Best Pickup Locations for Über Drivers

Aashish Dugar
Department of Computer Science
New York University
ad4025@nyu.edu

Chun Ming Ho
Department of Computer Science
New York University
cmh531@nyu.edu

Yash Jalan
Department of Mathematics
New York University
yj627@nyu.edu

Preet Gandhi
Center for Data Science
New York University
pg1690@nyu.edu

Abstract

Ride sharing services have changed our ways to get from point A to point B. Passengers no longer need to wait in line at the taxi station or hail at an incoming taxi. Ride-sharing companies, like Über and Lyft, have allowed passengers to request for a driver to pickup at their location. On the drivers' side, the ride-sharing app connects the driver with nearby pickup requests. However, drivers are often idle on the road to wait for the next pickup. Here we propose a prediction model to provide drivers with potential pickup locations with highest expected pickups based on a list of key features, such as time and weather. Idle drivers can benefit from our prediction model to get to a nearby location with higher probability of picking the next passenger.

NOTE: Please ignore the footnote on this page. It comes with the package we've used.

1 Problem Setting

Uber is currently the most popular ride-sharing company. The company operates in 83 countries and over 674 cities worldwide. In the United States alone, the company has over 77% of US ride-hailing market (ny.curbed.com, fortune.com). For this project, we set our focus on Über's historic ride data from 2014 to 2015. Even though taxi and other ride-sharing companies operate in New York City, we decided to focus on Über given its increasing popularity and large volume of data.

As Uber continues to grow as a company, it attracts more drivers and passengers to use its ride-service system. In New York City, there are currently 60,000 drivers working for ride-sharing companies. Meanwhile, there are only 13,587 yellow cabs in the city (ny.curbed.com). Given that there are many Uber drivers in urban cities, like New York, we assume that most of the drivers will spend time waiting for the next passenger request.

A partial solution to this problem was introduced by Uber to have surge pricing that is determined by basic principals of supply and demand. When the amount of ride request increases (given the number of drivers stays the same) or the number of drivers decrease (while amount of ride request stays the same, the price for each ride will increase according and vice versa. This solution helps to lead more drivers to neighborhoods with price surging, but it is not predicting the number of pickup requests for the driver in the near future. Therefore, we decided to develop a prediction model for our project to provide drivers with expected number of pickups in nearby neighborhoods, so they will be able to reduce their time idle to wait for the next request.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Our goal for the project is to predict the expected number of Über pickups at a specific neighborhood in a given time interval.

2 Data Sets

The following datasets were used:

2.1 Über taxi data

This dataset consists of Über time series data for 2014 and 2015.

2.1.1 Über 2014

The Über 2014 data has over 4.5 million Über pickups in New York City from April to September 2014. The source of the data is Github.com/fivethiryeight/uber-tlc-foil-response. Each record is in the following format.

Header	Definition
Date/Time	Timestamp of the Über pickup
Lat	Latitude of the Über pickup
Lon	Longitude of the Über pickup
Base	The TLC base company code affiliated with the Uber pickup

2.1.2 Über 2015

The Über 2015 data has over 14.3 million Über pickups from January to June 2015.The source of the data is Github.com/fivethiryeight/uber-tlc-foil-response. Each record is in the following format.

Header	Definition
Dispatching_base_num	The TLC base company code of the base that dispatched the Uber
Pickup_date	The date and time of the Uber pickup
Affiliated_base_num	The TLC base company code affiliated with the Uber pickup
locationID	The pickup location ID affiliated with the Uber pickup

2.2 New York City Weather

Weather is often a good indicator for ride service demand, especially in urban cities. When it is raining outside, we assume that more people are likely to request for a taxi ride to avoid getting wet. When a snow storm is in the city, we assume that most people will likely to stay indoor. With these assumptions, we decided to include weather data as features to our prediction model.

The source of the weather data is OpenWeatherMap.org. We collected the records for New York City from 2014 to 2015. Each record includes a timestamp with an hourly interval and six other features: temperature, humidity, pressure, weather type, wind direction, and wind speed.

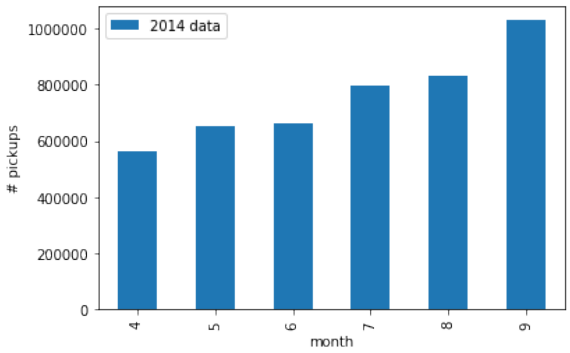
datetime	temperature	humidity	pressure	weather_type	wind_direction	wind_speed
2014-09-07 10:30:00	290.48	82	1016	light rain	340	2
2014-09-07 11:00:00	290.21	77	1016	fog	310	2
2014-09-07 11:30:00	290.21	77	1016	fog	310	2
2014-09-07 12:00:00	290.87	82	1017	scattered clouds	300	2
2014-09-07 12:30:00	290.87	82	1017	scattered clouds	300	2
2014-09-07 13:00:00	291.97	77	1018	scattered clouds	330	3
2014-09-07 13:30:00	291.97	77	1018	scattered clouds	330	3

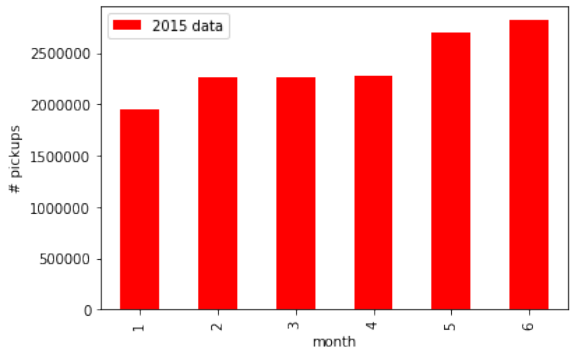
2.3 Federal Holidays

This wasn't strictly a dataset as much as a binary feature that was added to indicate whether a specific day was a federal holiday or not.

3 Data Exploration

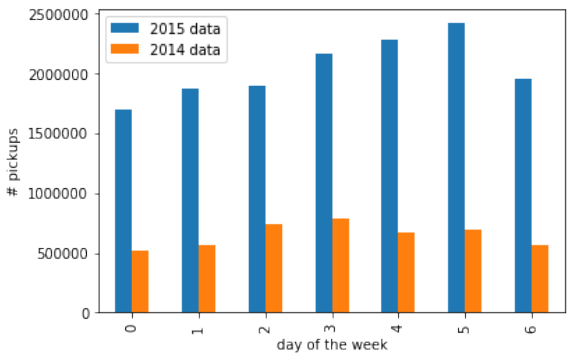
Before doing any preprocessing, we tried to find trends/patterns in the Über dataset to get an insight into the data. Below are two bar charts of the total number of pickups for each month in the 2014 and 2015 dataset.

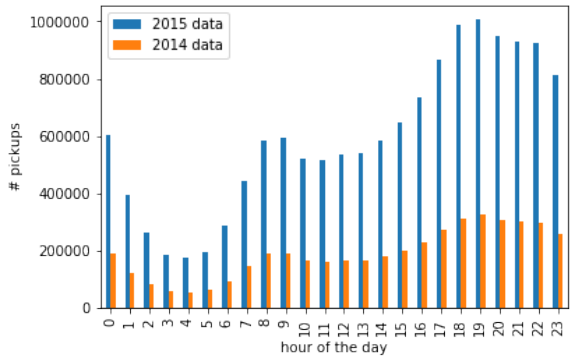




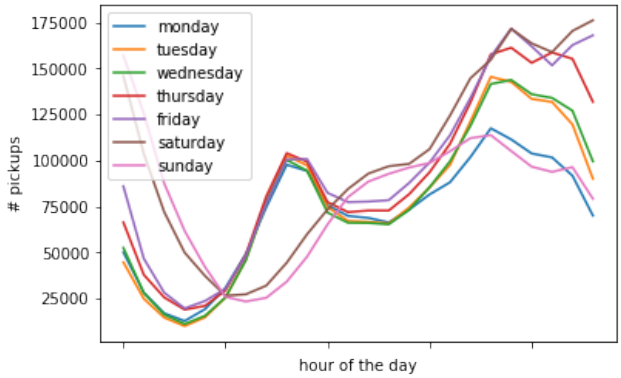
In 2014, the amount of Uber pickups increased each month, possibly reflecting the continuing increase in its popularity. Clearly, there are more pickups in 2015 than in 2014. In 2015 however, the pickups don't increase each month. The months March and April (3-4) see a decrease in the number of pickups, possibly attesting to better weather conditions.

Below are bar charts showing the total pickups for each day of the the week and each hour of the day for the 2014 and 2015 datasets.





It seems that in 2015, most pickups were on Fridays and Saturdays (5-6) between 5:00 pm and 6:00 pm. Below is a line plot that gives a deeper insight into this.



There exists a clear pattern/trend for the number of pickups in 2015 in a single day. Note the separate trends for weekdays and weekends (Saturday, Sunday).

4 Data Preprocessing

By far the most time consuming and difficult section of the project was data cleaning and preprocessing.

4.1 Über data

Since our goal was to predict the number of pickups for a given time interval and area, our first task was to convert locations to neighborhoods in New York City. For 2014, the dataset includes latitude and longitude of a pickup. We used an external API called MapBox to reverse geoencode the latitude and longitude into neighborhoods in NYC. However, the API imposes limited requests, so moving forward we decided to use only the 2015 Über dataset. For 2015, a simple mapping was used to convert the locationID of a pickup to a string represented NYC neighborhood (taxi-zone-lookup-with-ntacode.csv). In total there were almost 260 unique neighborhoods.

On deciding that we wanted to predict the number of pickups in time periods of 30 mins, our next and possibly most challenging task was to group multiple pickups in time periods of half hour for each neighborhood. At first, the dataset was divided into periods of half hour for the entire dataset by assigning a unique number to each half hour in the 6 months time interval. So for instance, all entries satisfying time constraint Jan 1 0:00-00:30 was assigned 0 and Jan 2 0:00-00:30 was assigned 48 (since Jan 1 had 48 different time intervals) and so on. Then for each neighborhood, the set of all entries with a unique date-time interval number were reduced to one single entry with a random time period from that set and new column with the count of all the entries in that set. For instance, the following dataset with entries

Pickup_date	Area	UID
2015-01-01 00:01:00	Lenox Hill West	0
2015-01-01 00:02:00	Lenox Hill West	0
2015-01-01 00:05:00	Lenox Hill West	0
2015-01-01 00:10:00	Lenox Hill West	0
2015-01-01 00:15:00	Lenox Hill West	0
2015-01-01 00:30:00	Lenox Hill West	0
2015-01-01 00:32:00	Lenox Hill West	1
...	Lenox Hill West	...

was converted to

Pickup_date	Area	Count
2015-01-01 00:10:00	Lenox Hill West	6
2015-01-01 00:35:00	Lenox Hill West	10
2015-01-01 01:24:00	Lenox Hill West	...
...	Lenox Hill West	...

Note: Choosing a random specific timestamp from a set of entries with a unique UID was required for using Prophet (our modeling package). Otherwise we could have easily gotten rid of the randomness by making a binary feature that indicated the half hour time period and day of week.

4.2 Weather data

The dataset received from the source was divided into time periods of one hour. Entries were duplicated to match the Über dataset of having time periods of half hour. So, for instance given a record for 11:00, a new record for 11:30 was created with the same field values as 11:00. However, there were several instances of missing fields, which were imputed with the average value for numerical fields and the closest value (based on time) for categorical fields. String categorical values for "weather_type" (which described whether the weather was snow, heavy snow, rain, heavy rain, etc.) were replaced with new columns rain, snow that were assigned numerical values based on whether the weather type was light, moderate or heavy rain/snow.

4.3 Final dataset

The Über dataset was combined with the weather dataset appropriately and a binary feature of holiday indicating whether that day was a federal holiday or not was added to the combined dataset. A one-hot encoding was done for the areas of the pickups. The final dataset inputted into the model looked like (Numbers are made up. The point is to demonstrate succinctly what the input looks like):

ds	y	6_Alphabet City	6_Battery Park	...	snow	holiday
2015-01-01 00:03:00	4	1	0	...	2	1
2015-01-01 00:05:00	1	0	0	...	2	1
2015-01-01 00:05:00	3	0	1	...	2	1
...						
2015-06-30 23:48:00	4	0	0	...	0	0
2015-06-30 23:55:00	1	0	0	...	0	0
2015-06-30 23:56:00	2	0	1	...	0	0

It contains 1,322,691 rows/examples and 268 columns/features.

5 Data Modeling

Having no experience with autoregressive models or Bayesian techniques for time series modeling, we (on your suggestion) decided to use Facebook's Prophet to model and forecast the data. Prophet, a forecasting tool for time series data was released by Facebook earlier this year. The forecast package uses many different forecasting techniques (ARIMA, exponential smoothing, etc), each with their own strengths, weaknesses, and tuning parameters. It uses an "Additive Regressive Model". Choosing the techniques and tuning parameter all happen under the hood. The user is unaware of which techniques are used. Prophet also handles a lot of the challenges by default such as large outliers, historical trend changes,and trends that are non-linear growth curves, where a trend hits a natural limit or saturates. The model is fit using STAN, a platform for statistical modeling and high-performance computation. Quoting their website, "Stan performs the MAP optimization for parameters extremely quickly (<1 second), gives us the option to estimate parameter uncertainty using the Hamiltonian Monte Carlo algorithm."

Prophet was quite easy to install and run. The documentation for the details of the methods were straightforward. Modeling involved only a few lines of code:

```
df = pd.read_csv('final-2.csv')

m = Prophet()

#add all other features except timestamp and target
cols=list(df.columns.values)[2:]
for col in cols:
    m.add_regressor(col)

#train
train=df.loc[0:(1322691-240522-1),:]

#validation= 240522 entries from last month, i.e. June 2015
cols=list(df.columns.values)
cols.remove('y')
valid=df.loc[(1322691-240522):,cols]

#train/fit model
m.fit(train)

#predict
forecast = m.predict(valid)
```

For the training set, we used entries from January to May and we used entries from June as the validation set. In total, there were 240,522 entries in the validation set.

6 Results, Evaluation and Plots

It took approximately 2 hours to run the model on a computer with 16GB RAM. The final output/prediction looks like:

ds	trend	yhat	yhat_upper	yhat_lower	seasonalities	...
2015-06-01 00:05:00	8.536669	2.575003	16.616040	-12.293565	0.608250	..
2015-06-01 00:06:00	8.536669	18.261268	32.981968	3.845832	0.608250	..
...						
2015-06-30 23:59:00	8.558754	20.277500	35.160594	5.253404	1.598266	...
2015-06-30 23:59:00	8.558754	5.102937	20.148062	-9.510802	1.598266	...

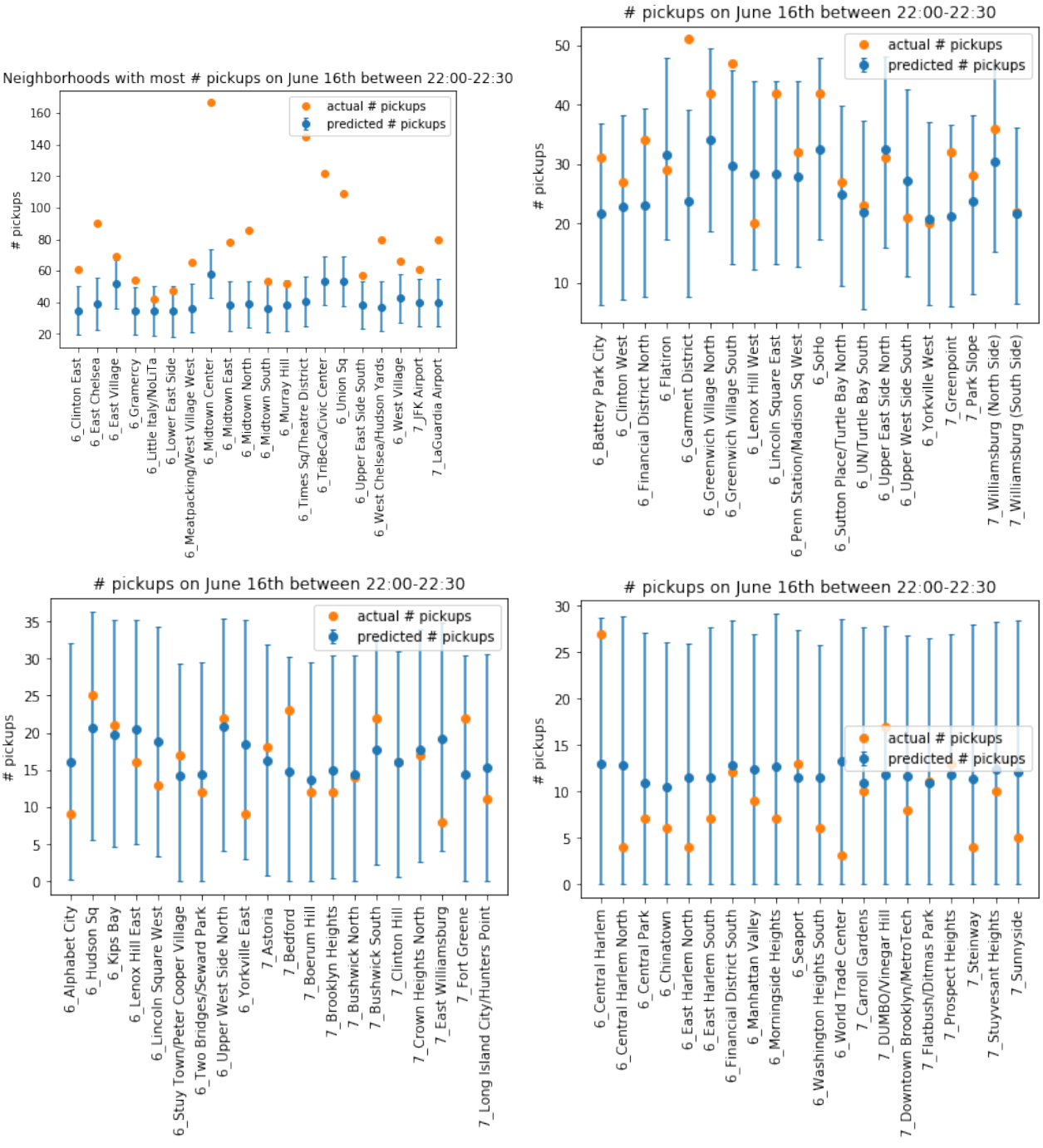
Prophet outputs a feature 'yhat' as an estimate for 'y',i.e. the expected number of pickups given a timestamp (random entry from half hour time period), area, and weather feature. It also produces an uncertainty interval giving upper and lower estimates of 'yhat'. Note that some 'yhat_lower' values are negative even though the number of pickups can never be negative. The model that Prophet uses does not restrict output to only positive values. There are also other interesting features added in the prediction, for instance 'seasonalities' and 'trend'. It possibly represents some kind of correlation coefficient detecting change in season or pattern.

For evaluation of the output ('yhat', 'y') in the validation set, we used mean square error (MSE) and mean absolute error (MAE). An obstacle that arose while computing MAE and MSE (and also for generating plots later below) was that since there were entries with the same timestamp (each entry representing different neighborhoods), the ordering of the prediction did not exactly align with the original dataset. Our solution was to do evaluation for each neighborhood separately and then average them. The following were the final evaluation scores:

MAE = 5.073039992794614

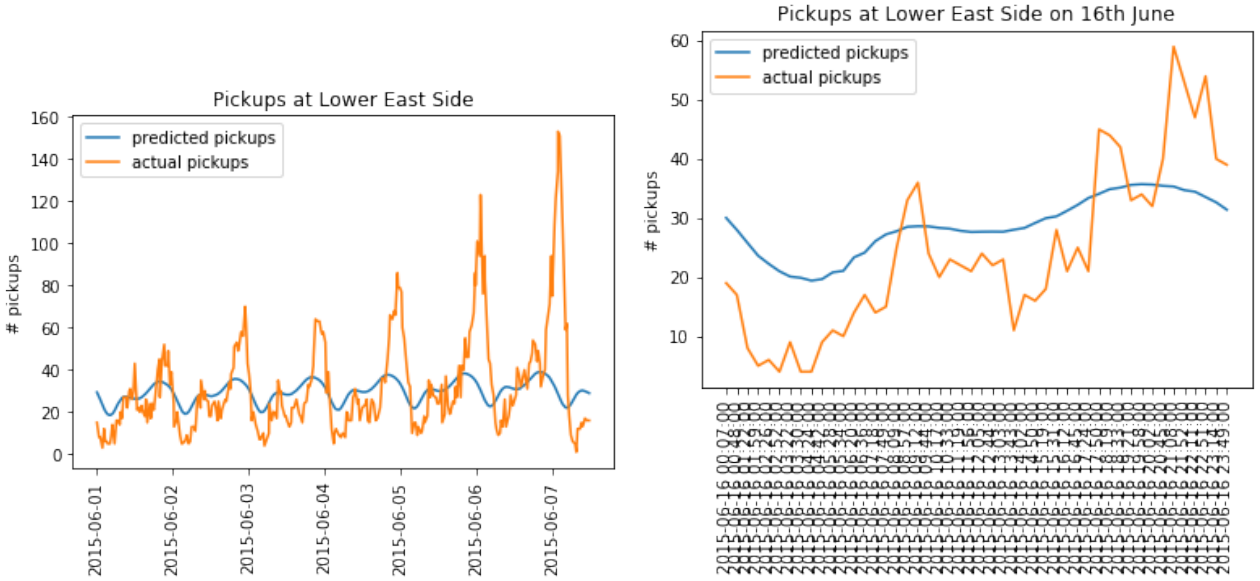
MSE = 98.71144416344363

Since our goal is to help Über drivers get an estimate of the number of pickups for a half hour time period in areas near him/her, below are four plots that would be outputted if a user wanted to know where (which neighborhood in NYC) the most number of expected pickups on June 16, 2015 between 22:00 and 22:30 would be (chosen randomly).



The top left plot estimates where the most number of expected pickups is likely to happen. It looks like model captures the trend, i.e. it can detect which neighborhoods are likely to have more pickups. For instance, it can detect that Midtown Central will have the highest number of pickups at that time. However, the scale for the estimated pickups is a lot lower than the actual. As we move to the areas with lesser amount of pickups (moving from top left to top right to bottom left to bottom right), the difference in scale reduces and the estimates fall very close to the actual results. Not included here, but estimates for the least amount of pickups are in fact negative. It seems that prophet tries to moderate extremely high values, but keep the ranking of neighborhoods (according to amount of pickups) in order.

Below are two line plots of the amount of pickups at Lower East Side from June 1-Jun 7 and of June 16.



It is clearer now that the model captures and moderates the trend of the number of pickups over time. It correctly predicts that in Lower East Side the number of pickups increase in the evenings and reach their highest at nighttime.

7 Conclusion

Even though our base prediction using Prophet gave decent evaluation scores and the model captured some important trends, the Prophet package does not provide users the freedom to choose models and parameters. If allowed to choose parameters/models, we could have potentially avoided negative outputs in our prediction. In the future, we could (1) learn about and use other forecasting packages and models, (2) use the yellow taxi data and 2014 Über data as additional examples, and (3) use additional features such as superbowl sunday, etc. to possibly get more accurate estimates.

References

https://facebook.github.io/prophet/docs/*.html

http://fortune.com/2016/10/20/uber-app-riders/

https://www.kaggle.com/selfishgene/historical-hourly-weather-data/data

https://ny.curbed.com/2017/1/17/14296892/yellow-taxi-nyc-uber-lyft-via-numbers

https://github.com/toddwschneider/nyc-taxi-data/blob/master/data/taxi-zone-lookup-with-ntacode.csv

https://research.fb.com/prophet-forecasting-at-scale/

https://arxiv.org/pdf/1709.01907.pdf