IBM Data Science Capstone Project

By

Krishnan G. Raghavan

Course provided by Coursera and approved by IBM

2020

Table of Contents

Abbreviations, Notations and Variables

This section lists some of the commonly used abbreviations, notations and variables in this report.

Abbreviations:

condt. – continued

df. - Dataframe

Abstract

This project involves determining the optimum location for a customer, at a new address (town or city etc.), which best satisfies the customer's needs.

First, venue information is obtained from a third-party application (Foursquare). This data is cleaned, and then visualized to look for outliers. After the outliers are omitted, this information is clustered using K-means clustering, and the centroid of these locations is calculated. The cluster results and centroid is shown on a folium map. The address information near the centroid is also returned.

Chapter 1. Introduction

In today's world, people travel a lot, be it for business, leisure or personal matters. Often this travel happens to an unfamiliar place. The choice of stay location one makes often determines, to a large extent, the outcome of their trip; whether it was enjoyable or mediocre.

Several tools out there can help us look for specific venues at this unfamiliar place, such as coffee shops, hotels, convention centers, restaurants, etc. But these tools typically involve the user performing multiple searches across categories to find the best location that's within proximity to these venues. This involves a lot of time and effort.

The following project attempts to use Machine Learning on location data, to determine the best location choice in a timely manner

1.1 Business Case

Problem: How do I find the most suitable location to match the customer's needs?

Customer's needs: Customer is a vacation goer, heading for a week-long vacation to the city of 'Bend, Oregon'. The customer has specified the following interest categories during this holiday:

- National Park
- Nature Preserves
- Hiking and Biking
- Museums
- Thai as food of choice
- Yoga
- Gym (cardio and other physical exercises)

1.2 Such a Project is useful for...

The process used in this project can be applied to the following cases to facilitate finding the optimum location for their needs:

- A business executive looking for the ideal temporary residence (hotel) to host a conference meeting at another town
- A holiday-goer that wants to find the perfect stay in proximity of restaurants, health amenities, and recreation
- A family moving to a new town/city and wants to find a location in proximity to malls, restaurants, parks, schools, hospitals and libraries among other things

1.3 Project Overview

In this project, we explore various locations at a given address and find the optimum location corresponding to the customer's requirements. The salient features of this project are:

- A Machine Learning based optimum location finder
- Obtain the user's interest categories, then look for venues matching those categories at the target address
- Pre-Processing to remove extraneous locations that might skew the final result
- Find the centroid of the venues' locations thus providing the user the optimal location within close proximity of the venues as well as returning the address of the centroid location
- *Visual (Map) as well as text-based representation of the analysis results*

1.4 Overview of this Report

The following is an overview of the contents of this report:

- Chapter 2: Data – Obtaining data and overview of implementation
- Chapter 3: Methodology – Deeper look into data analysis
- Chapter 4: Results – Results of the analysis
- Chapter 5: Discussion – Important observations during process, improvements
- Chapter 6: Conclusions

Chapter 2. Data

In this chapter, we talk about what kind of data was used in the project, where it was obtained from and an overview of how it was used in this project to solve the problem.

2.1 Data Acquisition and Implementation

First, location information for our address (latitude and longitude) is obtained using the geocoding application '*Nominatim'* (https://nominatim.org/).

Next, venue data is obtained from a venue finder application called '*Foursquare*' (https://developer.foursquare.com/). A request query is used to obtain a 'json' formatted list of venues matching our criteria (for example: 50 venues within 20 kms. of Latitude and Longitude of 'Bend, Oregon' obtained from Nominatim). This data is then stored in a dataframe. Table 2.1.1 provides an overview of this dataframe.

Table 2.1.1: Sample table of results derived from foursquare contained in a dataframe

	Address	Venue	Venue Latitude	Venue Longitude	Venue Category	Category ID
0	Bend, Oregon	Des Chutes Historical Museum	44.055381	-121.316909	History Museum	4bf58dd8d48988d181941735
1	Bend, Oregon	Sunriver Observatory	43.885204	-121.447711	Science Museum	4bf58dd8d48988d181941735
2	Bend, Oregon	Petersen Rock Museum	44.203122	-121.262437	Public Art	4bf58dd8d48988d181941735
3	Bend, Oregon	High Desert Museum	43.966331	-121.343029	Museum	4bf58dd8d48988d181941735
4	Bend, Oregon	Toomie's	44.058513	-121.313588	Thai Restaurant	4bf58dd8d48988d149941735

This venue information can be displayed on a geo-map using folium (https://python-visualization.github.io/folium/#:~:text=folium%20makes%20it%20easy%20to,as%20markers%20on%20the%20map.), as well as further processed and analyzed using tables, box plots and folium maps, to determine the best location in proximity to most of the venues. Folium can be imported to Python as well.

Chapter 3. Methodology

This chapter discusses the process utilized to go from ‘Question to Answer’. It elaborates on how the inputs were framed, how data was extracted, what kind of analysis was done on the data and how the final result was obtained.

3.1 User Inputs

The following are the inputs provided by the user of this process:

- **addresses**: ‘Bend, Oregon’ – customer’s location of choice
- **categories** (list of Foursquare categories for the customer’s ‘interests’):
    - Museum - 4bf58dd8d48988d181941735
    - Thai Restaurant - 4bf58dd8d48988d149941735
    - Yoga Studio - 4bf58dd8d48988d102941735
    - Gym/Fitness Center - 4bf58dd8d48988d175941735
    - National Park - 52e81612bcbc57f1066b7a21
    - Nature Preserve - 52e81612bcbc57f1066b7a13
    - Trail - 4bf58dd8d48988d159941735
- **LIMIT** (limit of search query): 500 – a high number
- **radius** (in metres): 20,000 – spanning typically convenient traveling distance

Note that the ‘categories’ consist of Foursquare venue category codes for each category as this is what will be used to perform the search query.

3.2 Libraries

The following libraries were downloaded and utilized to process the data:

- **numpy** – standard Python array library
- **pandas** – Dataframe Python library, also heavily based on numpy
- **math** – Python library to perform several math operations
- **json** – Python library for JSON files
- **geopy and nominatim** – Application to obtain location of addresses
- **requests** – handle requests to external apps. like Foursquare
- **matplotlib** – for plotting analysis graphs like box, bar, scatter etc.
- **sklearn (Scikit-learn)** – for performing clustering
- **folium** – for plotting world maps

3.3 Obtaining Venue Information

Once the user inputs are defined and the libraries imported, the next step is to obtain information about the venues close to the address. This step is the precursor to obtain the final optimum centroid location. Note that Nominatim was used to obtain an initial latitude and longitude location for the address. The following steps constitute the overall process:

- Obtain address location using Nominatim, in our case it was (*Latitude*: 44.0581728, *Longitude*: -121.3153096).
- Enter the Foursquare user credentials (client id, client secret, version)
- Generate the 'search' request url:
  'https://api.foursquare.com/v2/venues/search?categoryId={}&client_id={}&client_secret={}&ll={},{}&v={}&radius={}&limit={}'
  The {} can be filled using a Python 'format' statement

- Define a function to request the venue information from Foursquare, then interpret the json results to make a dataframe of the location and venue information with the following fields: *'Address', 'Venue', 'Venue Latitude', 'Venue Longitude', 'Venue Category', 'Category ID'*

- Implement the function(*getNearbyVenues*) to obtain a COMBINED dataframe of the results for every 'category'

The resulting dataframe looks like in Table 3.3.1 and had a size of 129x6.

Table 3.3.1: Initial Foursquare Results Dataframe (.head() results)

	Address	Venue	Venue Latitude	Venue Longitude	Venue Category	Category ID
0	Bend, Oregon	Des Chutes Historical Museum	44.055381	-121.316909	History Museum	4bf58dd8d48988d181941735
1	Bend, Oregon	Sunriver Observatory	43.885204	-121.447711	Science Museum	4bf58dd8d48988d181941735
2	Bend, Oregon	Petersen Rock Museum	44.203122	-121.262437	Public Art	4bf58dd8d48988d181941735
3	Bend, Oregon	High Desert Museum	43.966331	-121.343029	Museum	4bf58dd8d48988d181941735
4	Bend, Oregon	Toomie's	44.058513	-121.31588	Thai Restaurant	4bf58dd8d48988d149941735

3.4 Pre-Processing

Before using the data, the dataframe has to be cleaned of unsuitable values like 'NaN's. This was done in the following steps:

1. Drop NaN values and set 'Venue' column as Index. Note the shape of the resultant dataframe (in our case it is 129x5)

2. Print the number of unique categories and sub-categories (28)

The resultant dataframe is shown in Table 3.4.1.

Table 3.4.1: Pre-processed Dataframe (.head() results)

	Address	Venue	Venue Latitude	Venue Longitude	Venue Category	Category ID
0	Bend, Oregon	Des Chutes Historical Museum	44.055381	-121.316909	History Museum	4bf58dd8d48988d181941735
1	Bend, Oregon	Sunriver Observatory	43.885204	-121.447711	Science Museum	4bf58dd8d48988d181941735
2	Bend, Oregon	Petersen Rock Museum	44.203122	-121.262437	Public Art	4bf58dd8d48988d181941735
3	Bend, Oregon	High Desert Museum	43.966331	-121.343029	Museum	4bf58dd8d48988d181941735
4	Bend, Oregon	Toomie's	44.058513	-121.313588	Thai Restaurant	4bf58dd8d48988d149941735

3.5 Exploratory Analysis

Here we look at the data and determine any bad values or outliers. This involves understanding the data in both boxplot and folium formats.

Initial Folium Map: First, we look at a folium map to get a general idea of the area, and the distribution of the venues. We can also calculate the initial guess of the centroid at this stage and plot it on the folium map. The folium map of the centroid, location and venues is shown in Figure 3.5.1. The centroid for this stage is obtained as (Latitude: 44.04205, Longitude: -121.33228, Address: 1441 SW Chandler Ave (14th and Colorado)). The legends are added in separately for ease of readability.

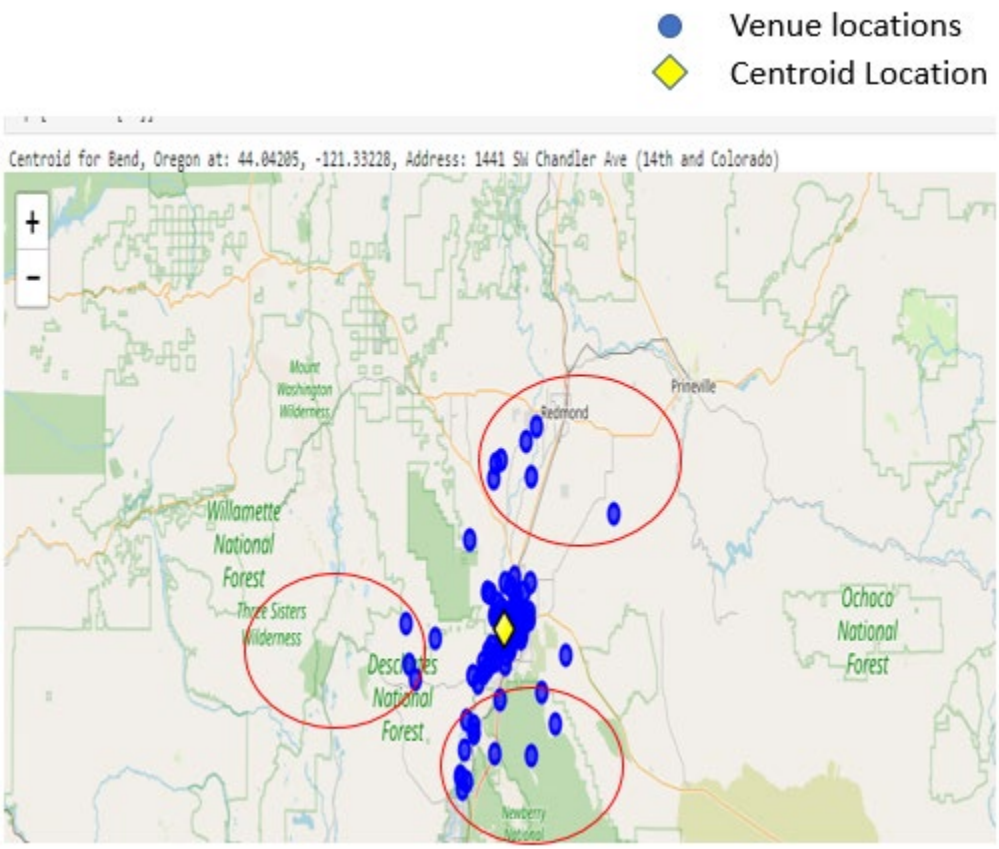

Figure 3.5.1: Folium Map for venue locations (Red Circular regions indicate possible outliers)

Box Plots: We obtain Box Plots for the latitude and longitude of the venues, grouped by category. The plots are grouped by category so that when we identify outliers, we do not inadvertently dismiss an entire category by omitting outliers. Figure 3.5.2 shows the plot for one category. Similar plots are obtained for each category. The complete set of plots can be found in 25.

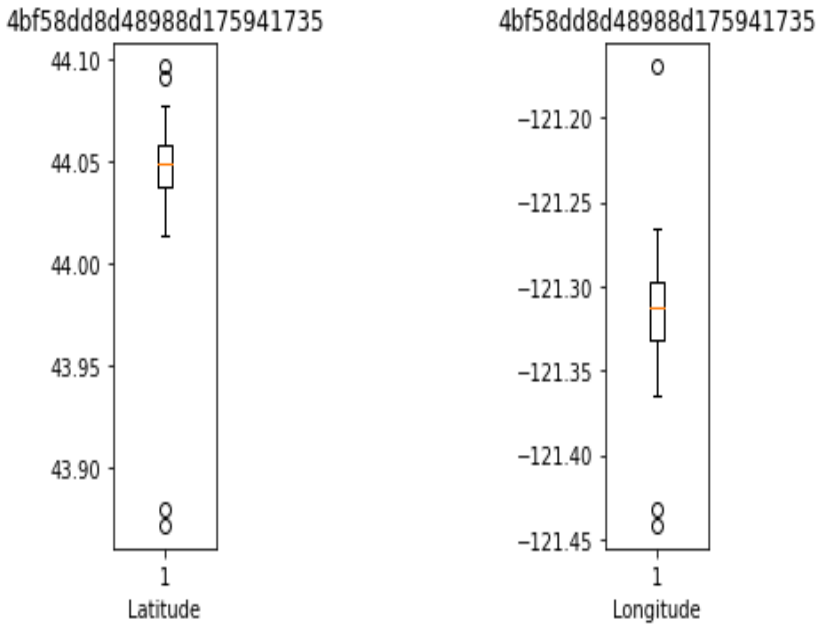

Figure 3.5.2: Box plots for venues of category 4bf58dd8d48988d175941735 (Circles indicate outliers).

Removing outliers: Since our dataset is seen to have outliers, it is best to remove these from the dataframe so that the outliers do not overtly skew the final result. The algorithm to do so is described below:

- Algorithm to remove outliers:

Initialize ‘flag’ to 1
while flag is 1
    set flag to 0
    for each category:
        a. create boxplot container (dictionary) for category name+latitude and category name+longitude
        b. reference the boxplot objects in the dictionary to obtain outliers arrays for ‘category+latitude’ and ‘category+longitude’
        c. if length of either outlier arrays > 0:

set flag to 1

d. for the outlier arrays:

Remove df rows corresponding to the latitude/longitude + category

The above process should remove the outliers from the dataframe. Note that that the process is repeated multiple times, till we have no outliers left, since each iteration the boxplot is evaluated and checked if any new outliers are produced. This is the purpose of the 'flag' variable.

Determine the shape of the resultant dataframe (should be less than the initial), in our case it is 105x5 suggesting the outliers could be removed. Verify by plotting the box plots again.

Verifying no outliers: Re-plot the box plots as in Figure 3.5.2 to check for any outliers. The plots for the corresponding 'category' are shown in Figure 3.5.3.

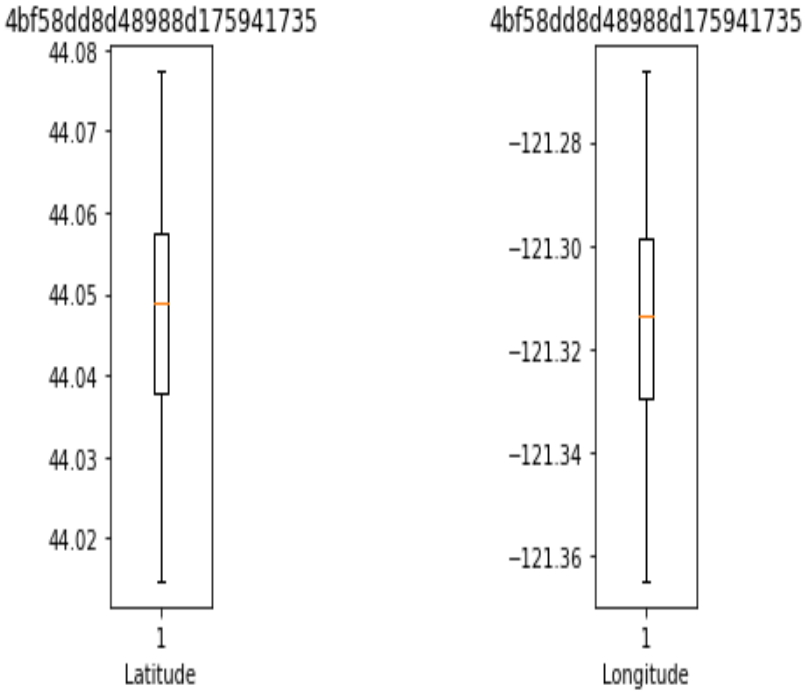

Figure 3.5.3: Box plots for venues of category 4bf58dd8d48988d175941735 (Circles indicate outliers).

Looking at Figure 3.5.3, we see that all outliers have been removed. Please refer to Appendix A for the complete set of plots corresponding to Figure 3.5.3.

Folium Plot without outliers: Next we re-plot the Folium map as in Figure 3.5.1, which is shown in Figure 3.5.4.

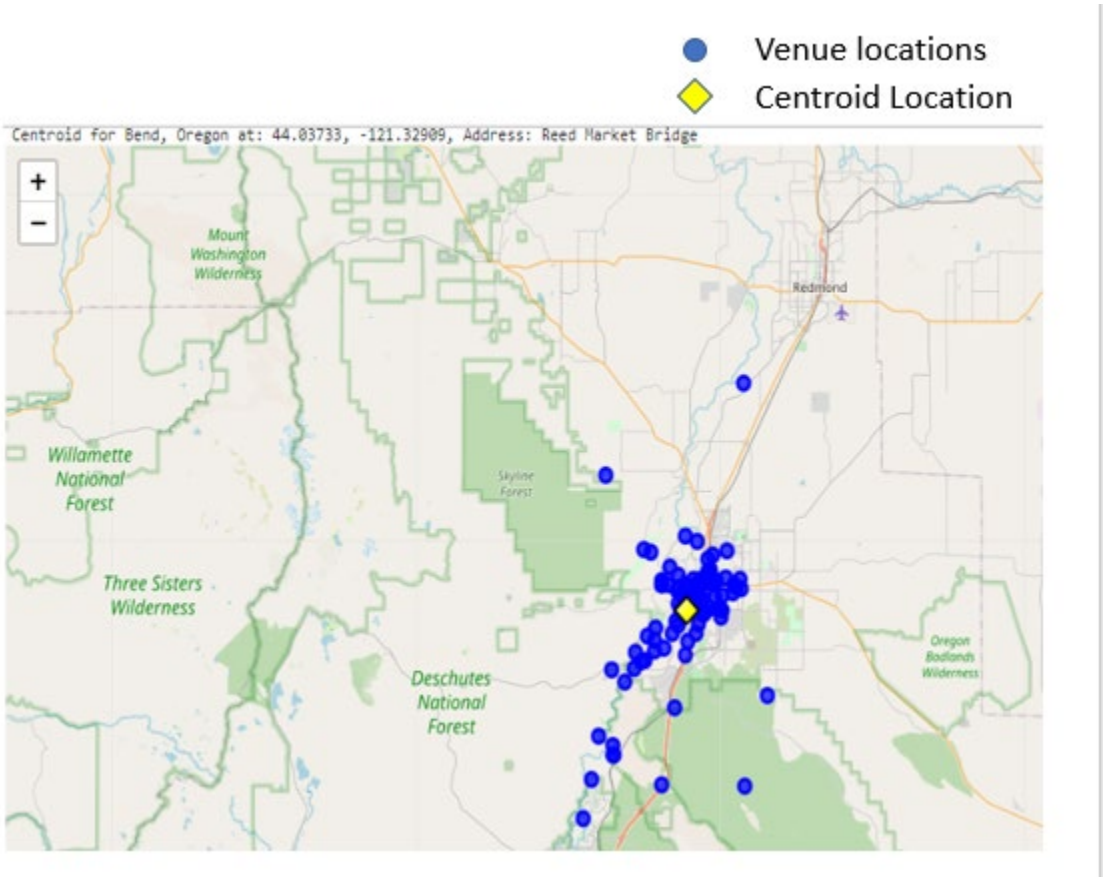

Figure 3.5.4: Folium Map for venue locations

Comparing to Figure 3.5.1, we see that a lot of the outer points are reduced. We also obtain a new Centroid location, (Latitude: 44.03733, Longitude: -121.32909, Address: Reed Market Bridge). Note that the 'Address' has also changed.

Calculate Centroid Shift: It is interesting to see how much the location of the centroid has shifted before and after this process of removing outliers. For this, we use a user-defined function '*geodistance'*. This function uses the mathematical *haversine* function to determine the curved distance between two points on the curve (in our case, the earth's curvature). It takes the latitudes and longitudes of the coordinates as list inputs and outputs the distance as float. The output obtained after running this function for our case, is:

Distance shift in Centroid before and after removing outliers is: **0.63kms**.

Distance shift in Centroid for Bend, Oregon before and after processing is: **2.78kms**.

Therefore, we see that there is about 0.6 kms or 0.4 miles of shift after removing outliers, and about 2.8 kms. or 1.72 miles of shift in location before and after the entire process thus far.

3.6 Encoding and Clustering

Now that we have our most reasonable estimate for the centroid, and we also have the corresponding venue information in a dataframe, we can proceed to display the venues grouped by categories on a Folium map. However, the first step in this process is to obtain the groupings, or 'clusters'. This process consists of two steps:

- ***Encoding***: We add columns corresponding to each 'category' and a '1' value if the corresponding entry belongs to that category, '0' otherwise.
- ***Clustering***: We use some clustering algorithm to group our points into clusters according to categories. In our case, we use a popular clustering algorithm called *K-means* (https://en.wikipedia.org/wiki/K-means_clustering). This is implemented in Python 3.6. using scikit-learn.

Encoding: The encoding is done according to category ID. The .head() is displayed in Table 3.6.1 to ensure that all categories are included and the dataframe is as expected.

Table 3.6.1: Dataframe encoded by category

	Address	Venue Latitude	Venue Longitude	Venue Category	Category ID	4bf58dd8d48988d102941735	4bf58dd8d48988d149941735	4bf58dd8d48988d159941735	4bf58dd8d48988d175941735	4bf58dd8d48988d181941735	52e81612bcbc57f1066b7a13	52e81612bcbc57f1066b7a21
Venue												
Des Chutes Historical Museum	Bend, Oregon	44.055381	-121.3169	History Museum	4bf58dd8d48988d181941735	0	0	0	0	1	0	0
Sunriver Observatory	Bend, Oregon	43.885204	-121.4477	Science Museum	4bf58dd8d48988d181941735	0	0	0	0	1	0	0
Petersen Rock Museum	Bend, Oregon	44.203122	-121.2624	Public Art	4bf58dd8d48988d181941735	0	0	0	0	1	0	0
High Desert Museum	Bend, Oregon	43.966331	-121.343	Museum	4bf58dd8d48988d181941735	0	0	0	0	1	0	0
Toomie's	Bend, Oregon	44.058513	-121.3136	Thai Restaurant	4bf58dd8d48988d149941735	0	1	0	0	0	0	0

Table 3.6.2: Clustered and encoded Dataframe (.head(2))

	Address	Venue Latitude	Venue Longitude	Venue Category	Category ID	4bf58dd8d48988d102941735	4bf58dd8d48988d149941735	4bf58dd8d48988d159941735	4bf58dd8d48988d175941735	4bf58dd8d48988d181941735	52e81612bcbc57f1066b7a13	52e81612bcbc57f1066b7a21	Cluster Labels
Venue													
Des Chutes Historical Museum	Bend, Oregon	44.055381	-121.3169	History Museum	4bf58dd8d48988d181941735	0	0	0	0	1	0	0	**3**
Sunriver Observatory	Bend, Oregon	43.885204	-121.4477	Science Museum	4bf58dd8d48988d181941735	0	0	0	0	1	0	0	**3**

K-means Clustering: In this section, a brief overview of the K-means algorithm is provided to give context to the clustering method used. The algorithm is as follows:

- An initial set of means (cluster centroids) is chosen. In our case we want to cluster by number of categories so the number of means we choose can be equal to the number of categories (=7). Note that some categories may have no data or venue locations. Also note that we are clustering by venue category and not location, so that we can see the categories spread on the map.

- The distance of all venue points to each mean is calculated, and initial clustering is done. In our case, each venue is clustered according to where it lies w.r.t. all category fields.

- The cluster centroids are moved to the center of their respective clusters, and the distance from each point to each centroid calculated again. Then the clusters are redrawn. This process is repeated till the centroids do not move significantly. In our case, the process might not need multiple iterations since the clusters are uniquely defined.

- A link to an animation showing the K-means clustering process: https://en.wikipedia.org/wiki/K-means_clustering#/media/File:K-means_convergence.gif

Clustering: In this project, the result of the clustering is a dataframe as shown in Table 3.6.2. The last column shows the cluster labels. Note that although the clusters are labelled by category ID, the labels do not necessarily correspond to the category ID. However, each label corresponds to a unique category ID.

Chapter 4. Results

In this section, we visualize and look at the results of the process so far.

4.1 Folium Map

A folium map of the results, as before, is shown in Figure 4.1.1.

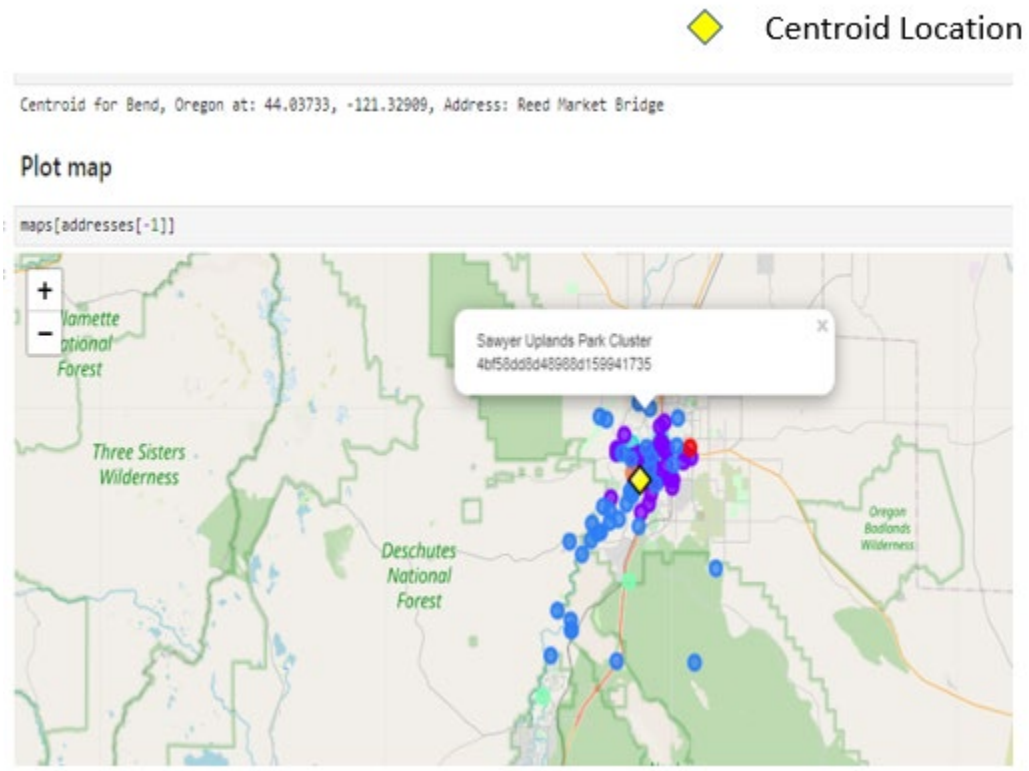

Figure 4.1.1: Folium Map Result for venue locations

The venues are colored by category.

4.2 Location Information

The final location information is:

- Latitude : 44.03733, Longitude: -121.32909
- Address: Reed Market Bridge

Chapter 5. Discussion

This chapter summarizes the important findings, learnings and remarks of the project and avenues for improvements.

5.1 Remarks

Some observations and remarks are:

- Care should be exercised for choosing the appropriate "radius" of search. Too wide can result in lot of scattered outliers and possibly clusters from other locations that could skew the result. Too small a radius may miss out on major points of interest. In our case, 20kms seemed to be a decent number.

- The 'Result' location can be used as a reference to look for hotels or other residence near that area.

5.2 Future Work/Improvements

- Code matching to category name instead of category ID. For example, "Trail" should be able to refer to "4bf58dd8d48988d159941735" in the database. This would require supervised learning to achieve, but we'd be able to work with and display more familiar objects

- Features can be added to automatically search for hotels, etc. closest to the result location

- After a few trials, it might be possible to automate the process, so that the data processing and analysis steps are performed automatically, thus speeding up the "question to result" dramatically. It might also be possible to provide this as a standalone application to a lay user, after making them familiar with the categories. This might be easier with category and name matching

Chapter 6. Conclusions

Over the course of this project, we have been successful at obtaining and interpreting the customer's requirements, and providing the best answer to their problem. To summarize:

- The project was able to find an appropriate location in close proximity to the venues as per customer's requirements
- Data extraction, pre-processing, analysis and result stages were successfully performed.
- Improvements have been recommended.
- *Hope the customer has a good vacation!*

The Python 3.6. code can be found at:
https://github.com/kgraghav/Coursera_Capstone/blob/master/Week4_5-bc.ipynb

Appendix A. Box Plots of Venues

This appendix includes the Box Plots for the pre and post outlier processing. Figure A.1.a through g provides the plots for pre-outlier correction, for each category, and Figure A.2.a through g for post outlier correction.

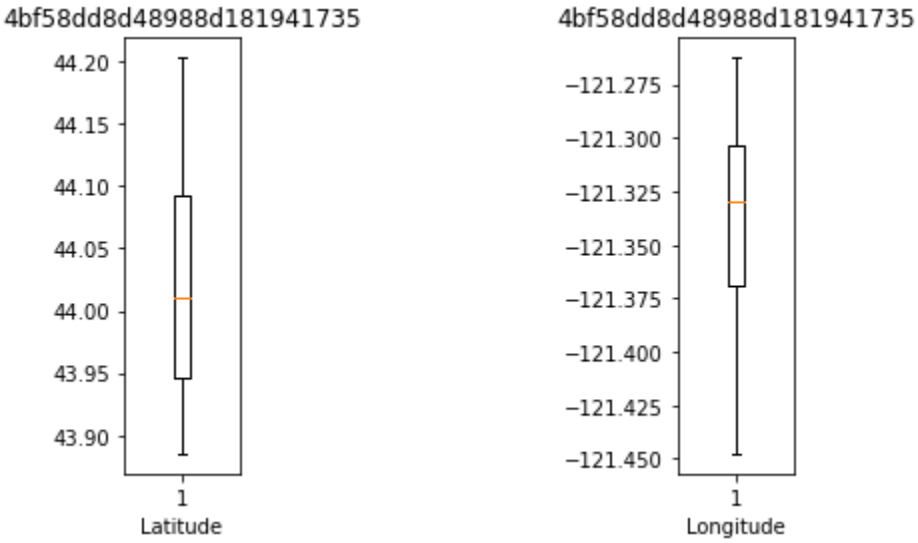

Figure A.1.a: Pre-outlier-processed Box-plots

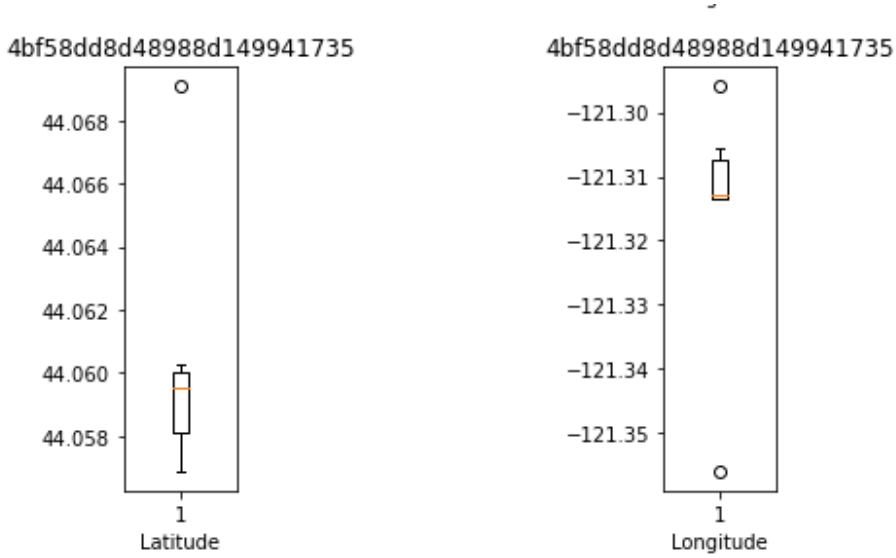

Figure A.1.b: Pre-outlier-processed Box-plots

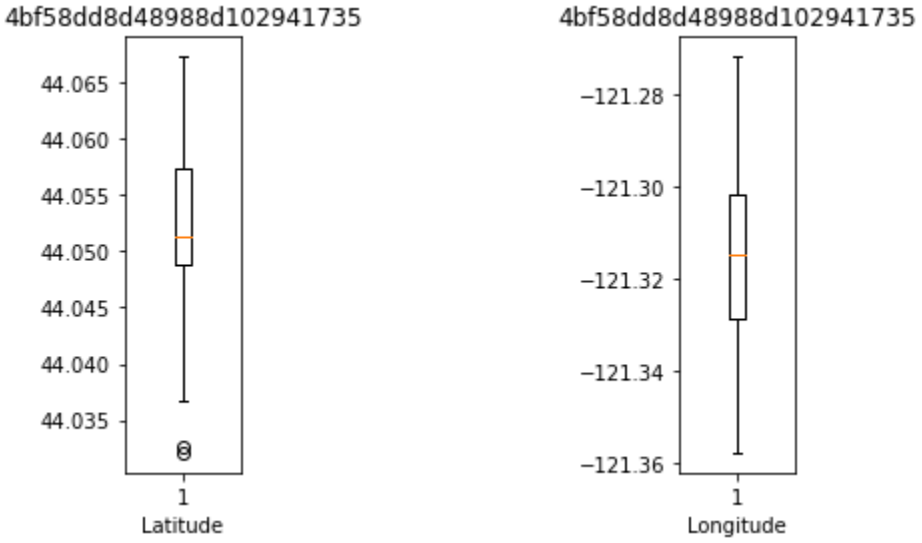

Figure A.1.c: Pre-outlier-processed Box-plots

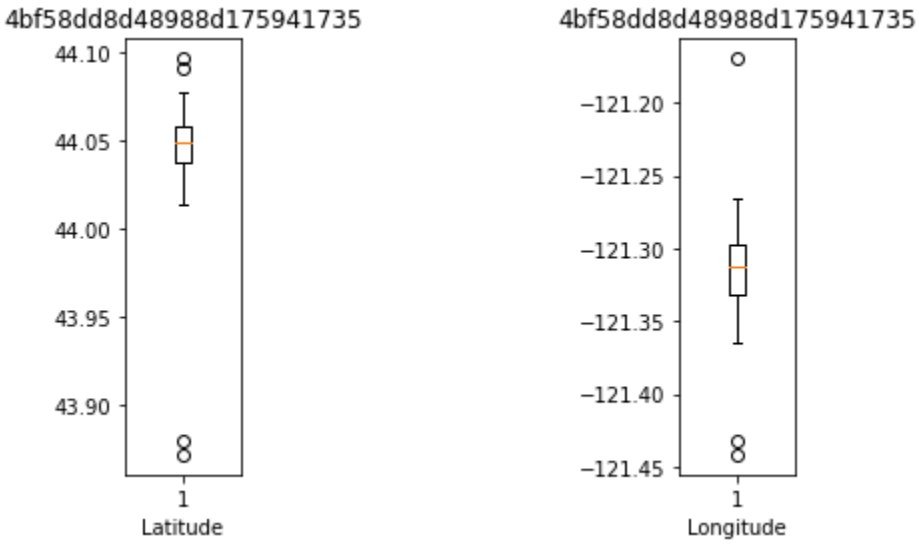

Figure A.1.d: Pre-outlier-processed Box-plots

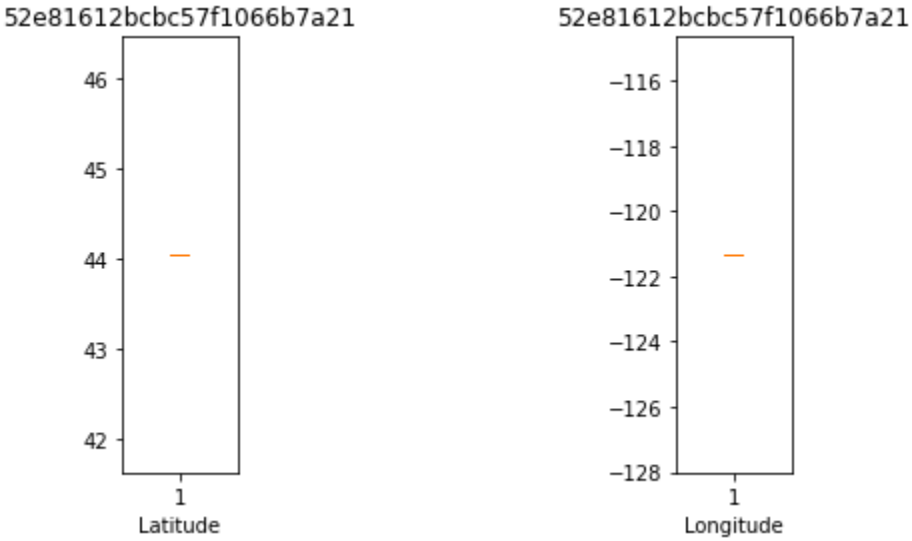

Figure A.1.e: Pre-outlier-processed Box-plots

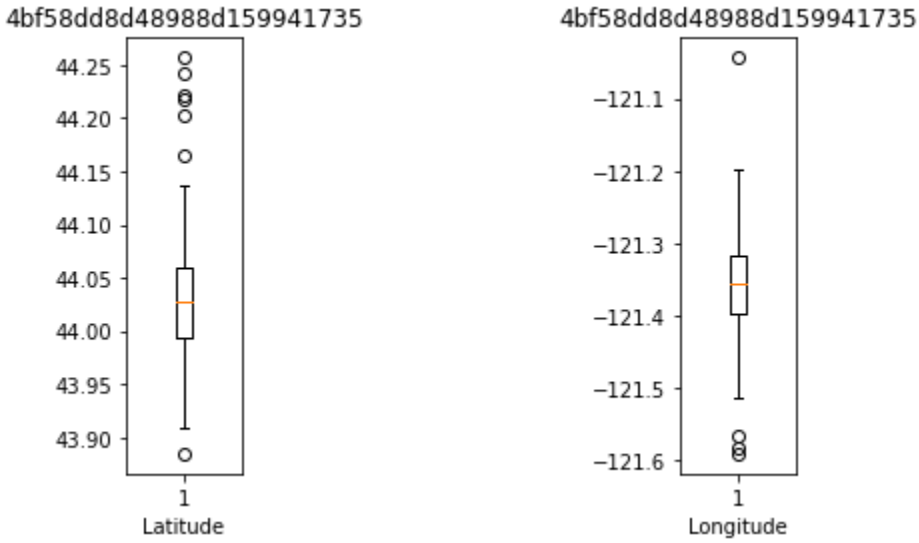

Figure A.1.f: Pre-outlier-processed Box-plots

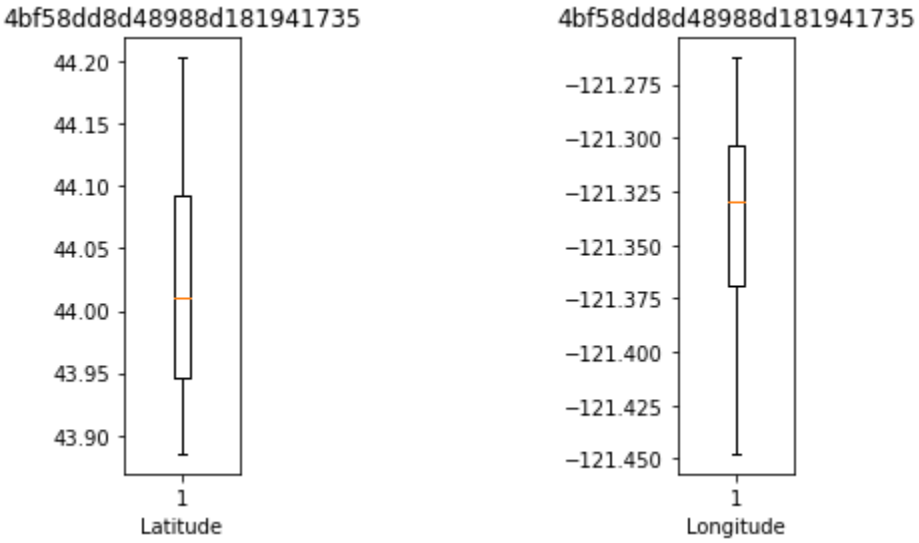

Figure A.2.a: Outlier-processed Box-plots

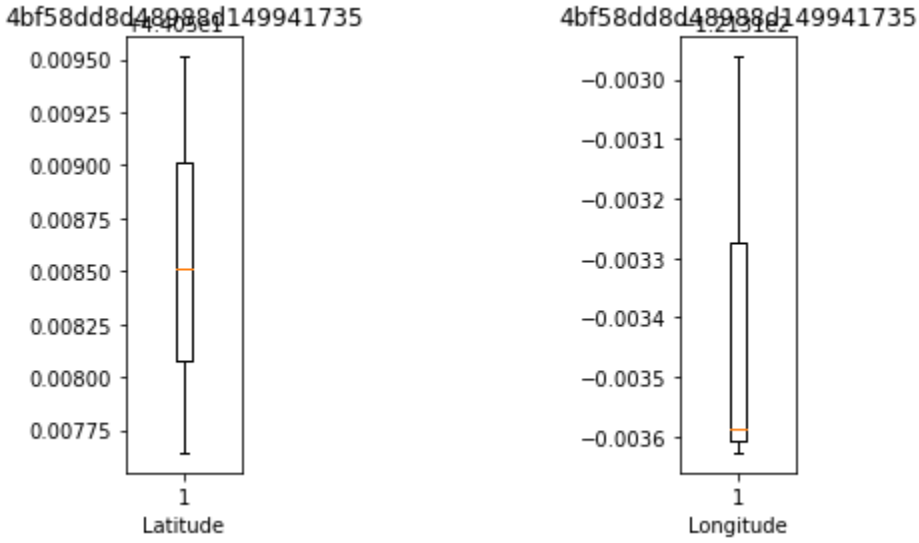

Figure A.2.b: Outlier-processed Box-plots

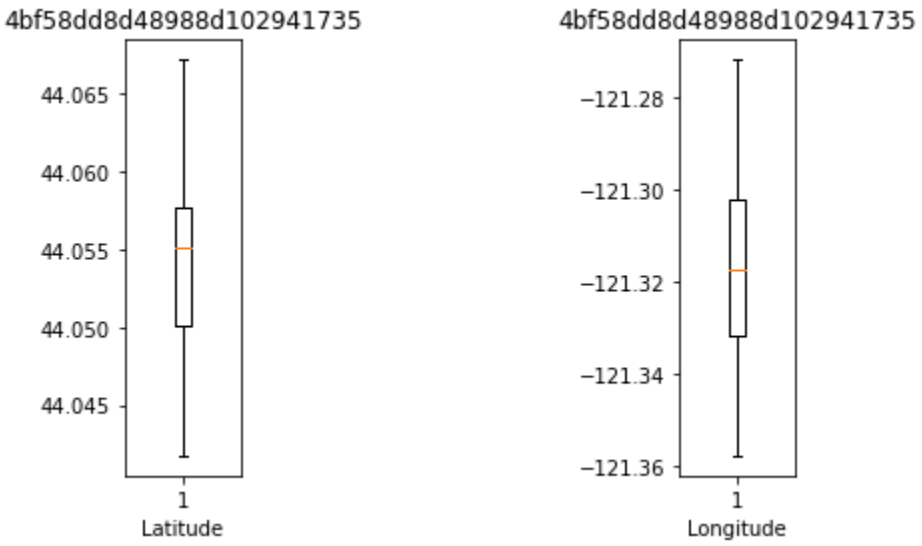

Figure A.2.c: Outlier-processed Box-plots

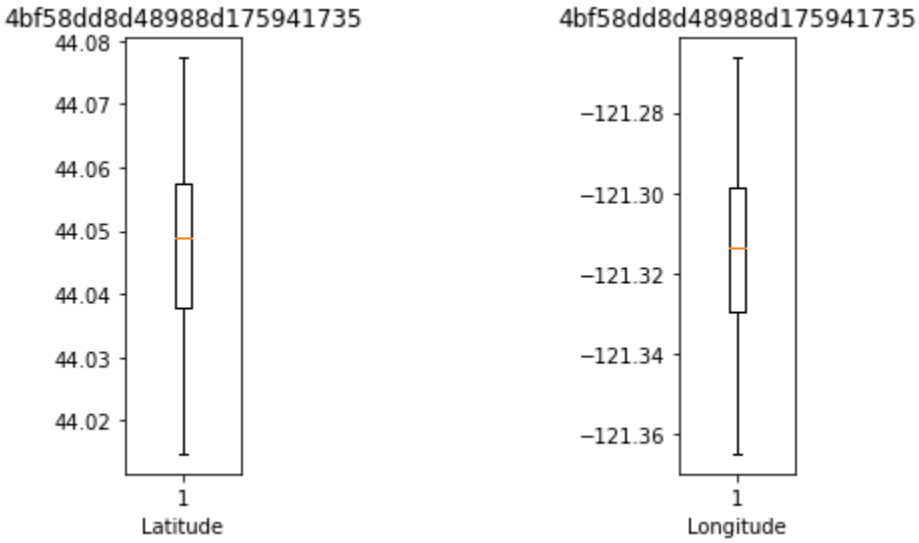

Figure A.2.d: Outlier-processed Box-plots

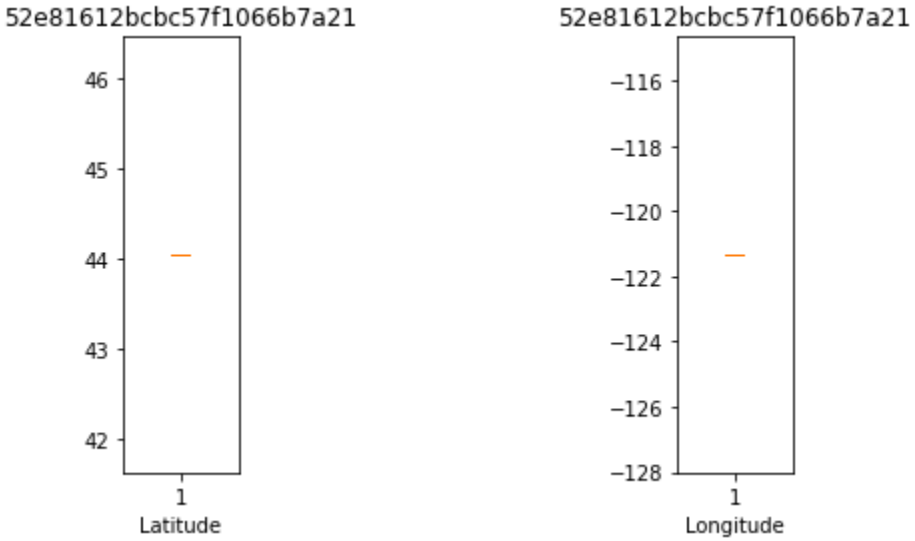

Figure A.2.e: Outlier-processed Box-plots

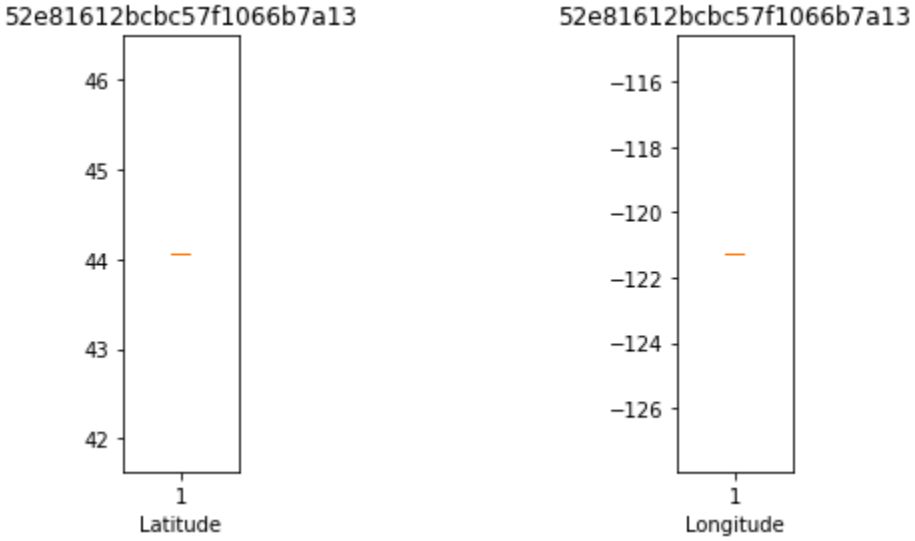

Figure A.2.f: Outlier-processed Box-plots

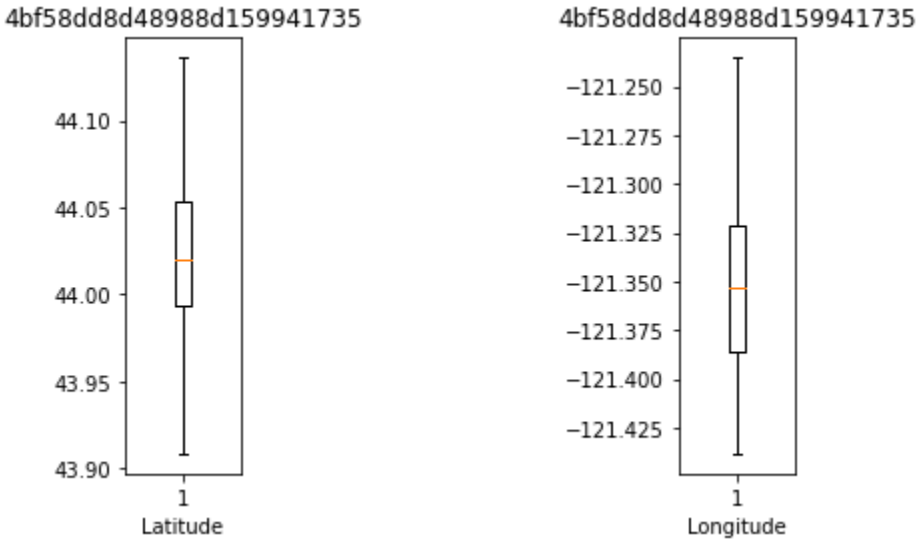

Figure A.2.g: Outlier-processed Box-plots