Via Data Challenge

Johannes Breitschwerdt
johannes.breitschwerdt18@imperial.ac.uk

Part 1: Analysis of NYC Taxi Data, Metric Creation, and Efficiency Analysis

Question 1: Propose a metric and/or algorithm to assess the potential efficiency of aggregating rides from many vehicles into one, given the available data. Make realistic assumptions and any necessary simplifications and state them.

Algorithm:
In this question, I create an algorithm which pools together passengers based on their location in time and space. In doing so, I make the following assumptions:

a) Passengers are willing to wait up to 5 minutes
b) Passengers are willing to initially move in a direction opposite to their target destination
c) Natural barriers (e.g. rivers, parks) do not play a role when pooling passengers
d) There can be at most 4 passengers in a taxi cab (max occupancy)
e) I do take already full vehicles (occupancy 4 or higher) into account when implementing the algorithm or in the efficiency metric, since these vehicles are not available for pooling. They are filtered out of the data before the algorithm is run.

These assumptions are also some of the algorithm's limitations. I implement a *greedy, myopic matching algorithm* which works on a single-event basis. The steps are as follows:

1) A list keeps track of which passengers have been pooled.
2) A passenger calls a taxi at time t. If the passenger has not been pooled, a matching process begins. At this instance, the data is filtered for all those other taxi hire events which occur in the time frame $[t, t+5 mins]$.
3) The time-matching taxi-hires are then filtered further to include only those which have an origin-location within 0.6 km of the calling passenger's origin and a destination-location within 1 km of the passenger's destination. This is implemented using the geodesic distance via an external API call.
4) The algorithm then greedily evaluates the qualifying rides in the order in which they appear. If the sum of the current taxi-hire's passenger count and the next matching hire's passenger count is less than or equal to 4, the rides are pooled and the tracker saves the identifier of the pooled ride. This continues for all qualifying rides until the occupancy reaches 4 or the list of matching rides is exhausted, whichever occurs first. If no matching rides are found, the taxi-hire is saved as an unmatched 'solo' ride.
5) The given taxi-hire is saved to the list of previously pooled hires.

A pseudo-code is as follows:

```
initialize ride_ID

for taxi_hire_event in taxi_data:
    occupancy = occupancy_stated

    filter for all other taxi_hire_events which occur within next 5 minutes
    filter for all other taxi_hire_events which occur within threshold distance of
current taxi_hire_event's origin and destination locations

    if matching rides exist:
        for other_taxi_hire in matching_hires:
            if occupancy + next_event_number_passengers <=4:
            #pool rides
            occupancy += next_event_number_passengers
            save pooled rides in dataframe with ride_ID with relevant trip info
    else:
        save taxi_hire_event as solo ride and append to dataframe with   ride_ID

    increment ride_ID
```

It is important to state that the time, origin, and destination thresholds are arbitrary, and are values which I have chosen based on my experience with ride sharing.

Efficiency Metric:
The efficiency metric is defined as the percentage of occupied seats which occur in a given taxi ride. For example, a taxi ride with 4 passengers will have a $\frac{4\ passengers}{4\ seats} * 100 = 100\%$ efficiency, while a taxi ride with 1 passenger will only have a 25% efficiency. This will allow for analysis later on regarding the effectiveness of pooling using the proposed algorithm.

Question 2. Implement your proposed method and evaluate Manhattan's overall efficiency using yellow taxi data from the first full week (Monday-Sunday) in June 2016. Discuss how your method would scale with more data; in other words, discuss the complexity of your implementation.

Given the size of the dataset (2.6 million+ rows) and the complexity of the code (more on this below), I evaluate the algorithm on a smaller subset of the data: 6 am - 9am, Monday, 6th June, 2016. As demand patterns are seasonal throughout the week and the morning presents a window of rapidly increasing demand with time, I believe this to be a suitable time window to test the pooling algorithm. Please see the explanation below for an elaboration on the code's complexity, its implications, and the likely steps to real implementation.

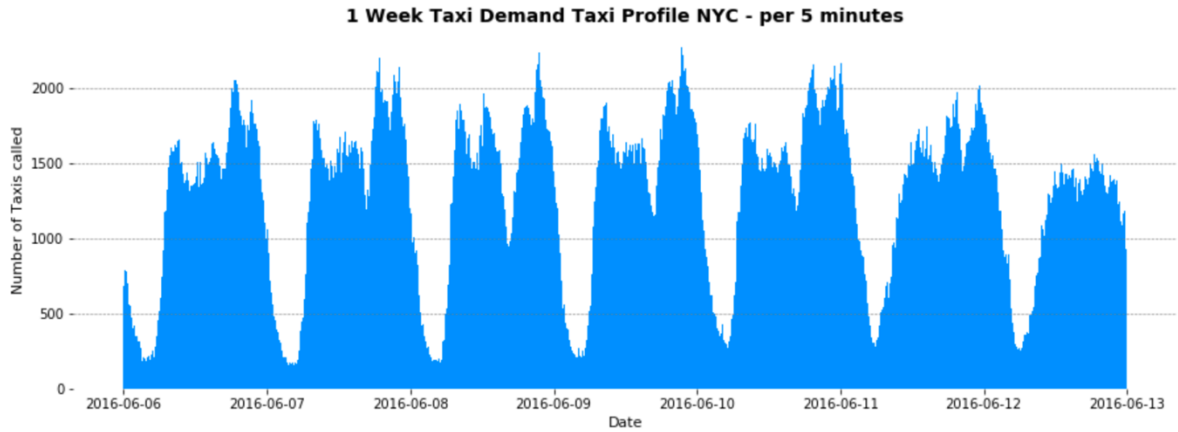

Figure 1: Seasonal Demand Patterns throughout the week

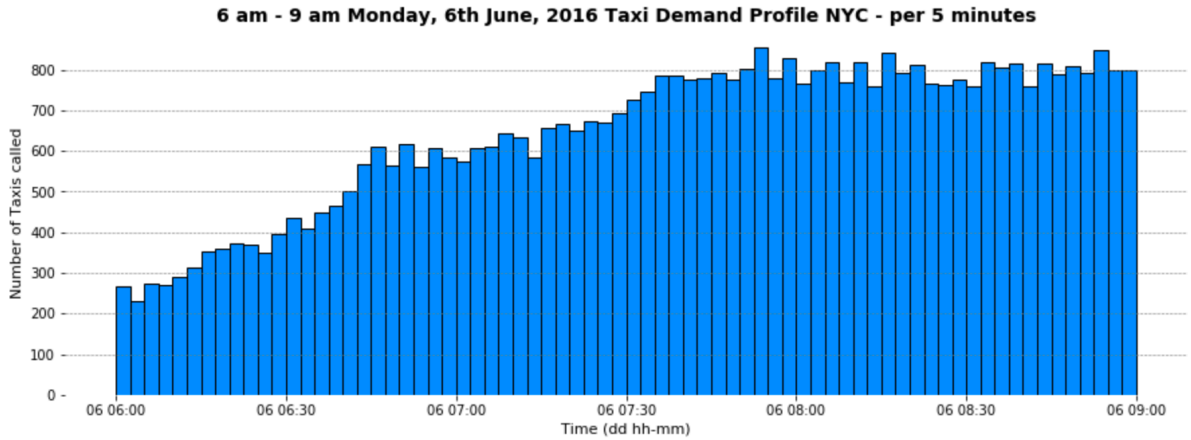

Figure 2: Demand from 6am - 9am June 6th 2016

Between 6 am and 9 am, a large increase in demand can be observed. The demand thus moves from near its lowest point to a local peak quite rapidly. As stated above for reasons of computing complexity and time, I thus use this data for the algorithm implementation. The total taxi hire events in this subset is 46,390 which is sufficient to give a good insight into potential for pooling rides.

Complexity of the Algorithm and Scaling

The algorithm is greedy as it looks for looks for all possible matches for a given taxi hire. The worst-case runtime scenario is as follows: All taxi rides occur within the time and distance threshold such that the matching rides datagframe contains all the entries as the pre-filtered data. If then all but the last matching trip do not meet the occupancy criteria (i.e. vehicle_occupancy + next_occupancy > 4) then the algorithm would have a complexity of $O(n^2)$. This quadratic time algorithm is computationally expensive.

Furthermore, the function calls to calculate the distance between each origin-origin and destination-destination pairs via the external geopy API is also computationally expensive.

In a production environment, such an implementation would require the use of a cloud-based distributed computing framework such as Apache Spark with sufficient parallel processing power and bandwidth.

Question 3. Based on your implementation in the previous question, use visualizations to show how efficiency varies with time and location. Discuss any potential business implications based on your findings.

Overall, the pooling algorithm increases the utilization of seats from 29.8% to 68%. This is equivalent to increasing the average number of passengers in the non-full taxis from 1.2 to 2.8 per ride.

Let's compare the seat occupancy distribution amongst all rides. In Figure 3, it can be seen that when pooling occurs, it results mostly in a full taxi/full utilisation of available seats. This is excellent news for riders, as it would mean, on average, their trip fare would be significantly lower than if booking a private ride.

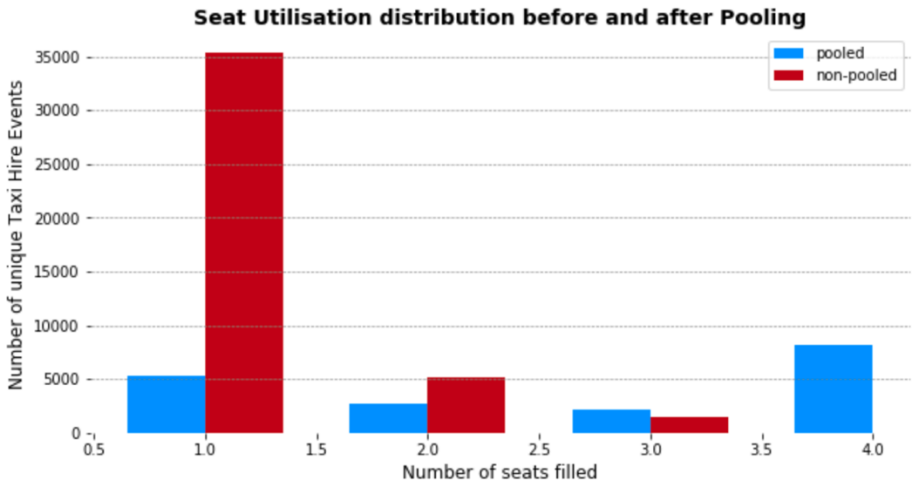

Figure 3: Seat Utilisation before and after pooling

Similarly, as seen in Figure 4, the number of taxis which are pooled increase and decrease in a pattern which closely mirrors that of the individual taxi calls. This suggests that the matching criteria for pooling continues to be met throughout the morning. This is a promising sign, as it means that a pooling approach to transport would not undergo sudden demand rises and drops but adjusts in the same way as regular taxi demand.

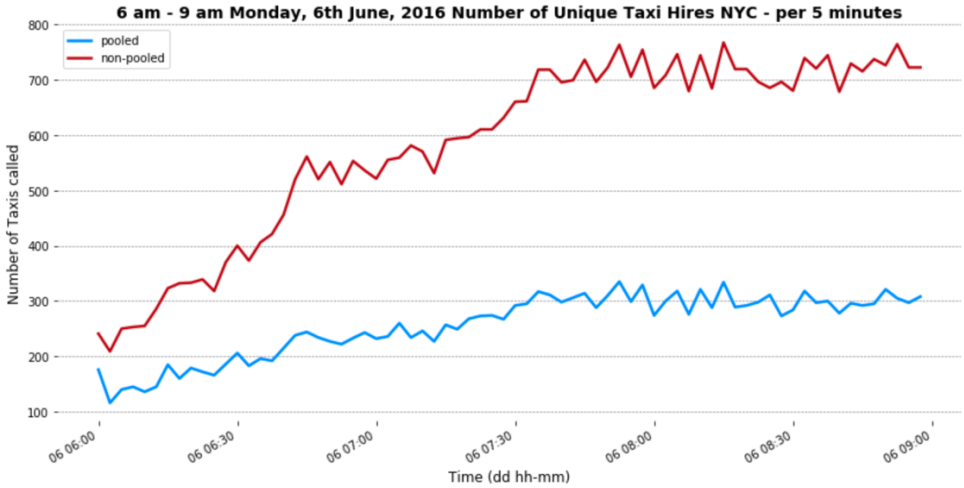

Figure 4: Pooling Demand throughout time

Finally, to get some geographical insight into the effectiveness of pooling, I use the NYC shapefile to plot the metrics. I split the pooled data into 3 sections: 6am-7am, 7am-8am, and 8am-9am. Plotting the efficiency metric (of the origin location) as coloured data points gives insight into where areas of high and low efficiency occur.

In Figure 5, it can clearly be seen that Manhattan provides the highest efficiency through all 3 time periods. Particularly as demand increases throughout the morning, efficiency increases (more people heading to work around 9 am). An increase in efficiency is also seen in the west of Brooklyn and Queens. Such insight has a useful business implication as it suggests that operating in west Brooklyn and Queens may be profitable at later morning hours i.e. the area of service for Via could be dynamically adjusted per time of day when demand patterns in these boroughs make it profitable.

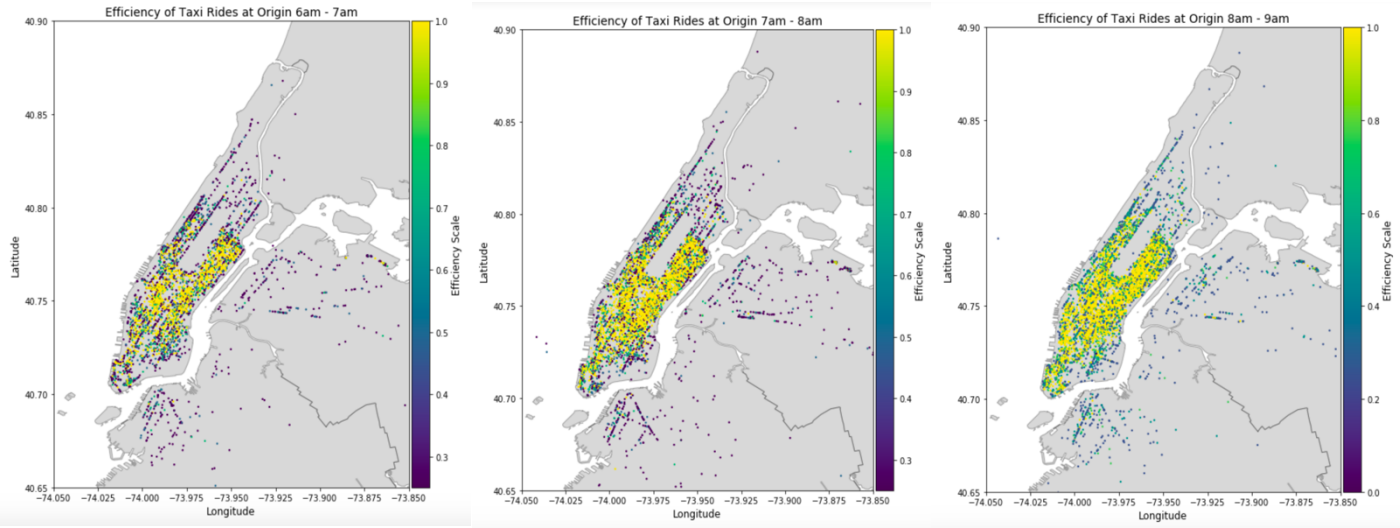

Figure 5: Geographic insights into pooling efficiency

Part 2: Statistics

Question 1. Assume Via has a demand prediction algorithm for predicting hourly demand, 24 hours in advance. Answer the following questions theoretically:

- **a) How could you compare the performance of this algorithm across cities of varying size? Propose a way to predict performance of the demand algorithm in a city we're considering operating in.**
- **b) A data scientist builds a new demand prediction algorithm, but it is more resource intensive than the current one - we'll only switch if we're sure the new one is better. How would you determine if the new algorithm is worth it?**

a) Consider two different cities. Firstly, to compare the algorithm across different cities in which Via currently operates, one would aggregate the predicted and true demand values of each city across a given time period and compare the aggregation of errors across the different cities (e.g. RMSE). If the two cities are very different, then we are likely to see differences in the accuracy of prediction. This is due to the fact that machine learning algorithms are based around finding the target function from a set of candidate functions, in this case to predict demand, given a set of input variables. However, this demand function may well be different for the two cities as each will have its own interactions between factors (inputs) such as population size and density, average age and income, public transport options and layout, competitors etc. just to name a few. The coefficients (weights) of these inputs would vary from city to city which would change the demand predictions.

Thus, to predict the performance of the demand algorithm in a city in which Via is considering operating in, one would look for a city with similar characteristics to the one in question and therefore would have similar associated coefficients to the inputs to the demand prediction function. Thus, the predicted performance will likely be similar to a closely "matched" city in which the algorithm is already in use.

b) This decision must be made by analysing the balance between the high implementation costs and potential economic gains of a better prediction algorithms. Any algorithm would need to be tested before being implemented. This testing phase would not be subject to the same time constrictions

and associated computing speed requirements, meaning the expansion in resources would not be required during testing of the algorithm - it can be done on existing hardware. Since taxi demand is highly seasonal (as seen in Part 1), a past demand profile would closely match a future one. As such, a high-quality training and testing dataset is readily available. Due to the seasonality, one can have high confidence that the performance on the past data is a strong indicator of the algorithm's performance on future data. Both the new and existing algorithm can be compared in this fashion.

Assume now that the proposed algorithm has superior performance to the existing algorithm. A cost analysis would need to be carried out which compares the cost of implementation (computing resource allocation, work-hours required, etc.) to the potential economic gains of better prediction (better driver assignment, greater rider and driver confidence in app - thus greater utilization). A cost-revenue analysis would then yield an economic outcome. If the potential revenue increase from a better prediction algorithm is greater than its associated cost, it may well be worth implementing it.

Question 2. A report claims that between 22.4% and 43.9% of Via rides have excellent music. You can assume that "excellent" is a binary decision at the ride level. What do you think the sample size was?

Since many riders use Via, the total population is likely to be large and so I use Cochran's approximation rule to find the upper and lower bounds of the probable sample size. The formula is as follows:

$$n_o = \frac{Z^2 \cdot p \cdot q}{e^2}$$

where:

- Z is the z-value - in this case of the 95% confidence interval (1.96)
- p is the proportion of the population which has the given attribute - 0.224 and 0.439
- $q = 1 - p$
- e is the desired precision - 0.05 in this case

Thus, we have:

$$n_{o(0.224)} = \frac{1.96^2 \cdot 0.224 \cdot (1 - 0.224)}{0.05^2} \approx 267$$

$$n_{o(0.439)} = \frac{1.96^2 \cdot 0.439 \cdot (1 - 0.439)}{0.05^2} \approx 378$$

At the 95% confidence interval, I thus assume that the sample size was between 267 and 378 persons.

//END OF DATA CHALLENGE