TAXI RIDE SHARING

A user-rating based approach with user preferences of save time or save money

By

Christopher Alphones	657081159	calpho2@uic.edu
Gopi Krishnan N	676680127	gnaras2@uic.edu
Sachin Hegde	671317517	shegde8@uic.edu
Tinesh Madan Mohan	663468363	tmadan2@uic.edu

Guidance of

Prof. Ouri Wolfson, PhD
TA: Aishwarya Vijayan

COMPUTER SCIENCE
COLLEGE OF ENGINEERING
UIC

ABSTRACT

Millions of rides are taken every year in United states and most of the solutions that were extensively used till recently including New York yellow taxi mainly served single ride. This is inefficient considering how many passengers book single rides with source and destination that are close or on the way. The primary goal of this implementation is to efficiently combine rides arising from single prominent source and destinations which are close-by or on the way. We further experiment with user score model which would encourage the passengers to invest in this system by providing them with incentives over time including priority in pooling, route selection and savings with respect to time and money.

Contents

1. Introduction

Ride sharing services like Uber and Lyft have shown rapid growth in the recent past. They are redefining the urban mobility by providing a convenient way of getting around from one place to another. Also, they help in reducing the number of on-road vehicles and thereby present positive impact by reducing pollution and traffic congestion. These services are user-centric and try to increase their market by improvising and coming up with better solution.

In this project we present a novel idea of incorporating "User Rating" which tends to increase, or decrease based on the passenger's ride profile history. The algorithm is validated with rides extracted from the New York City taxicab public dataset. Our experimental study considers ride-sharing with rider capacity of up to 3 simultaneous passengers per vehicle. The user rating is a deciding factor during every pool window, and the rides are merged based on the same criteria. The algorithm performs well for the test data that spans over the first week of February. We do see an average reduction in daily trip by 49% and the distance travelled reduces by 53% satisfying all the constraints.

The various aspects of the project are discussed in detail in the forthcoming sections. Especially, we get a better insight of the algorithm's performance from the Reports and Inferences section.

1.1 Data

The trip data that is considered is from the New York Yellow Taxi Data (TLC Trip Recod Data, n.d.) for the month of February,2016. This dataset includes trip records from all trips completed in yellow taxis from in NYC for the period. Records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. There was a total of 11.4 M records considering all source and destinations.

1.2 Pre-Processing

This is an essential step before working with the data, and involved multiple steps:

- *Filtering*
  Consider trips originating from JFK terminal, then the trip count reduced to approx. 33k.
- *Cleaning*
  Involved removing irrelevant data, where the passenger count was 0 or null.
- *Update*
  Recomputed the distance using Open Source Routing Machine (OSRM) for the trips considered. Also, set passenger count greater than three to three.

This preprocessed data is fed as input to the algorithm under consideration.

1.3 Assumptions and Constraints

For the algorithm under consideration there are a set of assumptions and constraints which are as mentioned below:

- No Walking
- Pooling windows 2 and 5 minutes were experimented.
- Single Airport terminal pickup point is considered.
- Merge only static trips.
- Assume taxi are always available
- User Matching will be based on drop-off point of the highest rated user.
- Maximum up to three passengers are allowed in a taxi.
- Dynamic Queue Based Pool Window.
- Disregard traffic condition.
- User will be given an option between saving time and saving money
- Ambiguity will be resolved with User Rating and the priority will be given to the high rating

2 Algorithm

The algorithm can be conceptualized as multiple parts handling specific functionality. User scores and preferences are baked into every aspect while trying to manage efficiency

2.1 Pooling

Input:
All users in the system
Algorithm:
Sort all the users in the system by time.
Consider the 1st user in the system:
  For all the users who are within the pooling window of the 1st user
  Add them to the pool
For all the users in the current pool:
  Divide the users into clusters with DBSCAN.
  With all the users selected in the pool, divide the city into grids.
  Consider the user with the highest rating and assign that user as pool leader.
  Consider the route to the pool leader as the initial route.
  With pickup and pool leader coordinates find the grids that are between two points with Bresenham's algorithm
  while the pool limit not reached:
    Add all users in the pool leader cluster to the final pool.
    Add all the users in the selected grid to the final pool.
Push all the remaining users if delay allows it. Provide individual ride otherwise.
Output:
All the users that are assigned to specific pool

2.2 Routing

Input:
All the users that are assigned to specific pool
Algorithm:
For all the users in the drop off pool:
  Get the routing distance to all the users in the pool
  Select the user with shortest route considering user rating and add him to the routing list
  From the current user repeat the algorithm till all users are added to the routing list.
Output:
Ordered list of users according to drop-off

2.3 User Preference

Each user in the system is given option to select save money or save time. This would affect users routing and savings that will be added to his trip.

If the user has selected save money, they are required to enter delay constrain. This is the total time the user is ready to wait if they cannot be added to the pool at that time.

2.4 Savings

Every user who are assigned a pool taxi are entitled to receive 20% savings on their trip

If user preference is “Save Money” Applicable savings are added on top of standard saving as below

If User rating > 4.7 – 22% additional saving

If User rating > 4.3 – 12% additional saving

If User rating > 4.0 – 4% additional saving

3 Technical Specifications

3.1 Database

The data from (TLC Trip Recod Data, n.d.), after preprocessing is stored in a MySQL Relation Database.

Tools used: *MySQL Workbench 6.3.10.*

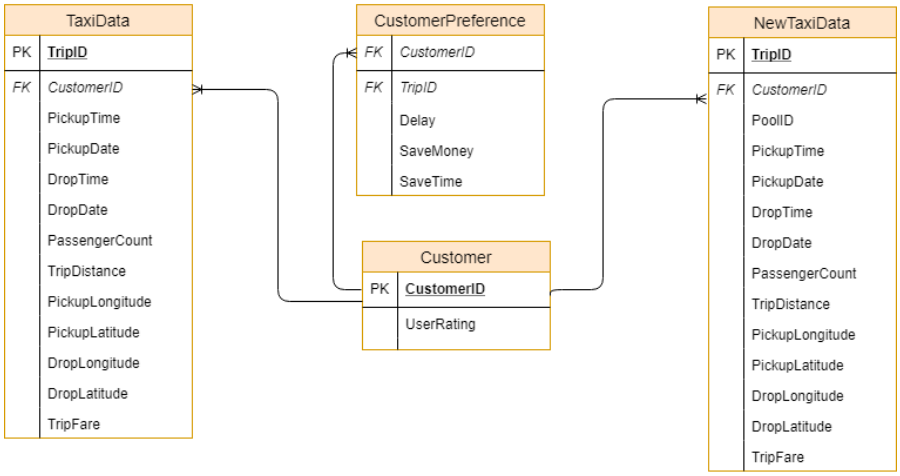

Figure 1 ER Diagram of the Database

3.2 Language

The entire project was implemented in *Python 3.6* on Anaconda Platform using Jupyter Notebook.

3.3 Libraries

Multiple libraries were used starting from initial preprocessing till obtaining the results. Listed below:

- To access external data stored in database, file and external services.
  - `pymysql`: Database Connector, enables python programs to talk to MySQL Server.
  - `urllib`: This module provides a high-level interface for fetching data across the World Wide Web, here to make calls to OSRM service.
  - `csv`: The csv module implements classes to read and write tabular data in CSV format.
- Routing and Pooling
  - `bresenham`: An implementation of Bresenham’s line drawing algorithm.
  - `scikit-learn`: Used sklearn.cluster.DBSCAN to implement the clustering.

- Data Processing
    - `numpy`: A fundamental package for computing supports N-dimensional array objects.
    - `json`: To support JSON (JavaScript Object Notation) is a lightweight data-interchange format
    - `datetime`: Has classes for manipulating dates and times in both simple and complex ways.
    - `collections`: Implements specialized container datatypes.

3.4 Services

All shortest paths were computed using *Open Source Routing Machine* (OSRM). This internally combines sophisticated routing algorithms with the open and free road network data of the OpenStreetMap (OSM) project to get the shortest path.

3.5 Reports

To visualize the results, we used *Tableau* a data visualization software and *Microsoft Excel* to plot the static graphs.

4 Analysis and Inference

4.1 User Rating

The distribution of user rating is uniform cross all the customers and they are evenly distributed over the subset of trips considered. This can be inferred from the pie chart that groups customers from 0.5- 1. 0, 1.0-1.5 so on until 4.5-5.0.

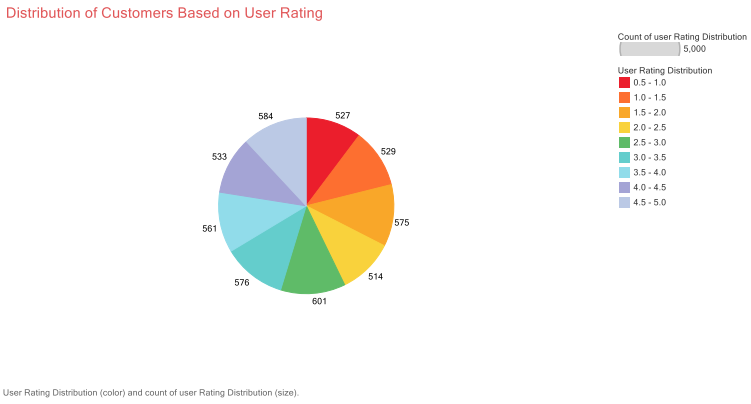

Figure 2 Pie Chart of User Rating distribution

4.2 Input Data

The input data processed is geo-spatial in nature, and the motive is to combine the rides. The map is layered to show all destinations of rides that took place in the first three days of Febraruary,2016. The are color coded based on the number of passengers in each ride.

Inference: Most of the rides are One Passenger rides.

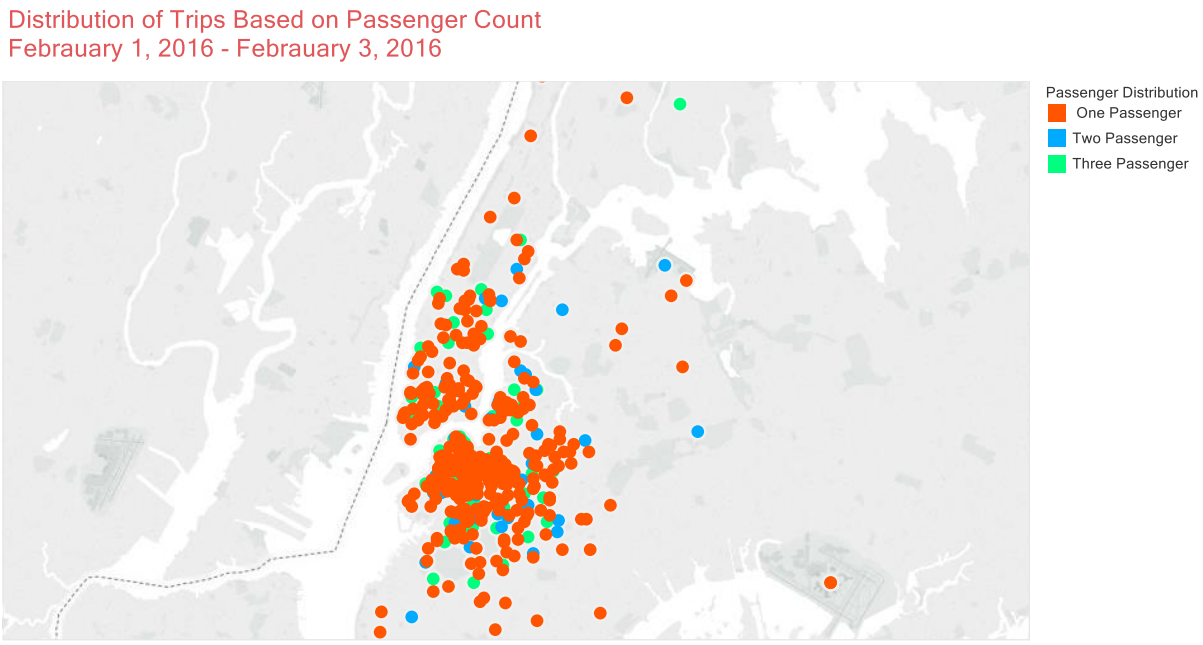

Figure 3 The Map of New York with trips grouped by passenger count

4.3 Number of Passengers in a Taxi

A bubble graph representation of the count of passengers in each of the rides before merging and after merging. As seen the number of Three Passenger trips has increased which shows clearly indicate that many single passenger trips have merged.

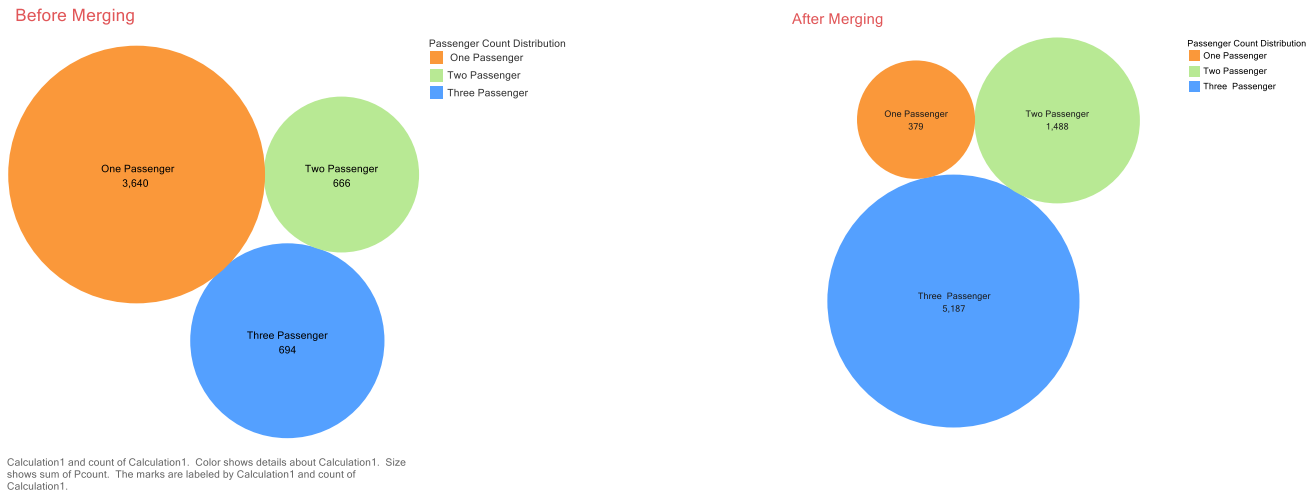

Figure 4 Bubble chart representation of Passengers in Taxi

4.4 Trip Count.

The motive of the algorithm is to combine trips satisfying all the constraints, this graph shows the fall in number of trips daily in the first week of Feb,2016. For example, on day 5 the number of trips before merging were 1062 and after merging is 593.

On an average there is ***48.98 %*** decrease in the number of trips daily.

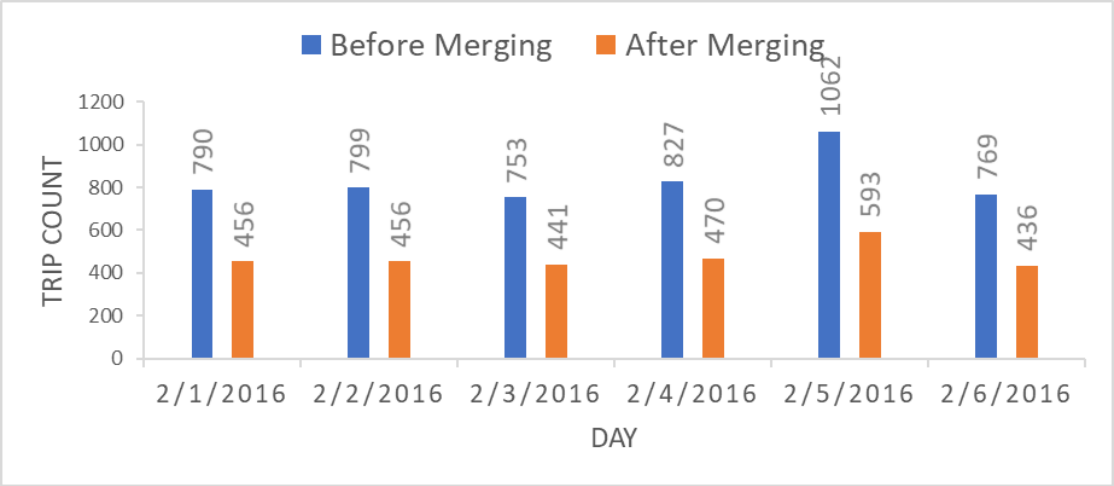

Figure 5 Bar chart to visualize trip counts

4.5 Trip Distances

Similarly the algorithm tries to reduce the number of miles covered by combing the trips , this graph shows the fall in miles covered daily in the first week of Feb,2016. For example, on day 5 the number of trips before merging was 17959 miles and after merging is 8240 miles.

On an average there is ***53.37 %*** decrease in distance covered daily.

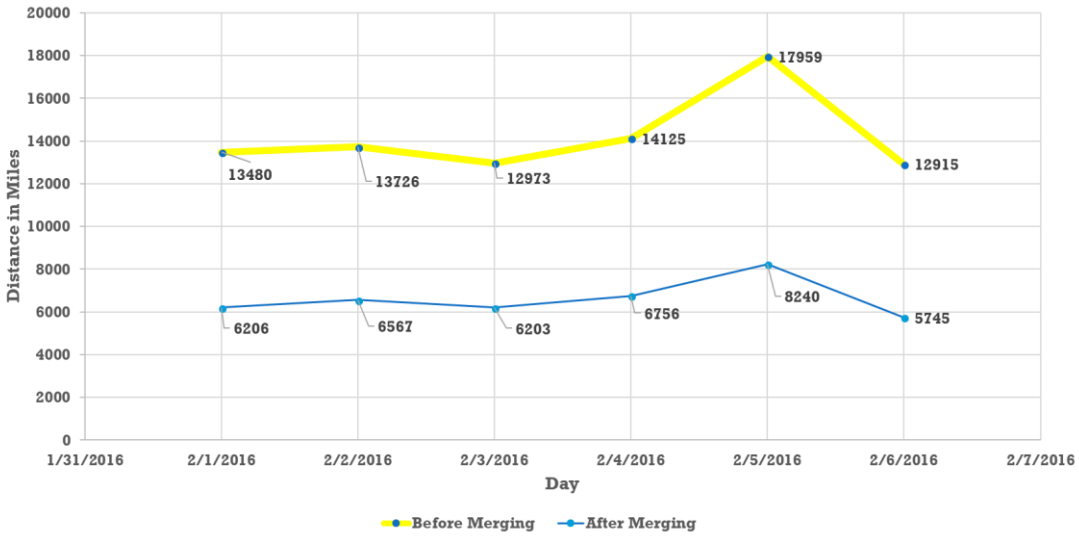

Figure 6 Line Chart to visualize miles saved

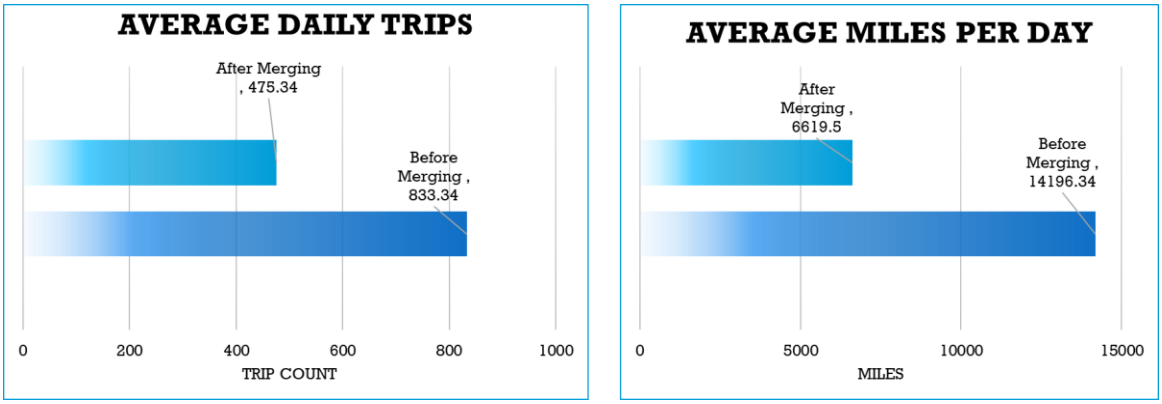

Figure 7 Average trip count and miles covered

4.6 Savings

Saving is measured in terms of two criteria: Money and Time. The algorithm performs well on both the metrics.

4.6.1 Money

Saving for every customer is also dependent on their rating, the more the rating the more the saving. The maximum saving is for customers above 4.7 rating approximately 5.68$ /customer.

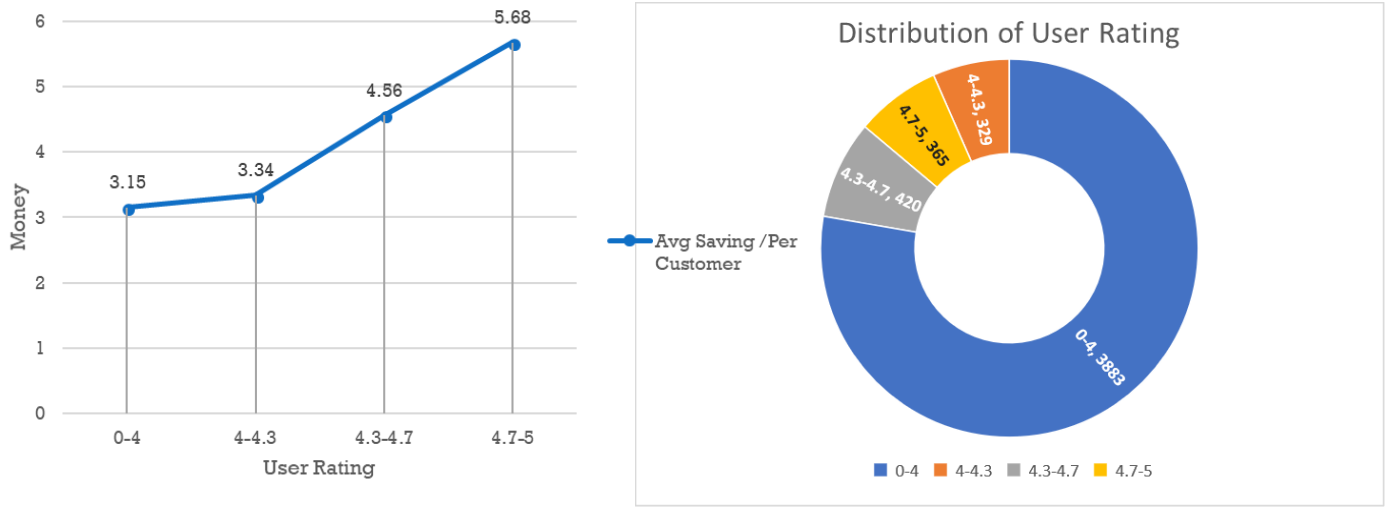

Figure 8 Incentives in terms of money saved

4.6.2 Time

Savings in terms of time for any passenger who is said save time and save money is as below:

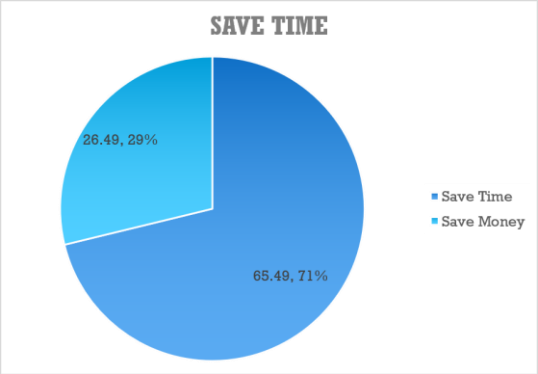

Figure 9 Pie Chart Distribution of Save Time

4.7 Performance of Algorithm

The algorithm is efficient in pooling the users in the system while making sure users does not experience any delay by the system. The implementation incorporates DBSCAN and Bresenham's Algorithm which are computationally efficient compared to other alternatives. Ref: (About Bresenham Algorithm)

In our testing with 6-day data took 900 seconds for processing 5000 records and divide them into pools. This processing time excludes OSRM request time.

5 Experiments

Following represent the various experiments that were conducted for varying pool size 2 and 5.

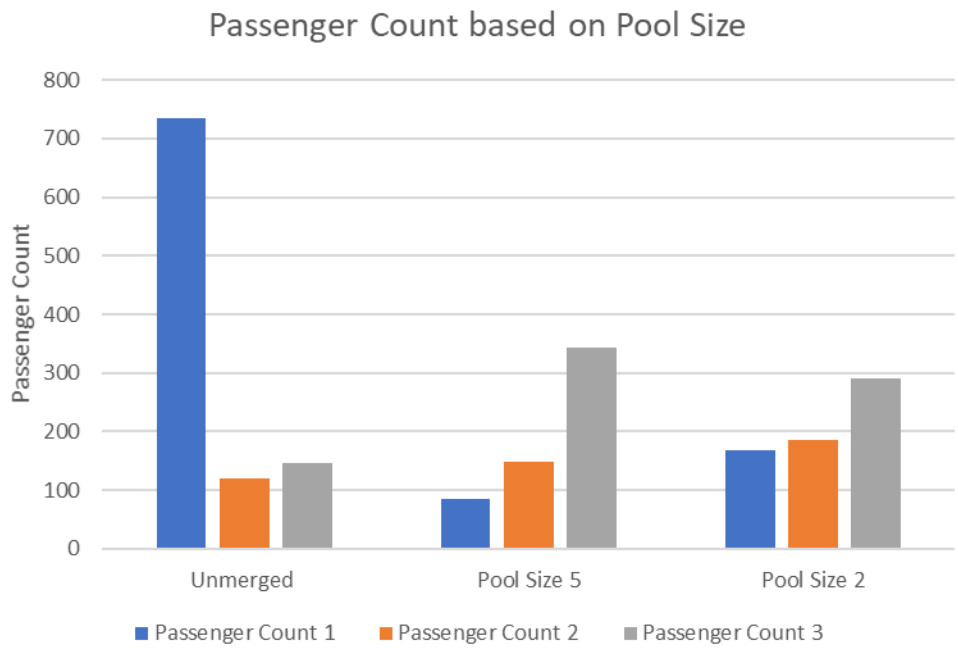

Figure 10 Bar char t to visualize effect of pool size on Passenger Count

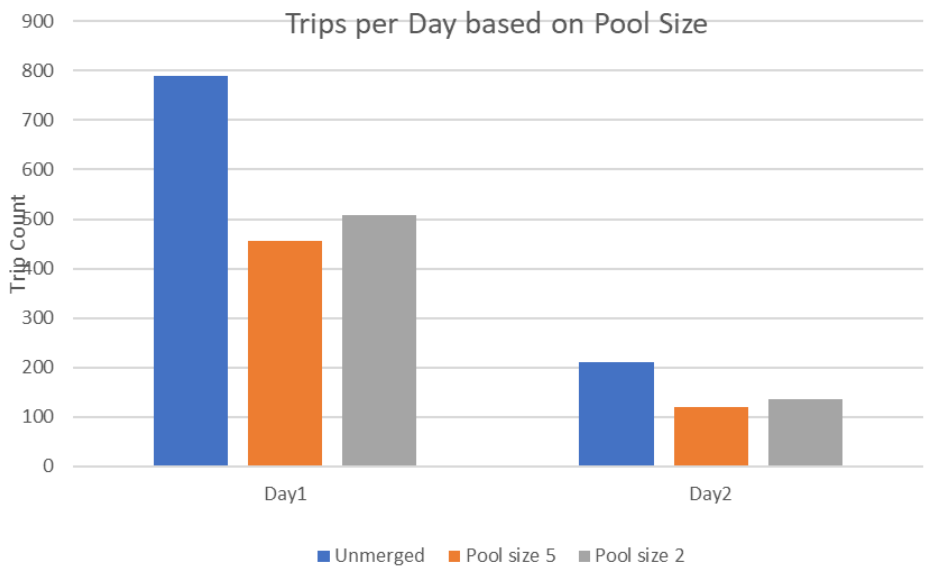

Figure 11 Bar chart to visualize effect of pool size on Trip count

6 Conclusion

The project implemented with the goal of combining maximum rides efficiently while considering the user rating has been successful. The implementation reduced the total number of rides and distance per day to almost half of the original values, it also efficiently managed pooling, routing and savings part while saving money and time of the user. The implementation reduced individual trips significantly.

Simultaneously the implementation also takes user rating into consideration which helps in retaining the existing users by rewarding them. This makes it a great option for commercial implementation as a next step.

7 References

About Bresenham Algorithm. (n.d.). Retrieved from http://www.phatcode.net/res/224/files/html/ch35/35-01.html

NYC Taxi Data. (2016). Retrieved from NYC Taxi and Limousine Commission's Industry Reports web page: http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

OSRM. (n.d.). Retrieved from http://project-osrm.org/