CS 6140: Intermediate Report - Analyzing Chicago Taxi Trips Behaviour via Data Mining

Group: Shaurya Sahai, Abishek Krishnan, Sushmitha Sunkurdi Nataraj

Dataset

Our data was collected from Kaggle [https://www.kaggle.com/chicago/chicago-taxi-trips-bq] which in turn is warehoused in BigQuery, under the table [bigquery-public-data:chicago_taxi_trips.taxi_trips].
This is being collected by the City of Chicago (the regulatory agency) from 2013 to the present. The agency does so through periodic reporting by two major payment processors believed to cover most taxis in Chicago. Currently, it has information about over 100 million Chicago taxi rides which includes geospatial locations of pickup, drop-off, fare, tip and payment method, etc.

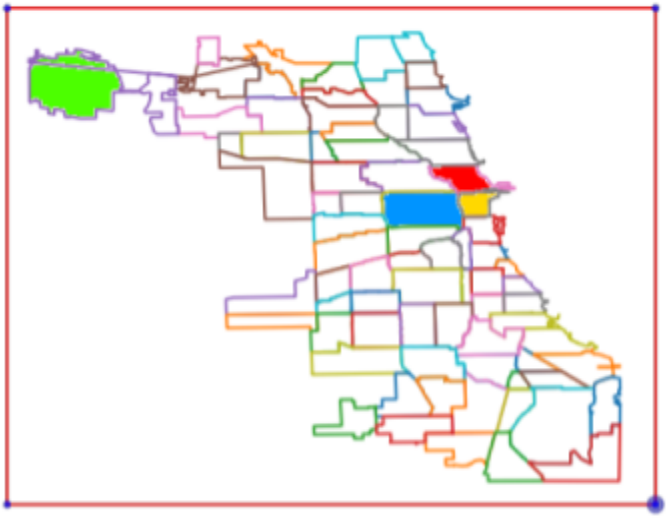
Progress towards proposed goals

Our proposed goal was to mine interesting insights to understand passenger behavior, help the taxi aggregators improve their service. We wanted to approach this problem by mining various relationships in the dataset using the techniques taught in the class and useful approaches that have been used in the wild. We have used clustering as a tool to find out the best locations for taxis to aggregate. In the future, we would like to explore more such relationships and devise methods that be used in the real world. The following sections give an overview of what we tried, what worked and what didn't.

Data Exploration

Data exploration is an essential task when working with a huge dataset. We have used Data visualization as a technique in laying out relevant attributes that would help us in exploring the data in an effective manner.

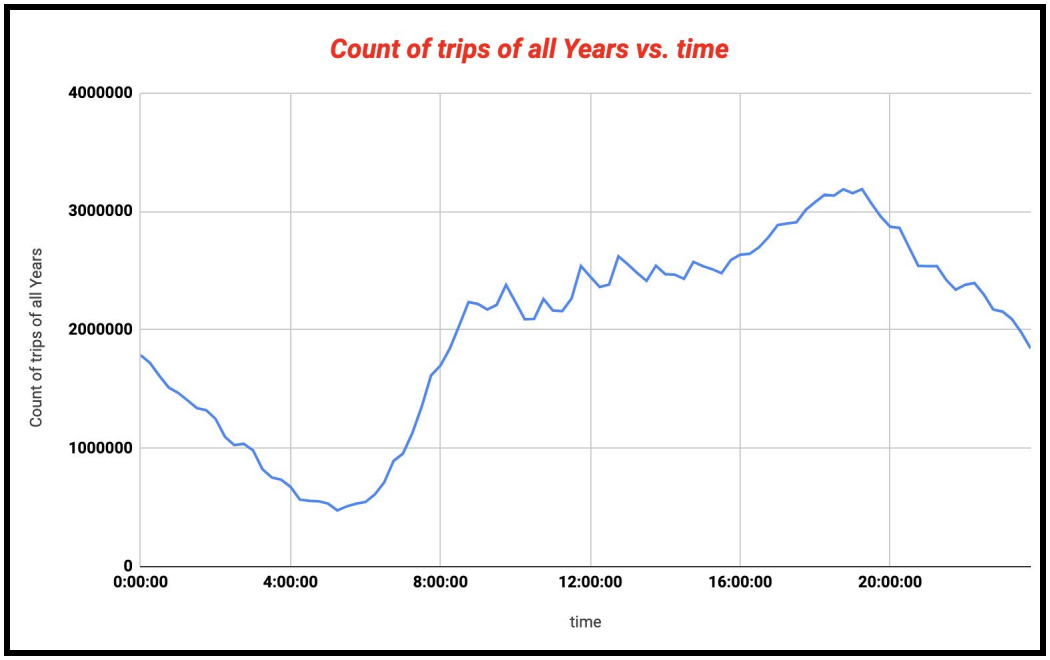
a) The following plot shows the top 10 busiest routes during 2019.



City of Chicago: Neighbourhood Map

As seen from the map, the color-filled neighborhoods had the most combinations of pickups and drop locations.
i) Red and Yellow - Central
ii) Blue - West Side
iii) Green - Far North Side

b) On analyzing the correlation between the number of trips and the pickup time, we found the peak time for booking a taxi is between 5 pm - 8 pm and the off-peak hours were 3 am - 6 am (2013-Present)



Given our preliminary results from data exploration, we wanted to use data mining approaches to concretize our wishlist.

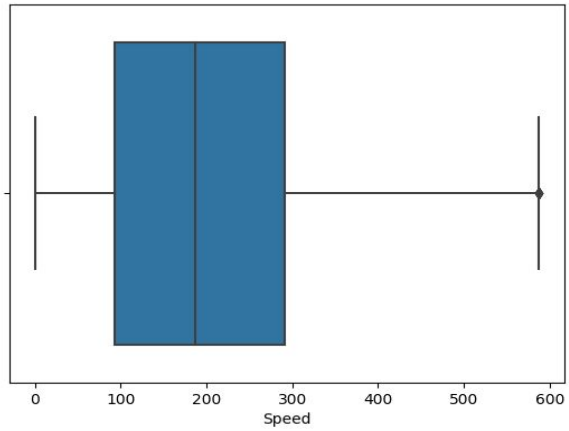
Clustering

a) K-means++

For the experiments below, we only consider data for the year 2016. The dataset provided pickup longitudes and latitudes. Our motivation behind clustering was to form pickup communities where the drivers could reach within a reasonable time. This would allow the drivers and users both to utilize the taxi-service efficiently.

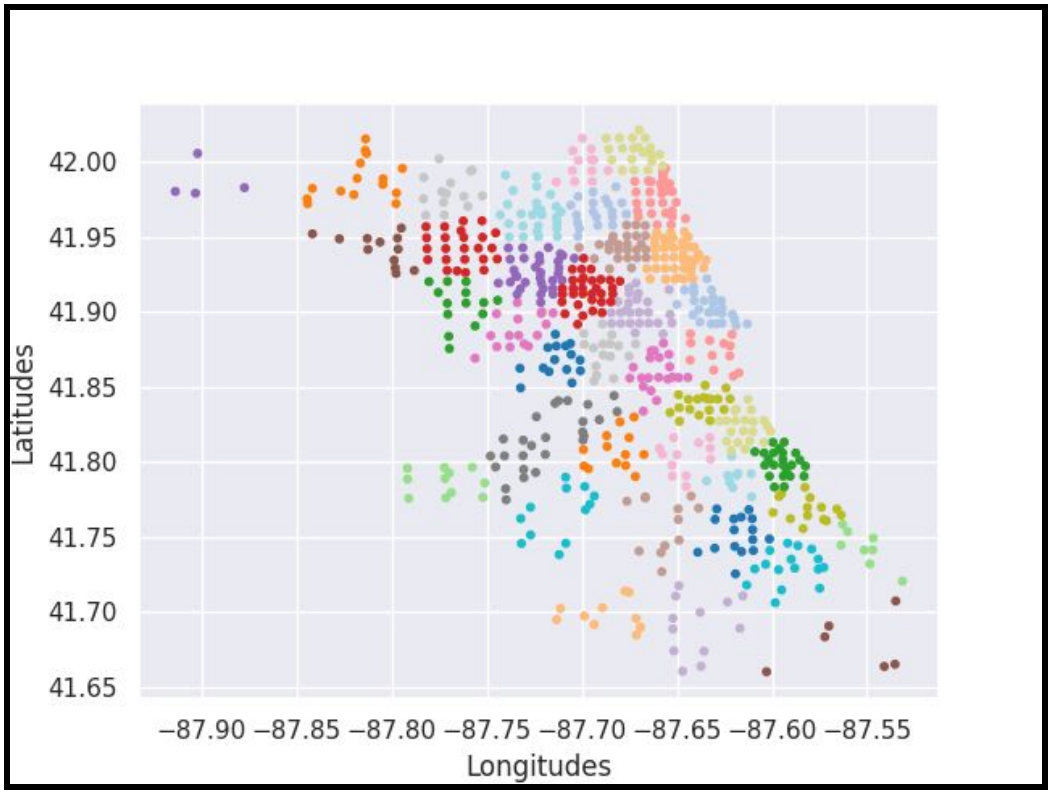
The first step was to find the average speed of the taxi driver in Chicago. From our cleaned data which eradicated about 0.3% of it, we found out the average speed to be 15.07 miles/hr. This means a driver can cover around 2.5 miles in 10 minutes (reasonable wait time).

The next step in the process was to determine the number of clusters for the K-means++ clustering. From our calculation, the desired distance within a cluster was 2.5 miles. At the same time, we didn't want to have a majority of clusters less than 0.5 miles since that would render clustering useless.



Data clean up for average speed

The best ratio of clusters with the above specifications was found with 40 clusters. The number of clusters less than 0.5 miles of distance increased substantially beyond this threshold; the number of clusters with distance more than 2.5 miles increased substantially below this threshold. With 40 clusters we have 16 clusters which have 2.5 miles as the distance between their farthest points. We used the haversine distance to calculate the distance between GPS coordinates.

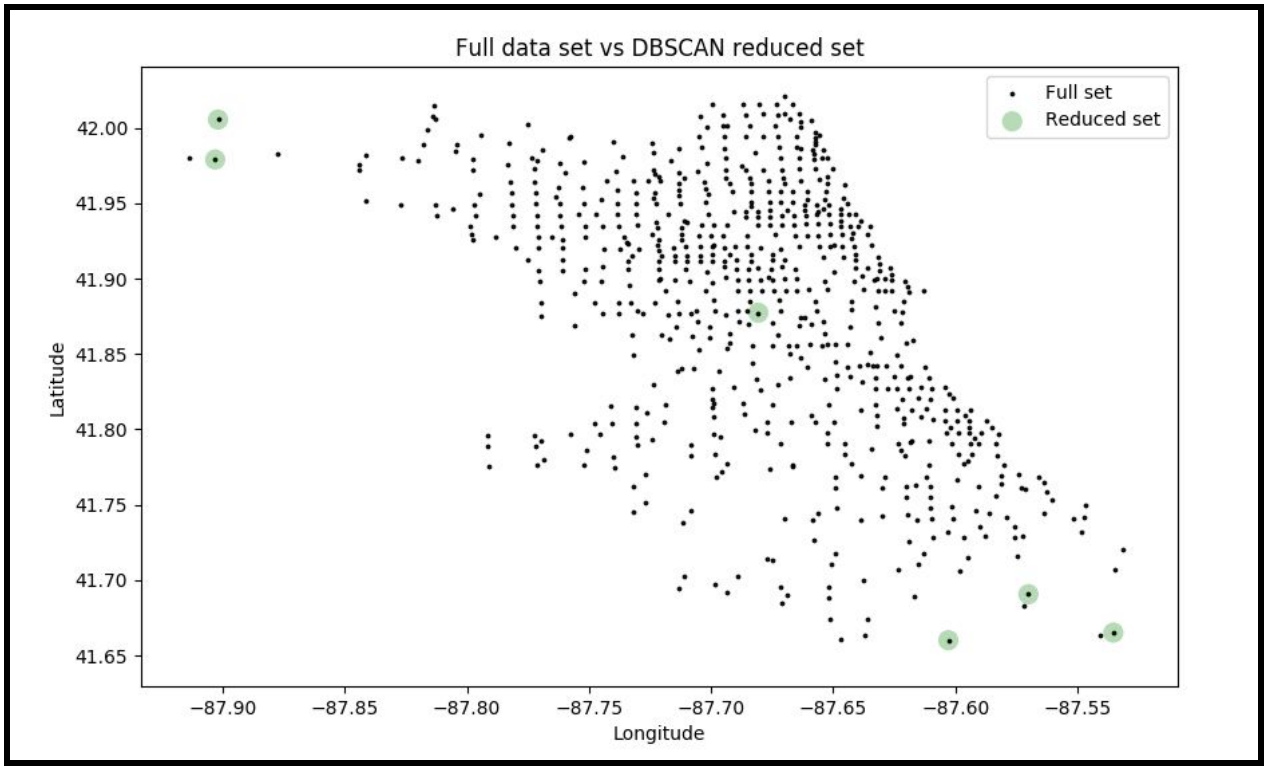


K-mean++ clustering of pickup locations.

b) DBSCAN

While we were exploring other algorithms for clustering, we decided to try DBSCAN as theory suggested it was superior to K-means clustering for geospatial points. We understood that

k-means is not an ideal algorithm for latitude-longitude spatial data because it minimizes variance, not geodetic distance. DBSCAN clusters a spatial data set based on two parameters: a physical distance from each point, and minimum cluster size. We use the haversine metric and ball tree algorithm to calculate great-circle distances between points. Unlike k-means++, DBSCAN doesn't require us to specify the number of clusters in advance – it determines them automatically based on the epsilon(2.5 miles) and min_samples (1) parameters.



We see an interesting observation that DBSCAN has clustered 687 points down to 6 clusters, for 99.1% compression. Given the frequency of trips expected, it would be beneficial if the taxi aggregators had a bay somewhere near the "Near West Side", "South Deering" or "O'Hare" neighborhoods of Chicago to optimize both the driver and user time, which is in contrast with K-means++ result which clustered into 16 possible locations.

Upcoming Goals

We would like to explore other data mining techniques to form a set of popular clusters (termed as a community now onwards) depending on the density of points in them. These communities along with the most popular hour of the day/day of the week would help us predict the demand and bring us closer to our goal. Further, we would use data for 2016 to predict numbers for 2017 and compare it with the real values in the given dataset to solidify our approach.