**Project Report – EQ Works**

This document highlights and explains all the methods undertaken to pre-process, analyze and model the sample POI data. The document is structured according to the assigned tasks. All the analysis was carried out using Py-Spark and Pandas.

The project can be run as a spark job. I have attached the .txt files of spark submit command and requirements for the same.

**Clean up:**

According to the problem statement, all the data points having the same time stamp and GeoInfo (Latitude and Longitude) are considered suspicious and should be removed from the analysis.

In order to achieve this, the three suspicious fields (Timestamp, Latitude and Longitude) were concatenated and duplicate entries were removed comparing the concatenated result of each record. Out of the 22025 data points in the DataSample.csv file, 2026 data points were removed leaving 19,999 original data points.

In the POIList.csv file, there were 4 entries with a redundancy between POI1 and POI2 and therefore the redundant entry (POI2) was removed.

A screenshot of a cell phone

Description automatically generated

**Labelling:**

1. As a result of the cleanup, all data points were mapped to 3 instead of 4 POIs. For labelling, the datasets were joined, first using the cross-join function, which resulted in a total of 59997 points (19999\*3).
2. Next, the distance (in kilometers) between the Latitude, Longitude pair of the original points and the POI points were calculated using the Haversine formula.
3. The data was then filtered to retain only the requests having minimum distance for each ID, this again reduced the data points back to 19999. This meant that each data point now contained a POIID which will serve as a label.

**Analysis:**

1. As per requirement, the average distance and standard deviation between each POI center and its assigned requests were calculated using group by aggregations. The standard deviation for POI4 looked fishy and therefore, an outlier analysis was required.

A screenshot of a cell phone

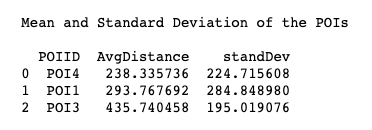
Description automatically generated

It was noted that the requests assigned to POI 1 had the least average distance, whereas POI3 had the least standard deviation. To check the spread of data, a visualization of data points was created on the world map.

A close up of a map

Description automatically generated

Clearly, the visualization showed presence of outliers. A function was used to used to remove the outlying points. The summary stats after outlier removal presented a much clearer picture.



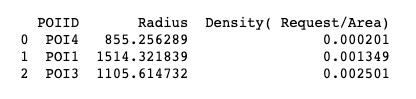
1. The next analysis involved drawing a circle with POI coordinate as center with a radius that would include all the data points assigned to it. Intuitively, it meant that the request with the maximum distance would be the radius of each POI centers.
2. In order to achieve this, **Matplotlib** was used, to generate the desired visualizations. To generate the great circle over the Canadian map, I borrowed a code from <http://www.geomidpoint.com/destination/calculation.html>. This code returned a bunch of latitude, longitude pairs which was used to generate the circle.

A close up of a map

Description automatically generated

Two circles are seen overlapping, as it includes the extreme points assigned to it. The cluster centers 3 and 4 represent Montreal and Moncton, NB. Since Montreal cluster includes Toronto and GTA requests as well, there is a sizeable overlap between POI 3 and POI 4. This, however, has nothing to do with the request assignment.

1. Using aggregation, the radius and density (requests/area) for the three POIs were calculated. POI3 was the densest cluster followed by POI1 and POI 4.

****

**Data Science Tasks**

My initial understanding of the problem was to determine the quality of clusters and therefore thought that using an internal measure such as Silhouette coefficient or Dunn Index would be a good starting point. However, such measures are for validating the clustering quality and not useful for individual cluster differentiability.

I did some readings to gain a deeper insight about geospatial POI data and its applications.

Based on the understanding of the process and data, there are two interesting features that can be used for building the model.

1. **Temporal feature**: All the requests have a corresponding timestamp. Based on the assumption that the POI is a potential business source, which is to be recommended to people based on proximity to the POI location, the temporal element becomes critical. As a part of feature engineering, weights were assigned to different time slots such that, requests during the peak hours would carry more weight than a request during off peak hours.
2. **GeoSpatial feature:** The distance becomes very significant while recommending POIs to a customer. Lesser the distance of a request from a POI, greater the likelihood of a request responding to a POI suggestion.

The next step was to check the distribution of data for all the three POIs. Here, Boxplot was used to visualize the distribution of data.

A screenshot of a video game

Description automatically generated

As can be seen from the figure, the distribution of the data greatly varies between the three clusters, and the range for outliers is remarkably different. What may be considered a normal point for POI3 may be an outlier for POI4.

**Building the model**

While I realized that the inverse function of distance should be used in the model, the [blog](https://gisgeography.com/inverse-distance-weighting-idw-interpolation/) on Inverse distance weighting interpolation, gave me an idea to enhance the overall influence of nearby points (to a POI center), for a better performing model.

A similar intuition was followed for the temporal component. Some papers discussed the effect of peak hours for POI recommendation. I incorporated the idea by building a time metric as a function of request time such that, requests coming in at peak hours will carry more weight and requests during non-peak hours will be scaled down.

I also considered factors such as, proportion of points within an area, to use that as a feature while building the model. However, those effects will automatically be captured by the distance measure and therefore, to avoid multicollinearity, I neglected them.

Algorithm to build the model:

Step 1: Filter the dataframe with the POI ID.

Step 2: Sort the filtered dataframe by distance in ascending order.

Step 3: Apply the formula score = 1/np.power(df.distance,2))\*np.power(df\_poi.Time\_Metric,2)

Step 4: Return the mean of the score

Step 5: Repeat Steps 1 to 4, for all 3 POI ID and store the results to a list.

Step 6: Apply Min Max Scaler to the list with (-10,10) as the upper and lower limit to scale the values between the limits.

Model Equation: **(1/distance)^2 \* (Time\_Metric)^2**

**Results**:

Based on these equations, the relative scores of the three POIs are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | POI 1 | POI 3 | POI 4 |
| Score | 10 | -6.3 | -10 |

Analyzing POI 1 cluster, we can see from the boxplot, that 75 % of data points are within a 250 KM radius.

Also interesting is the fact that 25 % of the requests (2400 requests) lie within a 15 km radius, as opposed to 470 km and 73 km for POI 3 and POI 4 respectively, therefore POI 1 is the most popular.

Between POI 3 and POI 4, it can be seen that there are only 100 requests at the source (Moncton) for POI 4 as opposed to 400 (Montreal) for POI 3. Since we are using an average measure, POI 3 performs better.