

Taxi Trajectory Cluster Analysis

Team member 1 (Davneet Kaur, 810944556, *dkaur1@kent.edu*)

Team member 2 (Siddhi Shah, 810933687, *sshah33@kent.edu*)

1. Introduction

With the increase in mobile and global computing, there is an increase in spatiotemporal data available. Previously, it was a problem that researchers did not had enough spatial data to answer useful questions or build convincing visualizations. But today with lots and lots of spatial data being available, the problem is to how to effectively use/visualize this data to get some useful insights from it. This data mainly originates from sources such as location-based services (LBS) applications, GPS (Global Positioning System) tracking systems, or mobile and wireless devices. Specially, with almost all vehicles equipped with Global Positioning Systems (GPS), access to large-scale taxi trajectory data has become very easy. With the increase in access and easy availability of such big data, increases the demand to understand these datasets to get useful information.

Vehicles in general and Taxis in particular, are valuable sensor objects and information related with taxi trajectory datasets can provide a lot of meaningful insight into many facets of a city life. With the use of GPS devices in taxi, it has become easy to record different data for taxis such as the real-time moving location as well as the location information like geographical coordinates, directions, the status whether the taxis are vacant or not and many more [1]. For example, for a city like New York, analysis of a taxi trajectory data can provide information about the life style of people such as what areas are usually busy over the weekends, how much people prefer to go to downtown in weekdays and so on. Such an analysis can also help the taxi vendors to analyze behavior patterns and preferences of their customers. Being able to visualize such analysis on map will give an efficient representation and define the hidden patterns of the data.

For visualizing some hot points over a map, the main techniques that can be used are clustering and heat maps. While heat maps provide just an overview of some aspects on the map, clusters provide a much more detail-oriented view. Cluster is a collection of data objects. The objects within a cluster are like one another and are dissimilar to objects in other clusters. Cluster analysis or clustering is an unsupervised learning in which a set of objects are grouped in such a way that objects in the same cluster or group are more similar (in some sense) to each other than in other groups. Cluster analysis can be used as a stand-

alone tool to get some insights into the data such as in case of data mining or machine learning or it can be used as a preprocessing step for other algorithms such as image analysis, information visualization etc [2].

2. Project Description

- Brief descriptions of your project

Large taxi trajectory data can be exploited for knowledge discovery in mainly transportation and city planning. For example, a simple question like “What are the most picked up locations from noon to 1 PM?” can help to plan for constructing new taxi stands. But even such simple queries can be time consuming with existing tools that typically require users to select and perform interactions with map for retrieving and exploring taxi data.

So, our goal is to provide an application to the users that has a simple UI where the user can select the criteria for the queries from available drop downs and perform simple queries in a timely manner. The initial query that are application supports is based on the time range query. Given the taxi trajectory data for some place, a user can select the time range for the pick-up locations. Once this range query is run against the dataset, the trips within the selected time range are displayed as clusters on the city map. Furthermore, the trips from various different vendors are also visualized as a bar chart that can help analyze the customer preferences at any given time range.

Initially, our application is based on the Mexico City dataset (details of the dataset to follow in the upcoming sections). However, this application can be easily adapted to any other dataset having some date time and location coordinate attributes.

- Challenges and technical contributions

The main challenge in analyzing a massive taxi trajectory data is to visualize the same on a map in an efficient way. Too many scattered points on a map can overwhelm a viewer looking for some simple information. Also, rendering a JavaScript web map (like Leaflet) with millions of data points on a mobile device can swamp the processor and be unresponsive. Our application uses the clustering technique to overcome this issue. It clusters the pickup locations and display one point per cluster on the map. The number of locations clustered together depends on the current zoom of the map.

- The workload distribution for each member in your team

We are a team of two members working on this project. The contribution of each member is as below:

Davneet Kaur: Worked on getting the dataset, data formatting, query processing, creating web UI, clustering and creating visualizations. Also worked on performance reports to get CPU processing times of various processes involved in application creation.

Siddhi Shah: Worked on dataset insertion into mongoDB, making server connection to fetch the dataset, making the dataset available for the application and making server connection to the localhost to host the application.

There is equal contribution from both members in creating the report and the presentation slides.

3. Background

- Related papers

By referring different papers, the numerous problems are disclosed. There are number of things that affects taxi services like customers demand of finding shortest route, waiting time, traffic details, information about other vendors like per mile cost and so on. Existing applications focus to improve the performance of taxi services. They classify different taxi service strategies in different forms like passenger service strategies, passenger delivery strategies and service area preferences [3]. Moreover, nowadays most of the vehicles have GPS systems which makes easy for users to access the taxi trajectory data.

Some papers focus on transportation related problems such as traffic crowding, frequency of accidents, vehicle discharges, vehicle usage rates and driver's behavior [4]. The modes of GPS data collection are defined such as vehicle-based and person-based. In vehicle-based data collection, the vehicles have GPS collectors which gives details of the location within certain time period. On other hand, one needs to have a smart phone having GPS facility to access the GPS data collection or GPS loggers. Taxi GPS data are the example of vehicle-based data collection as they are reliable sources to get information about the behavior of the vehicle. The GPS loggers will turn on when the drivers drive the taxi and as the GPS loggers are connected to the meters of taxis it will be easy to get start and

end point of customers trip. Hence, compared to GPS loggers the taxi GPS data is more useful to get information about the trip instead of knowing the quality of taxi GPS data.

- Software tools
 - DBMS: MongoDB
 - Pymongo – Python library to connect to MongoDB
 - Flask – Python library to connect to localhost and display results in web browser
 - JavaScript
 - Leaflet.js – JavaScript library for Maps
 - D3.js – JavaScript library for Visualization
 - HTML and Bootstrap – For GUI
 - Application is based on Windows Platform
- Required hardware
 - Processor: 2.50GHz or higher (recommended)
 - Memory: 8 GB (recommended)
 - Hard Disk Storage: Depends on input dataset
- Related programming skills
 - Knowledge of MongoDB
 - Basic knowledge of HTML
 - Basic knowledge of JavaScript
 - Basic knowledge of Python

4. Problem Definition

- Formal (mathematical) definitions of problems

Our project aims at mainly two aspects related to taxi trajectory analysis. First is to analyze the pickup patterns in a given time range. Second is to analyze the customer preferences in terms of vendors, that too in a given time range. So, our main goal is:

- To filter the Taxi trajectory data for a certain time interval and perform cluster analysis of the pickup locations.
- Visualize the hot pickup locations on map

- Visualize customer preferences in terms of vendors using bar chart
- Challenges of tackling the problems

The main challenges were to reduce the performance overheads associated with using the JSON data format. Although the initial dataset that we obtained from Kaggle.com [5] is a CSV (Comma Separated Values) file, once the data is uploaded in MongoDB and extracted, it has a JSON (JavaScript Object Notation) format. Larger JSON files tend to take more time while transferring the data to server-side data handling. For tackling such issues, one option is to import only those attributes in the mongoDB that are needed by the current application. Another option is to use the TopoJSON [6] data format for server-side data handling. TopoJSON is an extension of GeoJSON data format and encodes topology. It performs some optimizations, such as eliminating redundancy, on large data files so as to reduce the data sets file size. This format thus works well while visualizing very large datasets without any loss of information.

- A brief summary of general solutions in your project

Our application provides a web-based UI that has two sections. The left side is a map initialized to the (center) latitude and longitude of the Mexico City. The right side is a form that allows the users to select the start and end time for a time range query.

Once the user enters these details and submits, the records in the dataset are filtered for the pickup_datetime against the selected time interval and a clustering is done on the resulting records. We have used Leaflet's marker cluster [9] plugin to perform clustering. This plugin is based on the greedy clustering algorithm. and performs a clustering of the points within the 80 pixels radius of the current map zoom. The resulting clusters are displayed on the map.

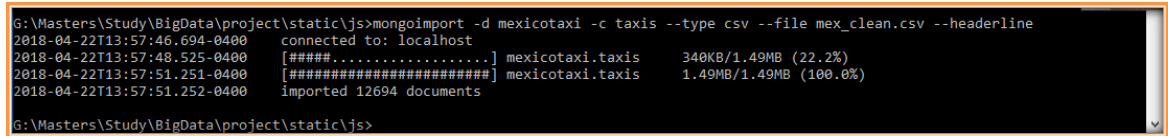
Another solution is to provide the most preferred vendors in the selected time interval. This is done by visualizing the trips per vendor via the bar chart. Hovering over a bar in the bar chart provides specific details such as the vendor_id and the total number of trips by that vendor in the selected time interval. A count of total number of trips that have a pickup_datetime within the selected time interval is also displayed.

5. The Proposed Techniques

- Framework (problem settings)

To setup the framework, first the dataset is imported into the mongodb. Below command can be used for the same:

```
mongoimport -d mexicotaxi -c taxis --type csv --file mex_clean.csv --headerline
```



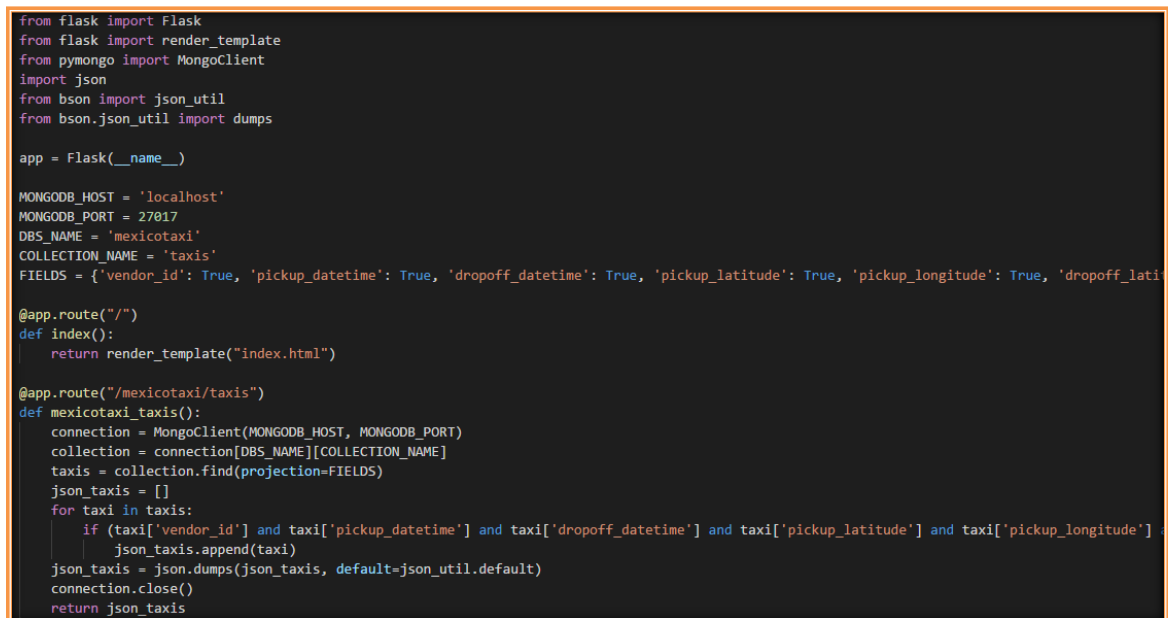
```
G:\Masters\Study\BigData\project\static\js>mongoimport -d mexicotaxi -c taxis --type csv --file mex_clean.csv --headerline
2018-04-22T13:57:46.694-0400    connected to: localhost
2018-04-22T13:57:48.525-0400    [#####.....] mexicotaxi.taxis    340KB/1.49MB (22.2%)
2018-04-22T13:57:51.251-0400    [#####.....] mexicotaxi.taxis    1.49MB/1.49MB (100.0%)
2018-04-22T13:57:51.252-0400    imported 12694 documents
G:\Masters\Study\BigData\project\static\js>
```

Figure1: Importing data in mongoDB

Once the data is imported in mongoDB, the next step is to fetch the data from database and make it available for the range query processing. Pymongo – a python library, is used to fetch the data from mongoDB and make it available for further processing.

Also, a connection with the local server needs to be established to render the final results of the query in the GUI. This is done using the Flask library of python.

Figure 2 shows the app.py where all the connection setups are done.



```
from flask import Flask
from flask import render_template
from pymongo import MongoClient
import json
from bson import json_util
from bson.json_util import dumps

app = Flask(__name__)

MONGODB_HOST = 'localhost'
MONGODB_PORT = 27017
DBS_NAME = 'mexicotaxi'
COLLECTION_NAME = 'taxis'
FIELDS = {'vendor_id': True, 'pickup_datetime': True, 'dropoff_datetime': True, 'pickup_latitude': True, 'pickup_longitude': True, 'dropoff_latitude': True, 'dropoff_longitude': True}

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/mexicotaxi/taxis")
def mexicotaxi_taxis():
    connection = MongoClient(MONGODB_HOST, MONGODB_PORT)
    collection = connection[DBS_NAME][COLLECTION_NAME]
    taxis = collection.find(projection=FIELDS)
    json_taxis = []
    for taxi in taxis:
        if (taxi['vendor_id'] and taxi['pickup_datetime'] and taxi['dropoff_datetime'] and taxi['pickup_latitude'] and taxi['pickup_longitude'] and taxi['dropoff_latitude'] and taxi['dropoff_longitude']):
            json_taxis.append(taxi)
    json_taxis = json.dumps(json_taxis, default=json_util.default)
    connection.close()
    return json_taxis
```

Figure 2: app.py file

In this file, the data is fetched from the mongoDB database after making a connection using Pymongo. Once the connection is established, a condition is checked to drop records having any missing values and fetch the rest of the records. And then Flask is used to render the results to index.html where all the JavaScript code is implemented.

- Details of major techniques

Leaflet.js [8] is used to render a map initialized at latitude and longitude of the Mexico City. This is an open source JavaScript library used to generate interactive maps. We have also used Leaflet's marker cluster [9] to do clustering. This plugin provides a number of features such as:

- Display the number of nodes in a cluster, as a label on each cluster. As the map zooms in/out, the clusters span or contracts and the labels also change accordingly.
- Hovering over a cluster displays a polygon showing an outline of the area covered by that cluster.
- Clicking on a cluster zooms to the next level of map and all the clusters adjust to cover the current display of the map. We have not utilized this feature in our project.
- When a cluster has only one node, or when the map is at the highest level of zoom, all the nodes are displayed as single markers. Clicking on a marker provides some information about that marker. This can be changed as per the requirements. For our application, it displays the vendor

D3.js [7] is a JavaScript library used to create visualizations. The D3 stands for Data driven Documents. It allows to bind the data to a Document Object Model (DOM) and then apply data-driven transformations to it. The elements can be manipulated in form of SVG, HTML and CSS in the context of dataset.

It has built in APIs to create visualizations such as Bar charts, Pie charts, Heat maps and many other kinds of visualizations. It also provides many features to add different interactions to the visualizations to make them more user friendly. For our application, we have created a bar chart visualization using the D3.js API and added mouse hovering interaction to it.

- Query processing algorithms (pseudo code) and query optimizations

Below are the steps for the query processing algorithm:

- Initialize the Map to center latitude and longitude of Mexico City

- Populate Start and End time to get a time range
- Enter Start and end time
- Filter trip records based on pickup_datetime between entered start and end time
- Perform cluster analysis using Leaflet-Marker clusters for filtered trips
- Display Hot pickup locations on the map
- Count number of trips per vendor within the entered time range
- Display Vender preferences via bar chart

6. Visual Applications

- GUI design

We have used Leaflet.js [9] to create an interactive map of the Mexico City. We also used D3.js to visualize the bar chart. The frontend design is done using HTML and Bootstrap. All the coding is done in the Visual Studio Code IDE. Figure 3 shows the folder structure of the source code:

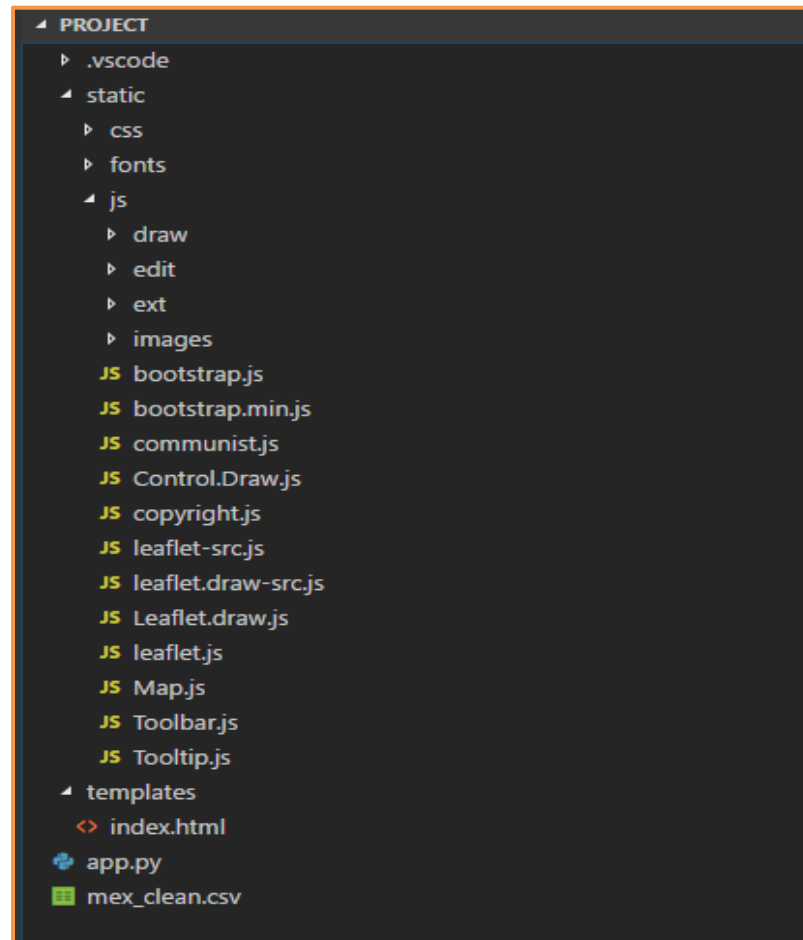


Figure 3: Source code folder structure

- Design modules
 - Index.html: This is the main file that generates the initial UI on a web browser. It creates a form that allows the user to enter the start and end time interval for the range query.
 - Map.js: This is a JavaScript file that does most of the processing work. It takes the dataset passed on by the app.py and performs the query processing over it. The main function filters out all the records within the selected time interval and passes the selected records to other functions to generate the visualizations.

```
function DrawOnMap() {
  d3.json("/mexicotaxi/taxis", function (error, data) {
    if (!error) {
      data.forEach(function (d) {
        d["pickup_datetime"] = parseTime(d["pickup_datetime"]);
      });

      //Get data for specified time
      var startTime = parseInt(document.getElementById("startTime").value);
      var endTime = parseInt(document.getElementById("endTime").value);

      //Filter Trips
      var selectedTrips = [];
      for (var i = 0; i < data.length; ++i) {
        var hours = data[i].pickup_datetime.getHours();
        var minutes = data[i].pickup_datetime.getMinutes();
        if (startTime <= hours && endTime >= hours) {
          if (endTime == hours && minutes > 0) {
            //do nothing
          }
          else {
            selectedTrips.push(data[i]);
          }
        }
      }

      DrawClusters(selectedTrips);
      DrawTextbox(selectedTrips);
      DrawBarchart(selectedTrips);
    }
    else {
      console.log('Could not load data...');
    }
  });
}
```

Figure 4: Map.js

7. Experimental Evaluation

- Experimental settings
 - Descriptions of real/synthetic data sets

We have used the real dataset for our application. The dataset of the Mexico City is obtained from Kaggle.com [5] and consists of 12 attributes as shown in figure below:

• Id	• dropoff_longitude
• vendor_id	• dropoff_latitude
• pickup_datetime	• store_and_fwd_flag
• dropoff_datetime	• trip_duration
• pickup_longitude	• dist_meters
• pickup_latitude	• wait_sec

Figure 5: Attributes of the Mexico City dataset

Out of these 12 we have used 4 attributes for our analysis. The attributes used are:

- **vendor_id**: Id of the different taxi vendors. It is a nominal attribute and has seven values: Mexico DF Radio Taxi, Mexico DF Taxi de Sitio, Mexico DF Taxi Libre, Mexico DF UberBlack, Mexico DF UberSUV, Mexico DF UberX, Mexico DF UberXL.
- **pickup_datetime**: A date-time attribute that gives the date and timestamp of the trip pickups.
- **pickup_longitude** & **pickup_latitude**: location coordinates of the pickup locations of each trip.

The other attributes can be leveraged for some analysis/data mining in future.

- The performance reports

We have measured the CPU time needed to load and process different sizes of our dataset. The processing time is measured for different aspects such as loading the dataset, counting trips within the selected time interval, doing clustering, getting vendor preferences (i.e. visualizing bar chart) and the overall

processing time. The total processing time however does not include the data loading time.

The dataset size is altered starting from 10 records to total records in dataset i.e. 12695. The times are measured for 10, 100, 1000, 10000 and 12695 records. The time range is selected from 7AM to 4PM. It may be noted however, that the processing time can vary for different time intervals. Figure 6 shows an analysis of the same.

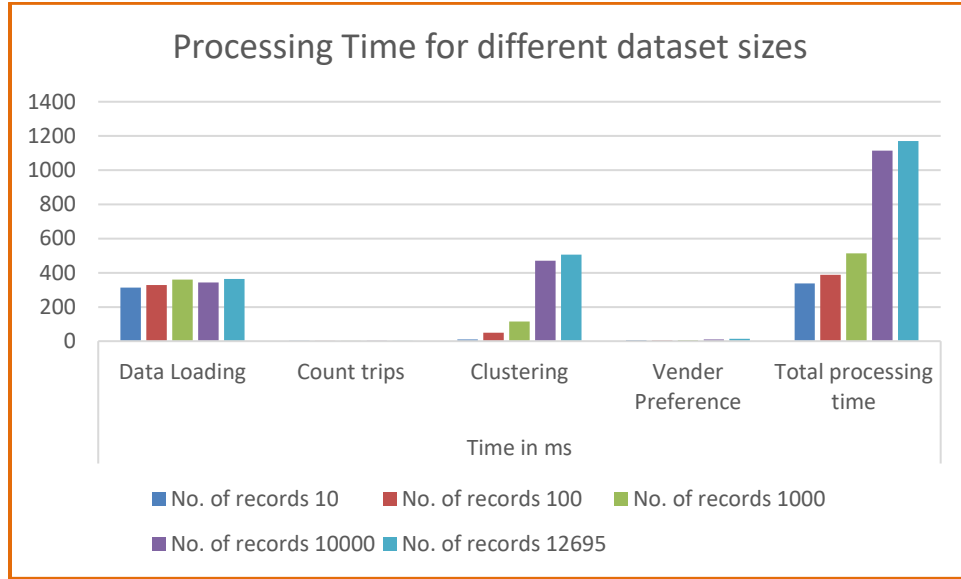


Figure 6: Processing time for different dataset sizes

The y-axis shows the time in milliseconds and the x-axis shows different processes for which time is measured along with the number of records.

We have also done measurements of the time taken for different aspects such as loading the dataset, counting trips within the selected time interval, doing clustering, getting vender preferences (i.e. visualizing bar chart) and the overall processing time in different time slots. The complete dataset of 12695 records is used in this analysis. The time slots are 2 hours each starting from morning 8am to night 8pm.

Figure 7 shows an analysis of the same. The y-axis shows the different process and x-axis shows time in milliseconds. The total number of trips in the selected time range are also shown alongside of each time slot in the legends.

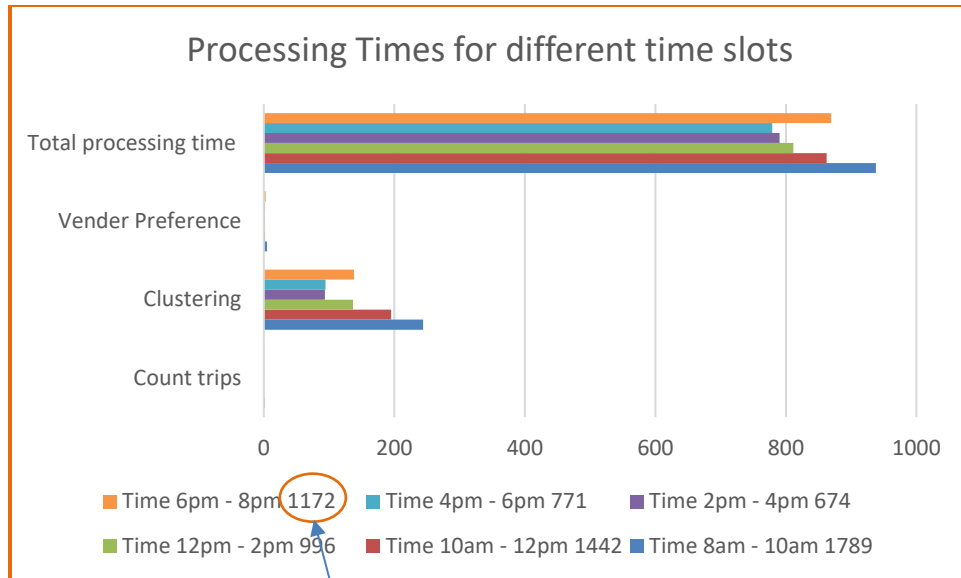


Figure 7: Processing times for different time slots

Trip count

- Screen captures

Below are the screen captures of our web-based UI

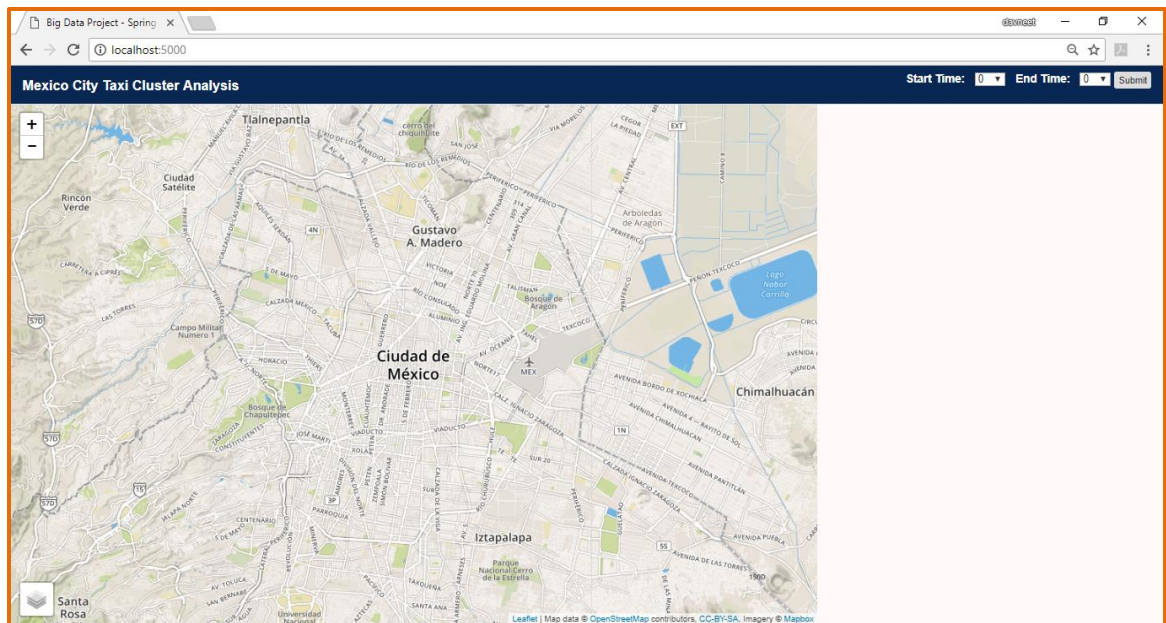


Figure 8: Initial UI

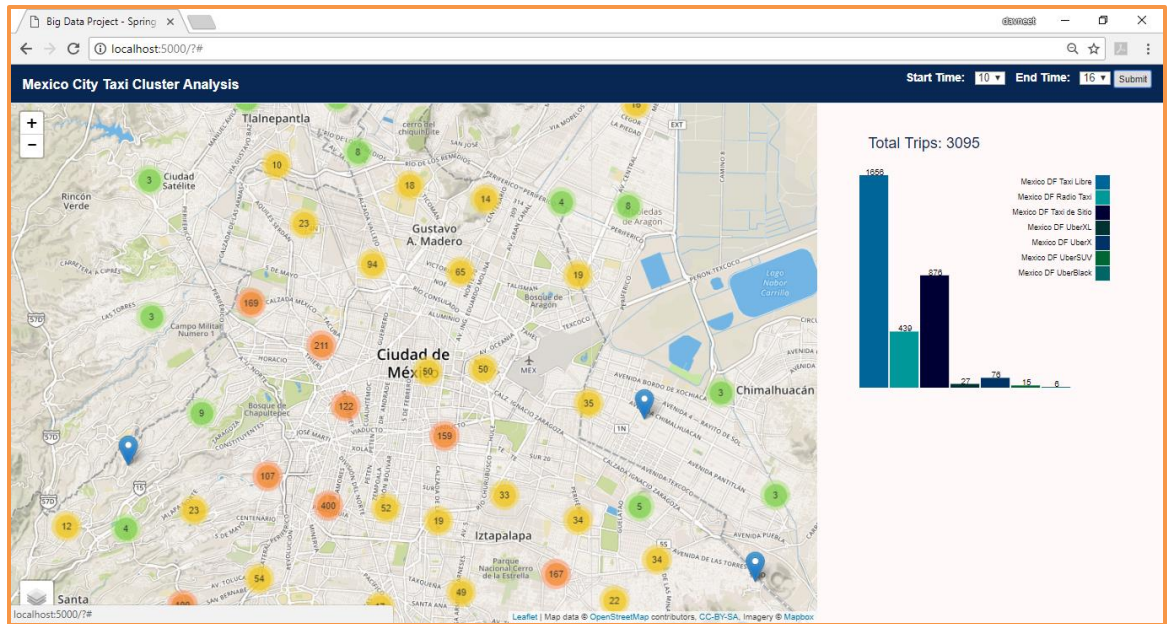


Figure 9: Clustering and Vendor preference for a particular time interval

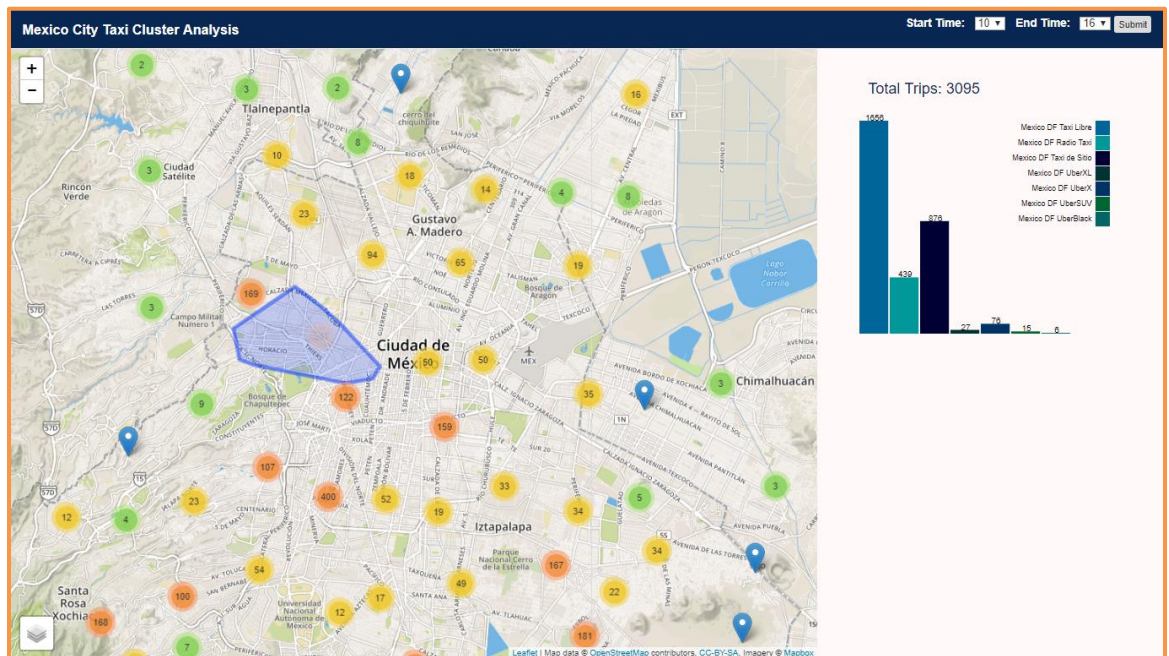


Figure 10: Hovering over the cluster highlights the area covered by that cluster

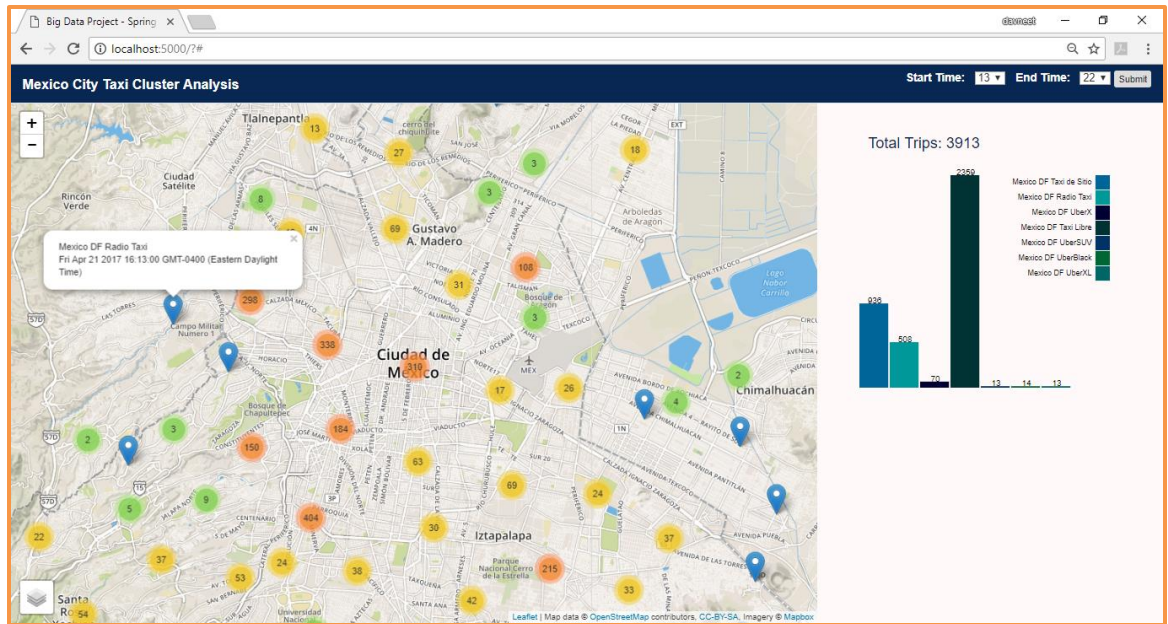


Figure 11: A standalone marker shows the Vendor Id and trip pickup time

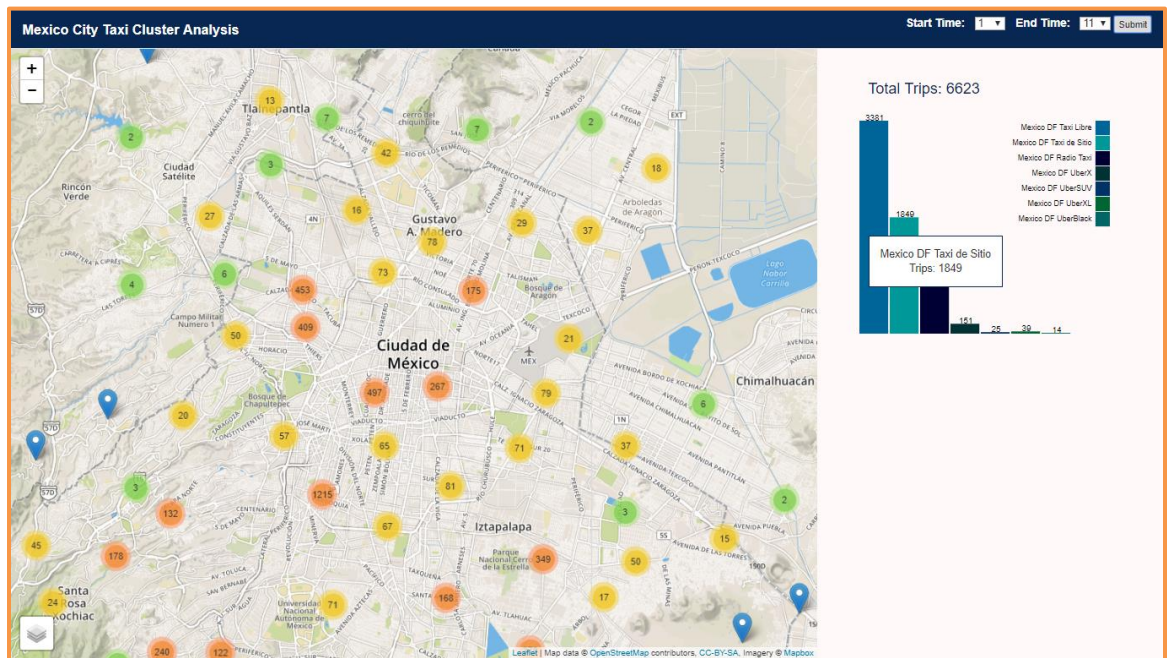


Figure 12: Hovering over a bar shows the Vendor id and trip count

8. Future Work

- Possible project extensions

A possible future work to our application is that the analysis of drop off locations along with pickups can help people share taxi with a common route. Also, by integrating with some other dataset, the customers loading status and speed information can be used different visualizations. Lastly, an analysis and comparisons of prices and taxi wait times among different vendors can be done that can help the customers to get best options for their rides

9. References

- [1] Hui Xiong, Lei Chen, and Zhipeng Guo. A Web-based Platform for Visualizing Spatiotemporal Dynamics of Big Taxi Data. In School of Remote Sensing Information and Engineering, Wuhan University, Vol XLII-2/W7, 2017.
- [2] P. X. Zhao, K. Qin, Q. Zhou, C. K. Liu, Y. X. Chen. Detecting Hotspots from Taxi Trajectory Data using Special Cluster Analysis. In ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-4/W2, 2015
- [3] Zhong Zheng, Soora Rasouli, and Harry Timmermans. Evaluating the accuracy of GPS-based taxi trajectory records. In Eindhoven University of Technology, Vol. 22, pages 186-198, 2014.
- [4] Daqing Zhang, Lin Sun, Bin Li, Chao Chen, Gang Pan, Shijian Li, and Zhaohui Wu. Understanding Taxi Service Strategies from Taxi GPS Traces. In IEEE Transactions on Intelligent Transportation Systems, Vol. 16, pages 123-135, 2015.
- [5] Mario Navas. Taxi Routes of Mexico City, Quito and more. 2017. [Online] Available: <https://www.kaggle.com/mnavas/taxi-routes-for-mexico-city-and-quito/data>
- [6] TopoJSON. [Online] Available: <https://github.com/topojson/topojson>
- [7] D3.js [Online] Available: <https://d3js.org/>
- [8] Leaflet.js [Online] Available: <https://leafletjs.com/>

[9] Leaflet-marker cluster. [Online] Available:
<https://github.com/Leaflet/Leaflet.markercluster>