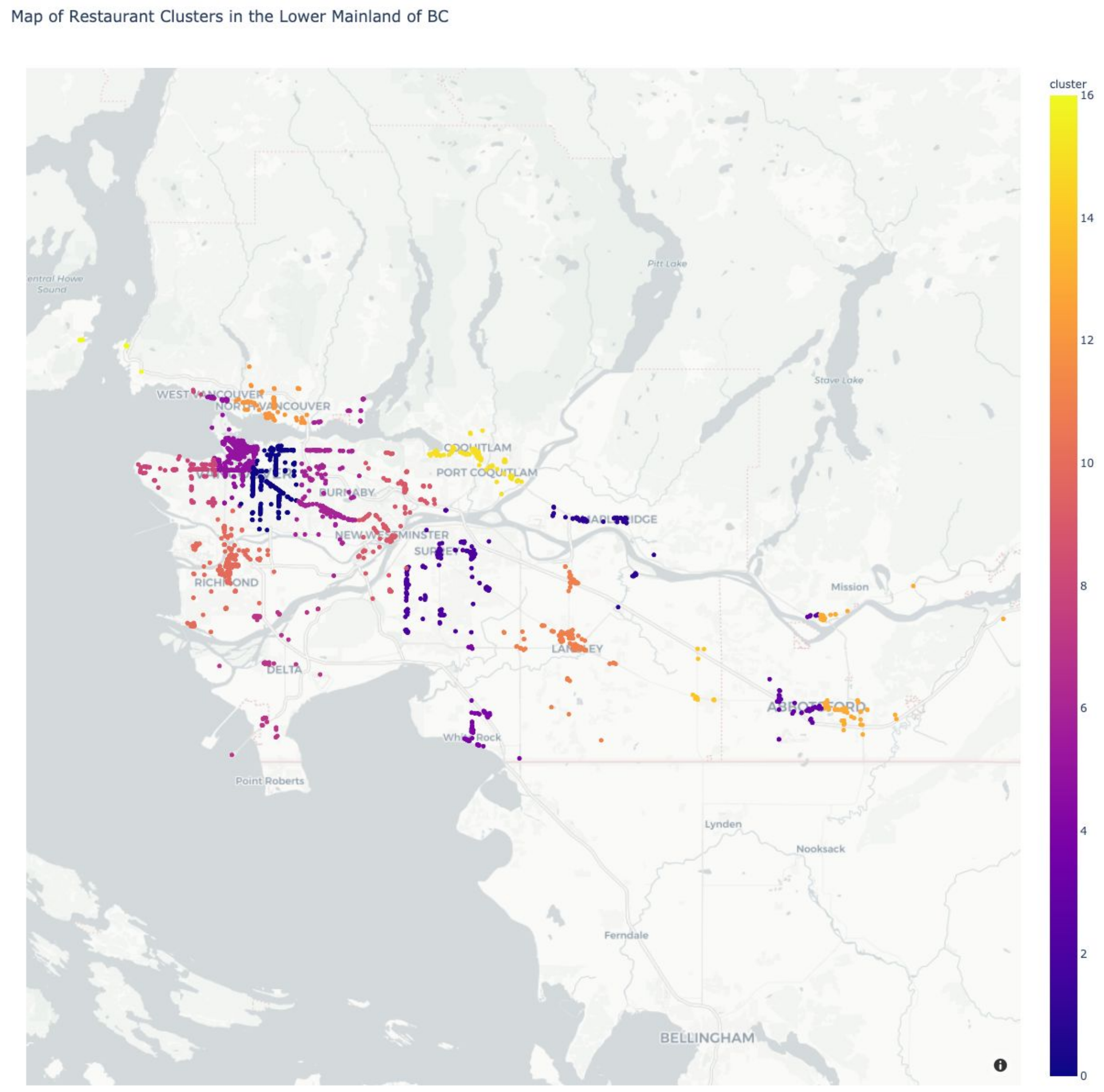


Figure 5: Map of Yelp Restaurant Ratings

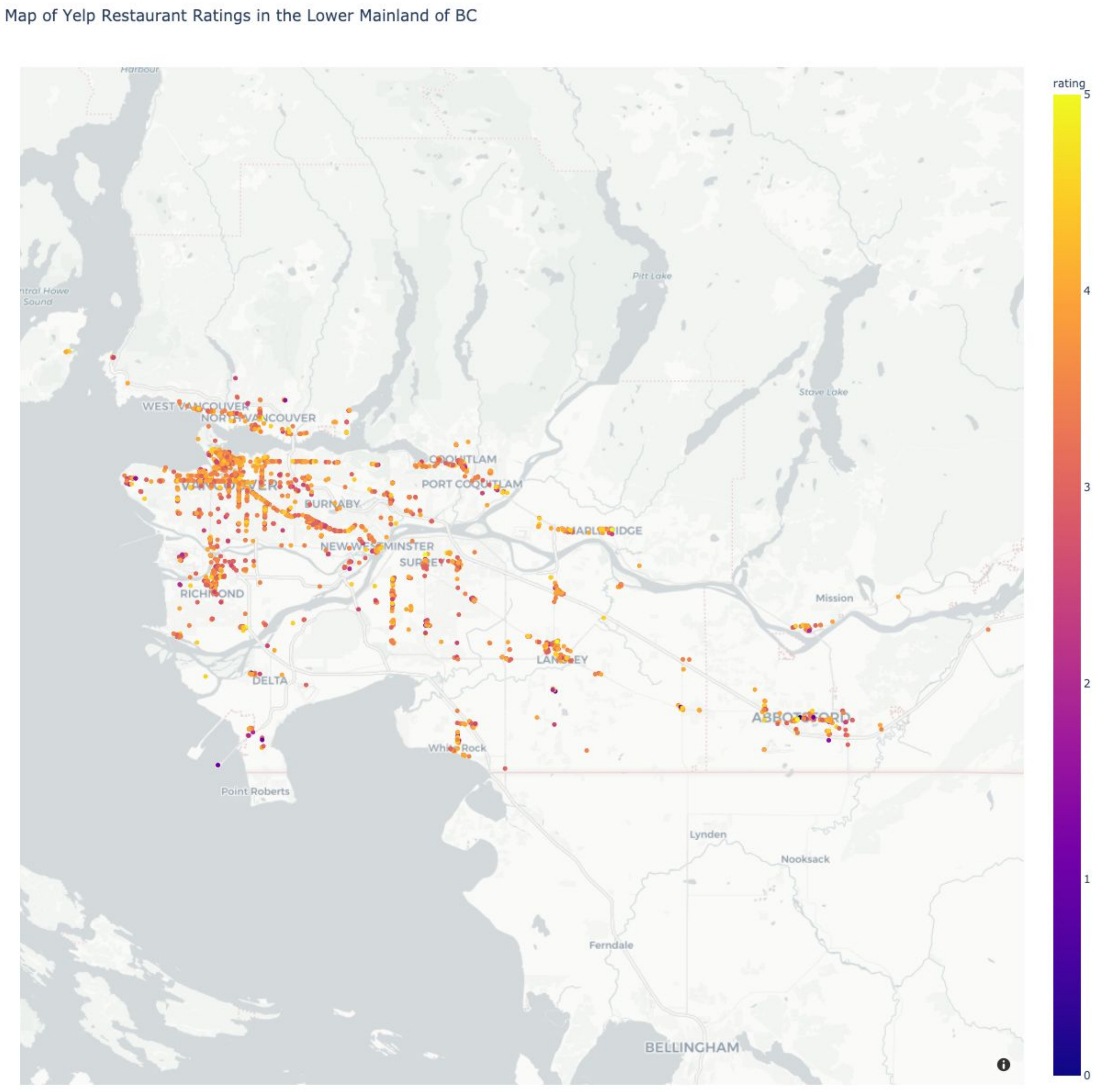


Figure 6: Restaurant Clusters from OSM prices analysis

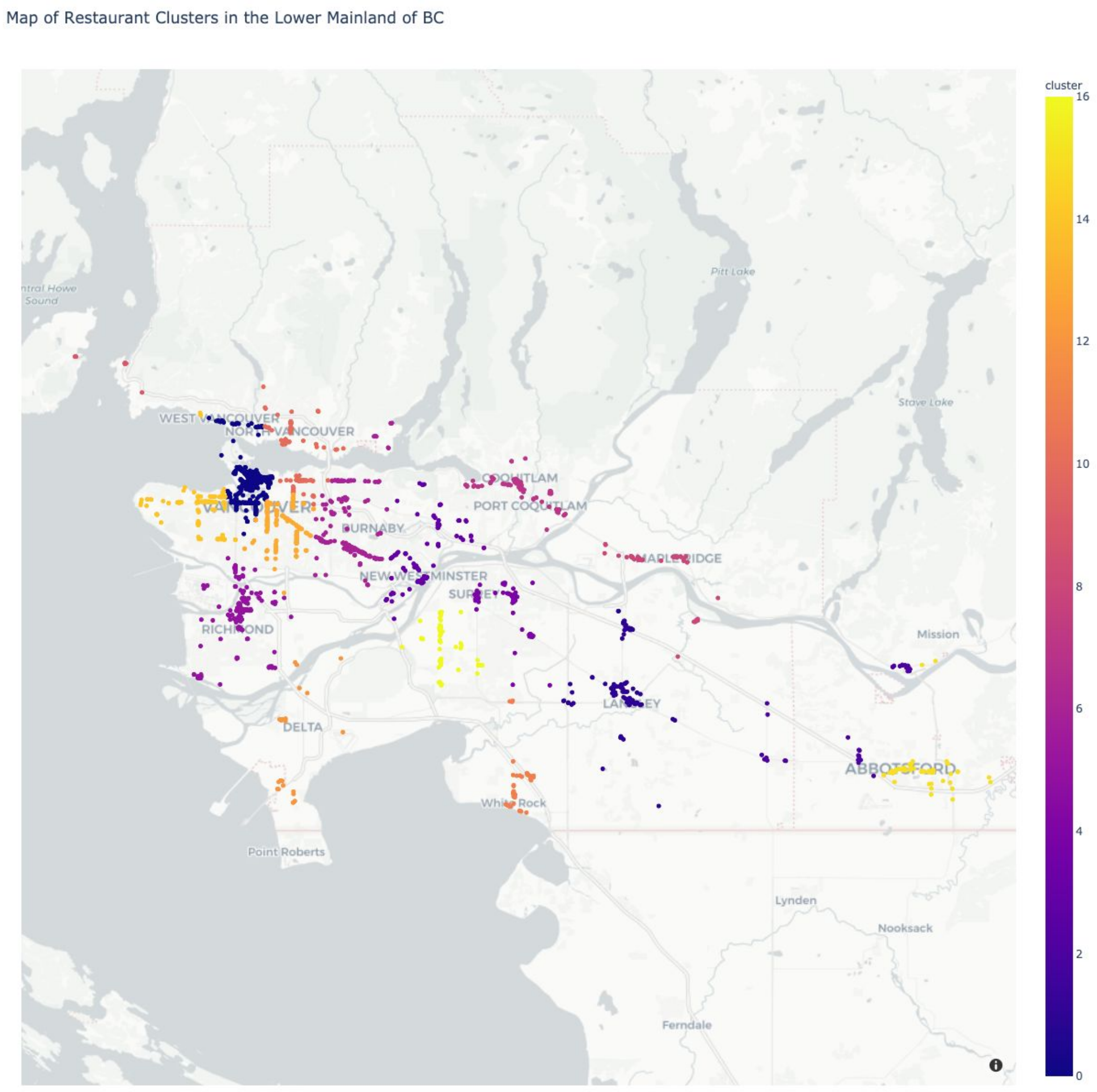
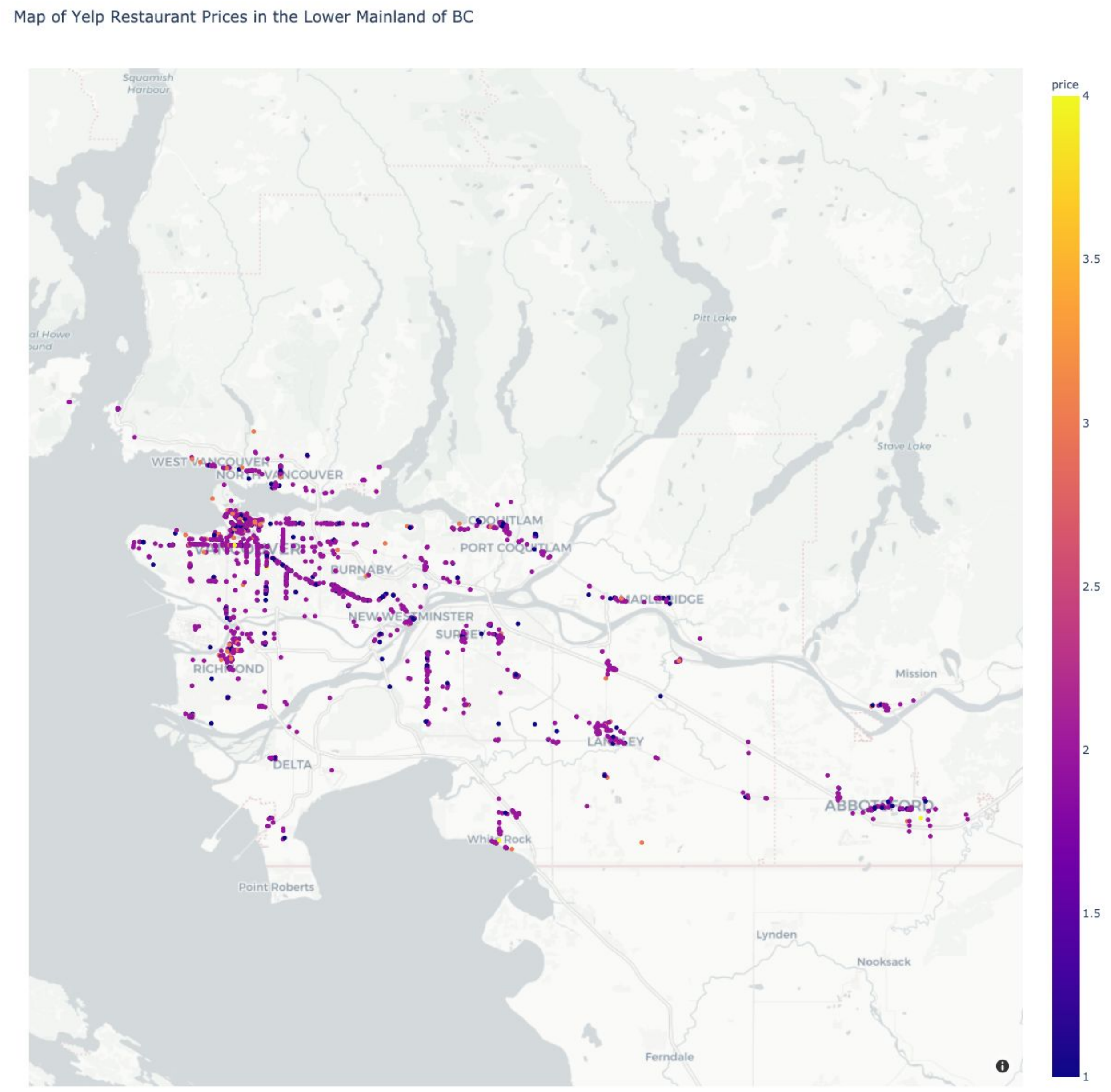


Figure 7: Map of Yelp Restaurant Prices



In terms of limitations, if there was more time, incorporating geotagged data generated from Instagram or Twitter posts would be an interesting heuristic to recommend “interesting” locations instead of using Yelp ratings alone. In retrospect, the data collection portion could have been done in one step rather than two steps. However, the prices analysis idea came after the initial data collection so the second round of data collection was needed.

Project Experience Summary

Colin Kwok

- Adapted Extract-Load-Transform procedure with Pandas to merge OpenStreetMap restaurant data with user recommendations from Yelp for statistical tests and machine learning techniques
- Engineered path recommender system with unsupervised nearest neighbors learning algorithm to suggest highly rated restaurants and attractions to users
- Utilized Unsupervised KMeans Clustering with Scikit Learn to determine restaurants clusters based on geographical locations in Vancouver
- Calculated Statistical Significance with non-parametric methods and post-hoc analysis to determine the difference in restaurant ratings, menu prices and chain restaurant density among restaurant clusters
- Visualized restaurant ratings, menu prices and chain restaurant density with Plotly to effectively communicate statistically significant results

Nicolas Ong

- Developed a data acquisition path based on a data exploration on OpenStreetMap data and research into OpenStreetMap's websites.
- Extracted relevant information from a big data dataset using Spark and Hadoop on a compute cluster.
- Engineered features to be fed into a machine learning model using Pandas and Numpy.
- Trained a model pipeline that can explain 35% of the variance in a hotel's score knowing just its location.
- Visualized the model-predicted quality of locations with Plotly to communicate results.
- Gained valuable insight about a model's inner process through inspection of its component parts.