CMPT353 Project

Project Topic: OSM, Photos, and Tours

Finding the Best Airbnbs for Tourists

Group: Pororo

Hyeonyoung Park 301414491

Jusung Park 301414852

Jooyoung Lee 301414539

1. Introduction

In this project, we focused on providing Airbnb lists to people who are looking for places that match their traveling style. Since everyone has different preferences, travellers from other countries or provinces need help finding suitable places to satisfy their specific tastes. Hence, we designed our project to respond to the following questions: If I was going to choose an Airbnb, where should it be? What places have good amenities nearby? We started by targeting tourists visiting Vancouver and using Vancouver's public transportation.

Our project has 3 main directories: *data-analysis*, *frontend*, and *backend*. In *the data-analysis* directory, we consume our collected datasets, clean and analyze them, score Airbnb with our business logic, and then output the Airbnb dataset with scores. The FastAPI application in the *backend* consumes that dataset and exposes a REST API endpoint to filter Airbnb with users' preferences. The React application in *frontend* is then used to interact with the end users to collect their preferences, make a request to the FastAPI instance, and then display the top 20 high-scored Airbnb list.

2. Data Collection

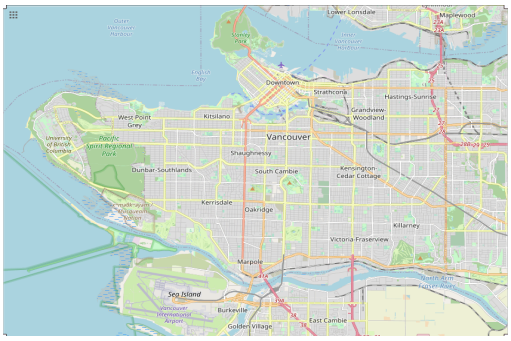
We collected 4 different types of data sets inside the data-analysis directory:

a. Airbnb dataset - *airbnb/raw_data/airbnb_listings.csv.gz*
Source: http://insideairbnb.com/get-the-data/

b. Public Transportation (Bus stops & Skytrain station) - *transportation/ Stops.csv, Stations.csv*
Source:
https://www.translink.ca/schedules-and-maps/interactive-system-map

c. OSM(OpenStreetMap) - *osm/*
Multiple steps were required to collect the osm data we wanted.

1. Extract osm data with the boundary below from https://extract.bbbike.org/ and save it at *raw-data/*.



coordinates:-123.28,49.18_-122.99,49.32)

2. Disassemble the XML data with the given `disassemble-osm.py`.

3. Use `osm-get-certain-tag.py` to extract nodes with specific tags. After looking through, all types of tags here(https://wiki.openstreetmap.org/wiki/Category:Features), we obtained these osm node datasets in json.gz format at *output/*; *amenity, leisure, shop, sport, tourism*.

d. Local area boundary - *airbnb/raw-data/local-area-boundary.geojson*
Source: opendata.vancouver.ca/explore/dataset/local-area-boundary

3. Data Cleaning - *data-analysis/*

a. Airbnb - *airbnb/logic/airbnb_cleaning*
The original *airbnb_listings.csv* dataset includes unnecessary columns so we only selected those we would use for further filtering. We also adjusted prices from the string type to the int type(ex: $1,000 -> 1000). These cleanings and exporting were done in *airbnb_cleaning.py*

- 'review_scores_rating': Used the MinMaxScaler to scale down each score from 0 to 5 to 0 to 1.

b. Public Transportation (Bus stops & Skytrain stations)
After merging two datasets: *transportation/Stops.csv, Stations.csv* together into airbnb/cleaned_data/*cleaned_transportation.csv,* we filtered out unnecessary categories and left only coordinates and names.

c. OSM(OpenStreetMap) - *osm/*
From the collected OSM(OpenStreetMap) data, we decided to filter out some unnecessary data and only leave the data that are needed for tourists. So we kept the data that has tag types of 'amenity', 'leisure', 'shop', and 'tourism'. Then we looked at the data of each tag more closely:

- 'amenity': According to the types of data in amenity tags, created two categories 'food' and 'entertainment'. Then, sorted the related data and saved them into two different files, *cleaned_food.csv* and *cleaned_entertainment.csv*.

- 'shop': Kept only the types that tourists might need and cleaned out data that is not related to shopping. For example, we eliminated 'copyshop', 'funeral_directors', 'construction_equipment', etc., and saved the remaining data into *cleaned_shop.csv*.

- 'tourism': Removed the 'guest_house' type since we are using our collected Airbnb data for the places to stay and we are more focused on tourist

attraction from this data set. Then, saved everything except 'guest_house' into *cleaned_tourism.csv*.

In order to make sure that our data is as consistent and accurate as possible, we dropped the 'marketplace' and 'arts_centre' types from the entertainment dataset and relocated them into the tourism dataset. The same steps were used for the 'spa', so we put them into the leisure dataset instead of the entertainment dataset. These rearranging procedures of the data were done in *relocate-types.py*.

4. Data Analysis - Tools (Libraries used)

a. GeoPanda
- Convert coordinates to Point, Polygon Object
- Visualize geometry data like Points, intersections, and intersection boundaries with its integrated Folium library

b. Pandas/Numpy
- Clean, filter, and analyze collected data.

c. Shapely
- Create Polygon object with a Points List

d. PySpark & lxml
- Disassemble and extract OSM XML data into json.gz files

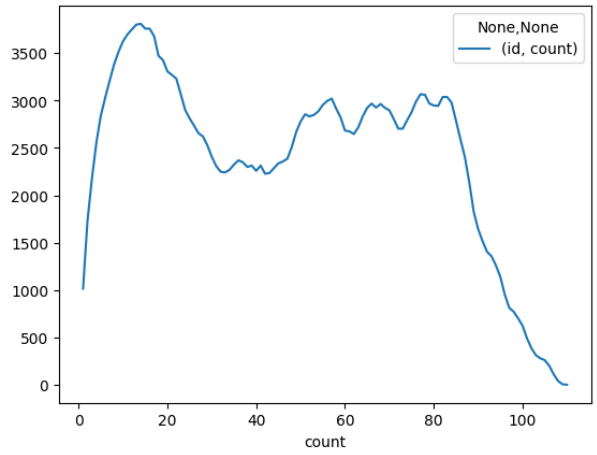
5. Data Analysis - Our Solution (Business Logic) - *data-analysis/airbnb/logic/*

<u>Scoring Airbnb Algorithm:</u>

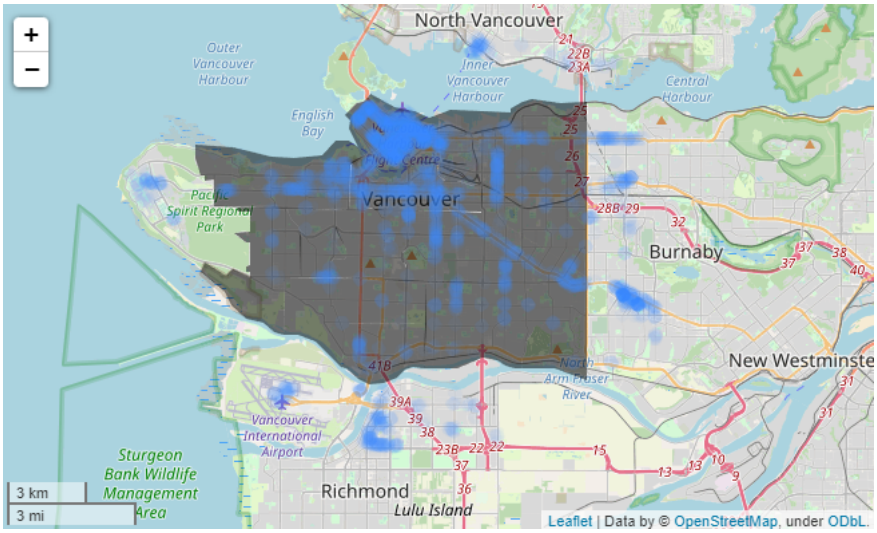
a. Into six extensive categories, divide collected datasets(places) in which travelers would be interested (done from data-cleaning).
   i. Entertainment
   ii. Food
   iii. Leisure
   iv. Tourism & Culture
   v. Public Transportation(Bus Stop & Skytrain station)
   vi. Shop

b. For every category - *airbnb_intersection.py*
   1) Add 300m buffer to each Point(coordinate pair: lat, lon)

   2) Find all intersections for boundaries of those buffers(where they overlap) and also count how many buffers are overlapped in each intersection



# of food restaurants for each intersection - although it was a bit right-skewed, we decided to use only those who are above average

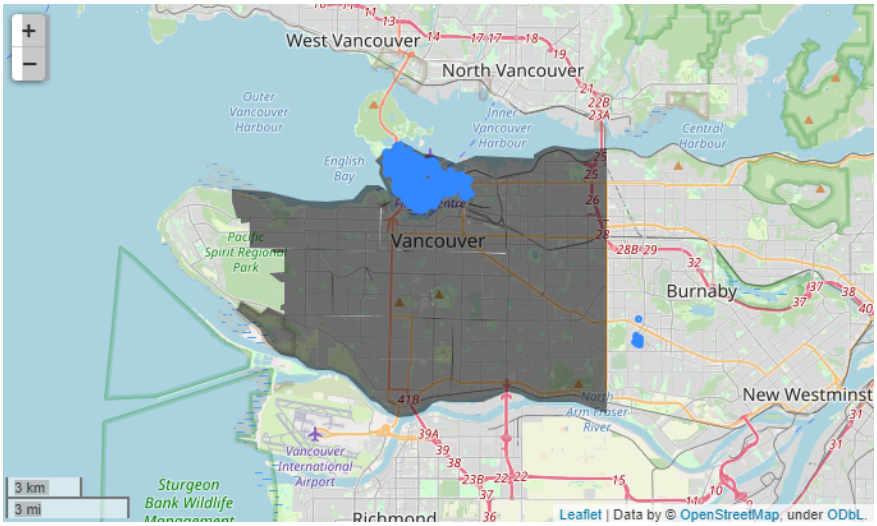


one example with food amenities - boundary buffer for all intersections (denser means a higher intersection count)

3) For each Airbnb, iterate through the intersection list and check if any intersection boundary contains the Airbnb inside. Add all 'count' values(or a scaled 'count' amount) for those intersections. O(N*M) operation where N is the number of intersection boundaries and M is the number of Airbnb

4) Save the 'count' score for all Airbnbs in a new column "category name" in the Airbnb dataframe.

5) MinMaxScaler is used to transform each type of score individually between 0 and 1.



one example with food amenities - Airbnb's only with food_intersection_score >= 0

c. Name the new Airbnb list with all category scores - *airbnb_score.csv*

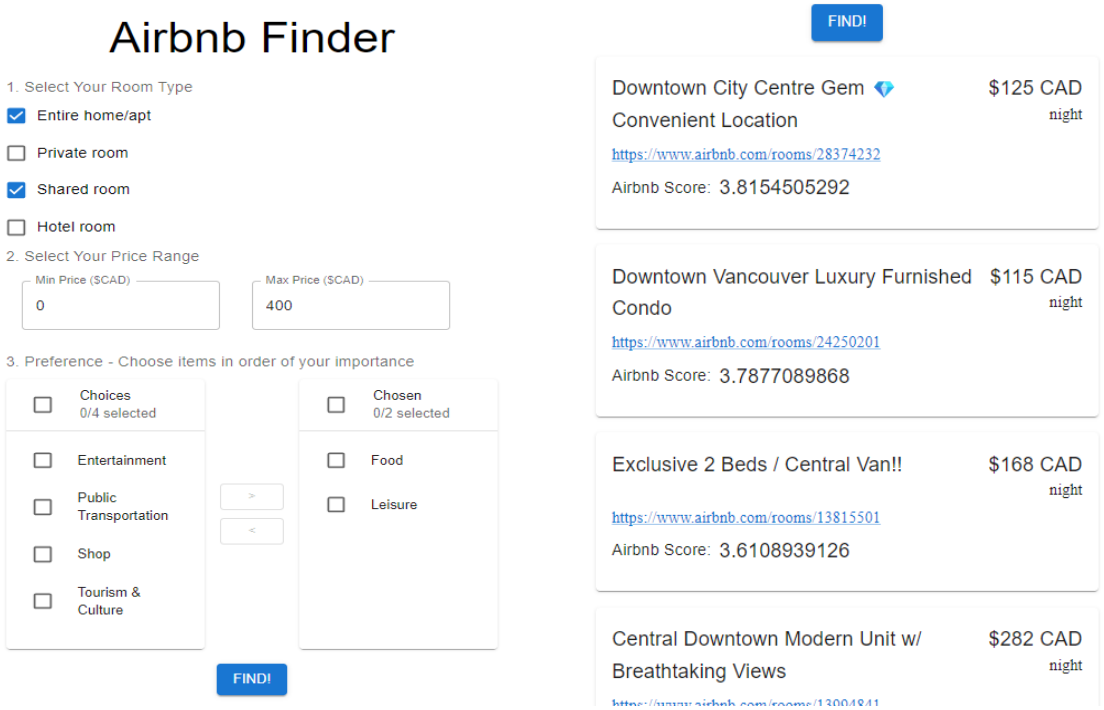
Final airbnb_score.csv's schema:

```
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 16 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Unnamed: 0                    5572 non-null   int64
 1   name                          5572 non-null   object
 2   listing_url                   5572 non-null   object
 3   neighbourhood                 3744 non-null   object
 4   latitude                      5572 non-null   float64
 5   longitude                     5572 non-null   float64
 6   price                         5572 non-null   float64
 7   room_type                     5572 non-null   object
 8   review_scores_rating          4760 non-null   float64
 9   entertainment                 5572 non-null   float64
 10  food                          5572 non-null   float64
 11  leisure                       5572 non-null   float64
 12  transportation                5572 non-null   float64
 13  shop                          5572 non-null   float64
 14  tourism                       5572 non-null   float64
 15  scaled_review_scores_ratings  4760 non-null   float64
dtypes: float64(11), int64(1), object(4)
memory usage: 696.6+ KB
```

6. Web

a. Frontend - *frontend/*

We created our own React web application to let users set their own preferences as input(room type, price, travel preference) and provide our solution (recommendation of Airbnb lists) as the output. Most of the logistics are in *src/App.js*. This is what the frontend looks like with the results:



b. Backend - *backend/*

FastAPI application hosts a REST API endpoint at api/airbnb_list which triggers *compute_airbnb* function at *src/airbnb_service.py.* This function takes in users' preference objects in json body and filters out Airbnbs depending on room type and price range. Then we weigh the scores of each travel place according to the user's order of importance. Order of importance is a List type where the lower indexed items have the higher priority. The weight starts at 1 and adds 0.25 per item to the list.

For example, given ["Food", "Leisure"], the weight is [1.25, 1] where food has a weight of 1.25 and leisure has a weight of 1. Similarly, given ["Shop", "Food", "Tourism & Culture"], the weight is [1.5, 1.25, 1]

Using the pandas dot() function, we calculate a weighted sum of travel place score. We then sum Airbnb's review score with the weight of 1. Finally, we sort the dataframe by descending order with the weighted sum field and return the top 20 high-scored Airbnb to the users.

7. Instructions on how to run our files locally

a. Data-analysis (Python version 3.10)
   Install the libraries specified in **2. Data-analysis-Tools** and run the python files.

b. Frontend - (Node version 18)
   1. `cd frontend` - Go to the frontend directory
   2. `npm install` - Install Node dependencies
   3. `npm start` - Start React app at http://localhost:3000

c. Backend (Python version 3.10)
   Dependencies: FastApi (uvicorn), Pandas
   1. `cd backend` - Go to the backend directory
   2. `pip install -r requirements.txt` - Install Python dependencies
   3. `python main.py` - Start FastAPI app at http://localhost:8000. You can test out the endpoint at http://localhost:8000/docs

8. Try out our Apps!

You can also try our web application here;
Frontend: https://airbnb-finder.up.railway.app/
Backend: https://airbnb-finder-backend.up.railway.app/docs

9. Limitations

a. Lack of Data
   1. If we can have a user's personal information or their booking history, we will be able to run some Classification ML algorithms to recommend our user an Airbnb based on them.
   2. Incorporate Yelp or Google rating scores into our logic:
      We had to pay for their API, pay 3rd party services, pay for 3rd party web scraper, or create a web scraper from scratch. We didn't have enough time for this data.
   3. Since we are using a pre-collected dataset of Airbnb lists, our output Airbnb list holds only the data when the lists were collected.

b. Better presentation/usability for our web service
   1. Incorporate React-leaflet or Google Maps so that the users can easily see where Airbnbs are located at.
   2. Show which places users might be interested to see. (This is where we could've used extra data left from our osm dataframes)
   3. Add more photos like Airbnb photos on our React App
   4. Categorize the places in more detail (e.g. Food can be divided into restaurants, cafes, etc. Restaurants can be divided into Japanese cuisine, Korean cuisine, etc)

c. Stronger Algorithm to find the "best" Airbnb:
Not sure how we can improve our current logic, but it'd be nice to spend more time investigating other recommendation systems

10. References:

idea of analyzing intersections of Points
https://towardsdatascience.com/optimising-the-location-of-food-delivery-couriers-using-geopandas-and-openstreetmap-e771433e5e9f

idea of scaling
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

11. Experience Summaries:

Jusung Park

- Collected datasets from online and adapted them to have only the necessary information
- Imported Pandas and Geopandas Python libraries and their attributes and methods to handle large datasets with real coordinates
- Applied Jupyter notebook kernel to visualize the coordinates and the map
- Built intersection counting scoring and algorithms that calculate all intersection points for a number of different coordinates and score all up to a single series

Jooyoung Lee

- Cleaned and organized the data to make the subsequent analysis consistent and accurate
- Designed frontend webpage using React JS to get the preferences from users
- Applied Pandas Python library for data manipulation to filter collected data

Hyeonyoung Park

- Architectured React frontend and FastAPI backend structures and hosted the services to Railway
- Downloaded, extracted, and collect node data from OSM XML file using PySpark and Pandas
- Implemented logic to filter Panda dataframe with users' inputs on the backend