Location Based Demand Prediction for the NYC Ride Sharing And Taxi Market

CS6220 Big Data Systems & Analytics

Marcel Gwerder, Xiaoying Ji, Andrew Jo, Feng Xiao

Agenda

* Background
* Goals
* Proposed Concept
* Datasets
* Data Processing
* Modeling
* Evaluation of Results
* Architecture
* User Interface
* Summary
* Lessons Learned
* Q&A

Background

* Ride Sharing and Taxi Market

World’s Taxi Market Value currently valued >$100 billion and is highly fragmented

Ride Sharing, leading as a major building block of the “sharing economy”

- Exponential growth in recent years:

1. Over 1.5 million drivers and 42 million passengers, and 2 billion annual rides.
2. Uber’s expected IPO of $120B in 2019 (more than Ford, GM, and Chrysler combined)

* Pain points
    - Unpredictable wait time and fare income
    - Nonoptimal supply and demand

Goals and Plan of Activity

* Innovations
* Provide an option for the companies to be proactive
* Risks
* Internal goals and distribution for the project:

Levels	Details	Note/Deadline
Level 0	Data Visualization/UI	11/11/2018
Level 1	Prediction	11/29/2018
Level 2	Utilize additional variables	Realistic Goal
Level 3	Additional predictions	Reach Goal

Proposed Concept

- Grid-based Model Training
  - Dividing the minimum square containing NYC into 30*30 grids
    - Mapping coordinates of the data to the corresponding region
    - Aggregating data in the same region
  - Train and store the model in Cloud
- Prediction
  - Input: timestamp, weather, location
  - Output: predicted number of customers in each region
- Heatmap
  - Heatmap color depending on demands prediction in the selected day
  - The user pick up a day in the future

Datasets

- Uber Pickups in NYC
  - Source: Kaggle
  - 4.5 million Uber pickups from 2014
  - Date/time, Latitude, Longitude(Pick up & drop off)
- Taxi & Limousine Trip Data in NYC
  - Source: NYC Taxi & Limousine Commission
  - 165 million data from 2014
- (Level 2) Weather Data
  - Source: Weather.com
  - Temperature(in hour), precipitation(rain, snow)...

Data Processing

- Used Amazon Athena to handle size of the datasets (~24GB, 165 mio rows)
- Combine rides with historical hourly weather data
- Remove erroneous data points (e.g. out of range lat/long)

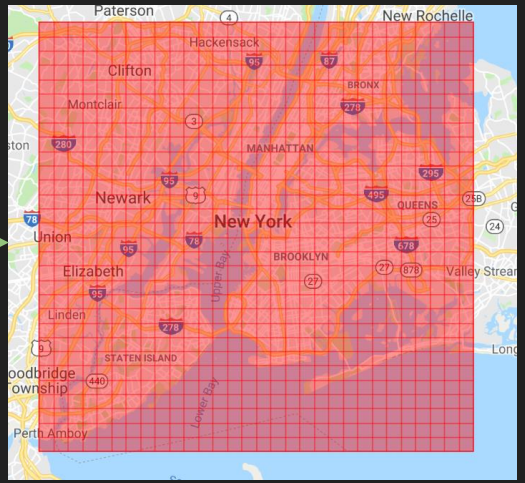
	pickup_datetime	pickup_latitude	pickup_longitude
1	2014-06-29 00:14:00.000	40.732315	-74.007685
2	2014-06-29 00:14:00.000	40.691362	-73.993780
3	2014-06-29 00:14:00.000	40.738440	-73.986000
4	2014-06-29 00:14:00.000	40.721337	-73.997502
5	2014-06-29 00:14:00.000	40.720995	-73.986895
6	2014-06-29 00:14:00.000	40.719027	-73.984927

	pickup_datetime	pickup_latitude	pickup_longitude	temp	precip_hrly
1	2014-08-11 13:40:08.000	40.737787	-73.988175	55.0	10.0
2	2014-08-11 13:40:08.000	40.756623	-73.989976	55.0	10.0
3	2014-08-11 13:40:08.000	40.742605	-73.986720	55.0	10.0
4	2014-08-11 13:40:10.000	40.757687	-73.997051	55.0	10.0
5	2014-08-11 13:40:10.000	40.774054	-73.872902	55.0	10.0
6	2014-08-11 13:40:10.000	40.760001	-73.981041	55.0	10.0

Data Processing

- Map the pickup location to the respective tile in the grid
- Aggregate data to get the demand for every tile and every hour of the year

	demand	day	day_of_week	hour	temp	precip	x	y
1	12	274	Wednesday	5	56.0	10.0	17	16
2	6	274	Wednesday	5	56.0	10.0	16	13
3	24	354	Saturday	18	19.0	10.0	18	19
4	1	354	Saturday	18	19.0	10.0	16	23
5	4	358	Wednesday	20	55.8	2.2	22	18
6	1	337	Wednesday	15	42.5	8.0	16	11



Modeling

- Train model that predicts demand for specific tile at given time in the future
- Alternative is the classification into low/medium/high demand
- Derive reasonable predictors from limited input given e.g. day of week from date/time.

Promising options include (continuous response / non-linear relationship)

➔ Regression Trees / Random Forest
➔ Deep Neural Network

Modeling - Random Forest

- A first test using a random forest with regression trees gives an R^2 of **~0.985**
- Adding predictors one by one shows that the **x,y** combination has by far the biggest impact followed by the **day_of_week** dummy variables and **hour**

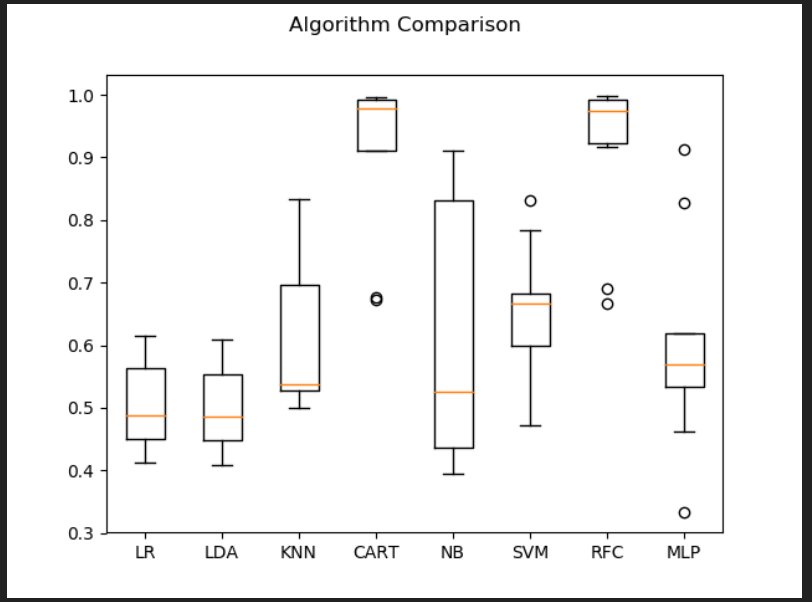
	demand	day	hour	temp	precip	x	y	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
0	6650	306	1	34.5	10.0	15	17	0	0	0	1	0	0	0
1	6395	62	19	0.0	10.0	16	19	0	1	0	0	0	0	0
2	6321	84	19	14.0	10.0	16	19	0	0	0	0	0	1	0
3	6290	105	19	41.5	7.5	16	19	0	0	0	0	0	1	0
4	6267	83	18	2.0	10.0	16	19	0	1	0	0	0	0	0

```
(project-RKpx-M55) dev@laptop7:/mnt/c/Development/CS6220/project$ python model.py
R^2: 0.9859169914729624
Mean Absolute Error: 15.83829224188525
Mean Squared Error: 3369.475871446246
Root Mean Squared Error: 58.04718659372085
```

Modeling - Comparison of Algorithms

- Used 40K rows of data with 6 inputs and 1 output (demand) classified into 1 or 0, depending on where they fall in the mean for the hour and the day. Decided to use Random Forest due to its overall reliable accuracy and speed.

Models	Mean	Standard Dev.	Time (s)
Logistic Regression (LR)	5.1%	7.15	1.29
Linear Discriminant Analysis (LDA)	50.1%	7.08	0.28
K Neighbors Classifier (KNN)	61.1%	11.68	0.69
Decision Tree Classifier (CART)	91.1%	12.15	0.79
GaussianNB (NB)	61.2%	20.13	0.12
Support Vector Classifier (SVM)	65.7%	9.93	1601.33
Random Forest Classifier (RFC)	91.4%	12.07	2.28
Multilayer Perceptron (MLP)	60.0%	15.80	32.22
Average	60.7%	12.00	204.88



Modeling - Hyperparameter Tuning

- Once we have decided to use Random Forest Regressor, we tuned hyperparameters and compared against base parameters of 20 trees and 15 levels.
- Even after increasing number of trees to 200 and depth to 40, the overall accuracy of the model only increased from 60.71% to 61.17%, while taking exponentially longer time to calculate, so we decided to stay with the base parameters for the project.

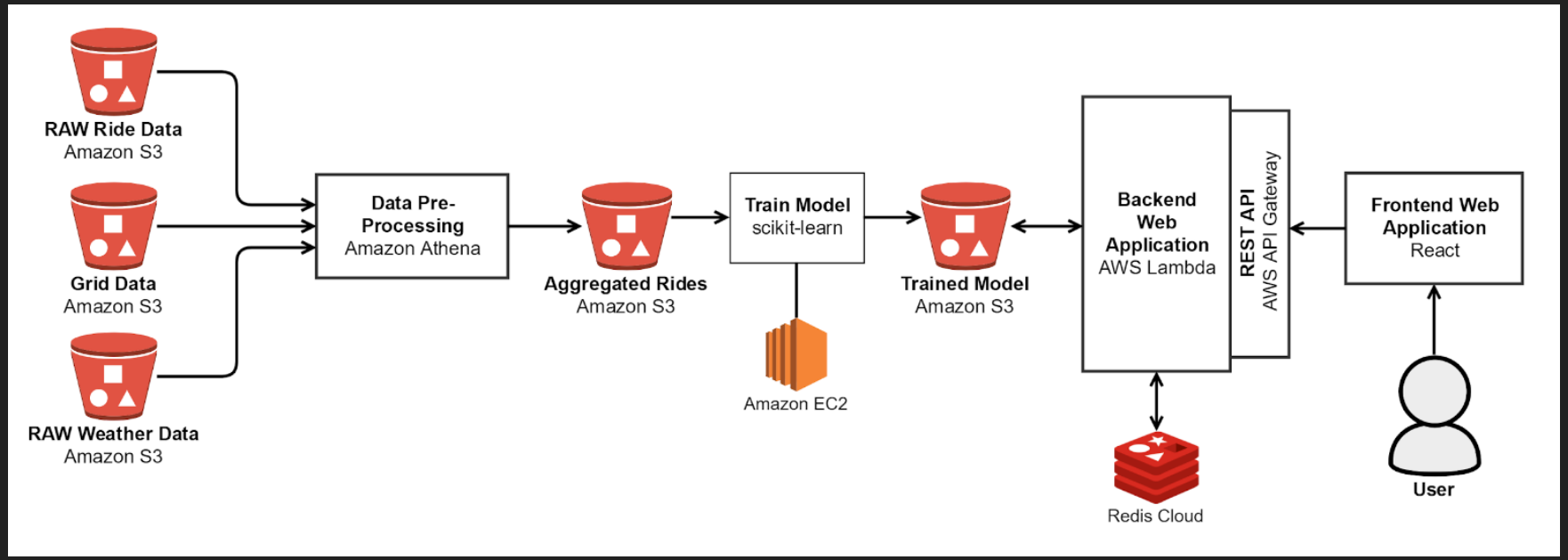
random forest tuning:
Best parameters: {'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 40, 'bootstrap': True}
Best score: 0.985177877164777

(Base) Model Performance
Average Error: 15.8474 degrees.
Accuracy = 60.71%.
(Tuned) Model Performance
Average Error: 15.4759 degrees.
Accuracy = 61.17%.
Testing Accuracy % Change of 0.75% from 60.71% to 61.17% (change of 0.46%).

Evaluation of Results

- Aggregated data still consists of ~1 mio data points which provides flexibility when spitting into training, validation and test datasets
- Comparison of Algorithms: Given the 40k rows of data (~4% of the aggregated data), quickly came to conclusion that decision tree and random forest are the only viable options, as other algorithms are unreliable (overall accuracy and standard deviation), time consuming, or both. When fed with the entire aggregated data, random forest was the superior algorithm due to the complexity and size of the data.
- Hyperparameter Tuning: Randomly conducted 100 different combinations with 10 number of folds for cross validation each from a wide range of provided parameter grid, and came to conclusion that exponentially increasing the number of trees and levels resulted in slight improvements in prediction accuracy. We have thus decided to stay with the base parameters to optimize performance vs running time trade-offs.
- High R^2 indicates a high prediction accuracy but there could also be a problem in the model (needs further investigation)
- Required prediction accuracy is subject to end-user perception
- Used visualization end results to gauge the expected demands from the Random Forest model on an hourly an daily basis

Architecture



Architecture - Main Components

- Data Storage
  - Amazon S3 for raw data and processed data
  - Redis for caching prediction results
- Data Processing
  - Amazon Athena for processing the raw data using SQL
- Prediction API
  - AWS Lambda and AWS API Gateway to create a low cost, highly scalable prediction API
- Frontend
  - React web application which allows the user to select a time up to seven days into the future and see a heatmap based on the predictions made by the model.

Prediction API - Example Prediction

https://owtjarn4j7.execute-api.us-east-1.amazonaws.com/prod/rides?date=2018-12-10&hour=16

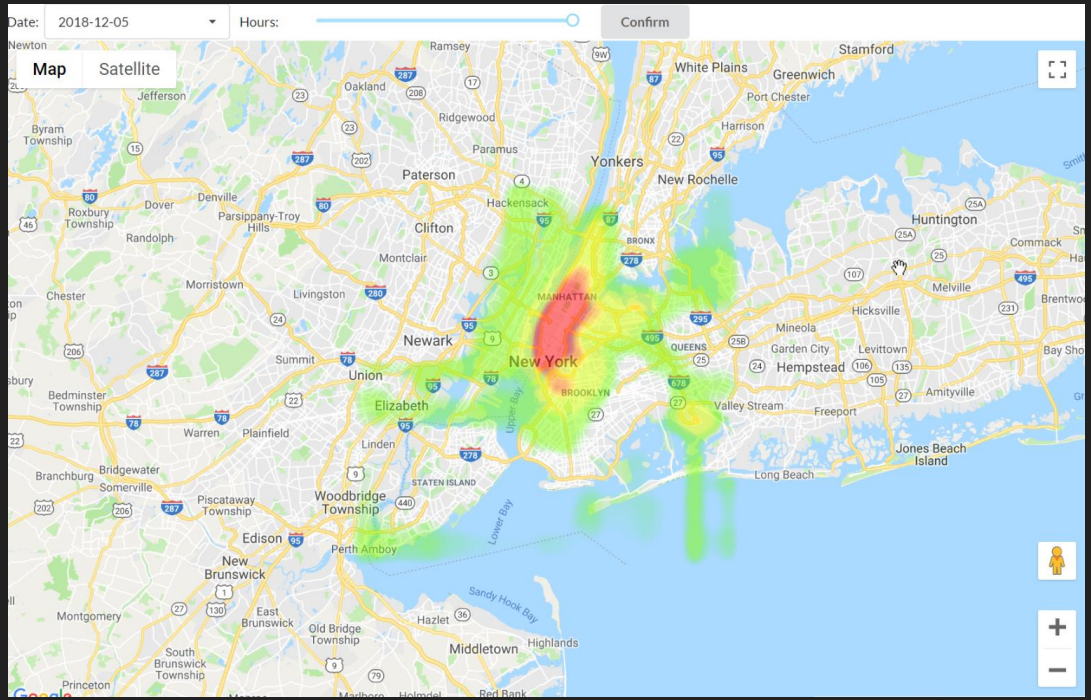

```
[{"x": 0, "y": 0, "demand": 1.8089505440945088, "demand_scaled": 0.00032283925784015385}, {"x": 0, "y": 1, "demand": 1.8089505440945088, "demand_sc
"demand_scaled": 0.00030247942050906466}, {"x": 0, "y": 3, "demand": 1.4974023765191369, "demand_scaled": 0.0001969932903536347}, {"x": 0, "y": 4,
"demand": 1.4847752832177015, "demand_scaled": 0.00019189273468216666}, {"x": 0, "y": 6, "demand": 1.4847752832177015, "demand_scaled": 0.0001918927
0.00021208960639961666}, {"x": 0, "y": 8, "demand": 1.508936132092096, "demand_scaled": 0.00020165220598818492}, {"x": 0, "y": 9, "demand": 2.12091
2.054413142508807, "demand_scaled": 0.00042199079007225 99}, {"x": 0, "y": 11, "demand": 1.402926384409191, "demand_scaled": 0.00015883090049317004}
05}, {"x": 0, "y": 13, "demand": 2.4076923076923076, "demand_scaled": 0.0005646934696654313}, {"x": 0, "y": 14, "demand": 1.0611052754435106, "dema
"demand_scaled": 0.00020067550704113563}, {"x": 0, "y": 16, "demand": 1.062666784436161, "demand_scaled": 2.138731066745475e-05}, {"x": 0, "y": 17,
18, "demand": 1.3096618659601977, "demand_scaled": 0.00012115787019508 98}, {"x": 0, "y": 19, "demand": 1.1138872180728738, "demand_scaled": 4.20771
9.593548579677445e-05}, {"x": 0, "y": 21, "demand": 1.2472205514062071, "demand_scaled": 9.593548579677445e-05}, {"x": 0, "y": 22, "demand": 1.1472
1.296092731857335, "demand_scaled": 0.00011567678897924337}, {"x": 0, "y": 24, "demand": 1.1146965553269303, "demand_scaled": 4.2404082830837034e-0
06}, {"x": 0, "y": 26, "demand": 1.0259194883150193, "demand_scaled": 6.5437021530175e-06}, {"x": 0, "y": 27, "demand": 1.0207682962235287, "demanc
"demand_scaled": 4.462942835743722e-06}, {"x": 0, "y": 29, "demand": 1.0207682962235287, "demand_scaled": 4.462942835743722e-06}, {"x": 1, "y": 0,
"demand": 1.6261919234048539, "demand_scaled": 0.00024901620949361 81}, {"x": 1, "y": 2, "demand": 1.5757884806937188, "demand_scaled": 0.0002286563
0.00012317024200709896}, {"x": 1, "y": 4, "demand": 1.3020166625280463, "demand_scaled": 0.00011806968633563085}, {"x": 1, "y": 5, "demand": 1.3020
1.3020166625280463, "demand_scaled": 0.00011806968633563085}, {"x": 1, "y": 7, "demand": 1.3520166625280463, "demand_scaled": 0.00013826655805307 91
0.00020165220598818492}, {"x": 1, "y": 9, "demand": 2.1209199918238757, "demand_scaled": 0.0004488553961512227}, {"x": 1, "y": 10, "demand": 2.0544
1.402926384409191, "demand_scaled": 0.00015883090049317004}, {"x": 1, "y": 12, "demand": 1.1723303552584023, "demand_scaled": 6.568453210693652e-05
0.00050410285451308 46}, {"x": 1, "y": 14, "demand": 1.0611052754435106, "demand_scaled": 2.0756558731249014e-05}, {"x": 1, "y": 15, "demand": 1.506
1.062666784436161, "demand_scaled": 2.138731066745475e-05}, {"x": 1, "y": 17, "demand": 1.056829351455298, "demand_scaled": 1.9029352965981364e-05}
05}, {"x": 1, "y": 19, "demand": 1.0263872180728737, "demand_scaled": 6.732635711374779e-06}, {"x": 1, "y": 20, "demand": 1.1037681704538262, "dema
"demand_scaled": 3.7989699083616074e-05}, {"x": 1, "y": 22, "demand": 1.1037681704538262, "demand_scaled": 3.7989699083616074e-05}, {"x": 1, "y": 2
24, "demand": 1.071244174374549, "demand_scaled": 2.4852039552577046e-05}, {"x": 1, "y": 25, "demand": 1.0554321158881037, "demand_scaled": 1.84649
1.9614372679381542e-05}, {"x": 1, "y": 27, "demand": 1.0207682962235287, "demand_scaled": 4.462942835743722e-06}, {"x": 1, "y": 28, "demand": 1.020
1.0207682962235287, "demand_scaled": 4.462942835743722e-06}, {"x": 2, "y": 0, "demand": 1.5338842310971614, "demand_scaled": 0.00021172967709217304
0.00021172967709217304}, {"x": 2, "y": 2, "demand": 1.4834807883860264, "demand_scaled": 0.00019136983976108211}, {"x": 2, "y": 3, "demand": 1.2223
1.209708970220354, "demand_scaled": 8.078315393418752e-05}, {"x": 2, "y": 5, "demand": 1.209708970220354, "demand_scaled": 8.078315393418752e-05},
```



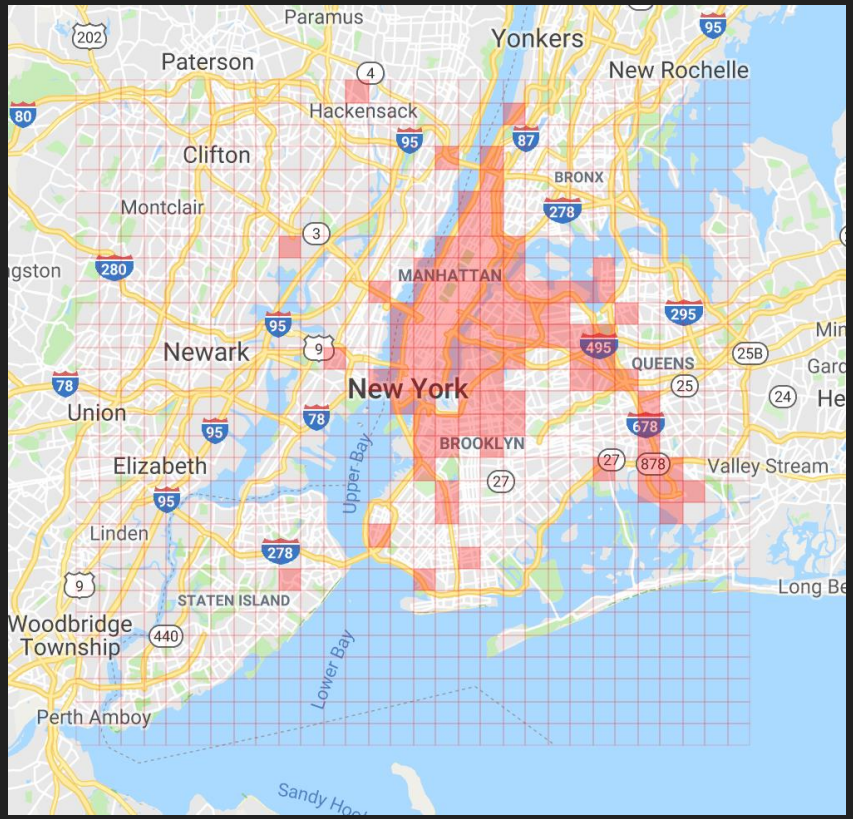
```
{
    "x": 16,
    "y": 15,
    "demand": 3.0067515569591725,
    "demand_scaled": 0.0008066759258373128
},
{
    "x": 16,
    "y": 16,
    "demand": 50.705815297639774,
    "demand_scaled": 0.02007411335409611
},
{
    "x": 16,
    "y": 17,
    "demand": 357.3986979121402,
    "demand_scaled": 0.14395884949048912
},
{
    "x": 16,
    "y": 18,
    "demand": 967.5706859744578,
    "demand_scaled": 0.3904301568599958
},
{
    "x": 16,
    "y": 19,
    "demand": 2476.6406342015757,
    "demand_scaled": 1
},
```

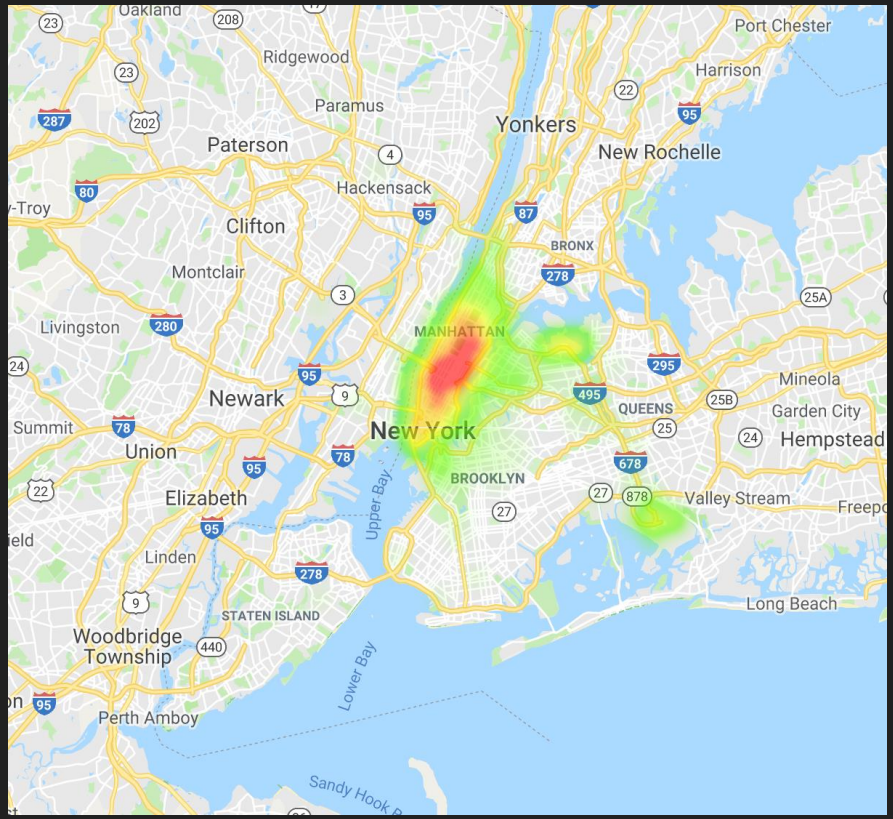
UI

DEMO: https://youtu.be/QhIMlnaKrTk

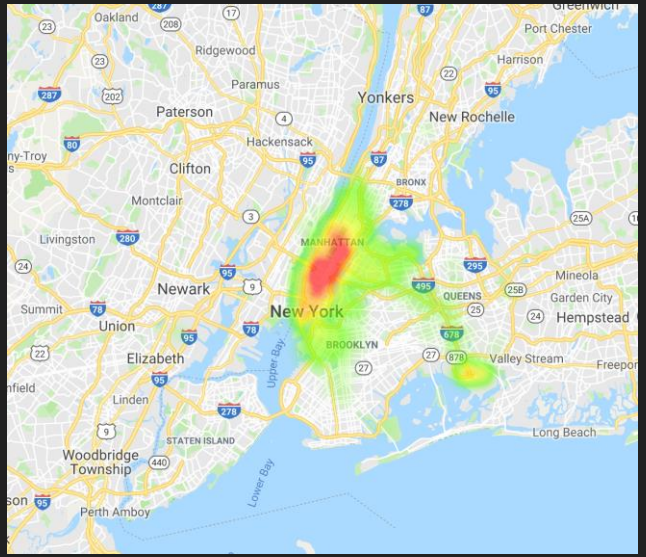


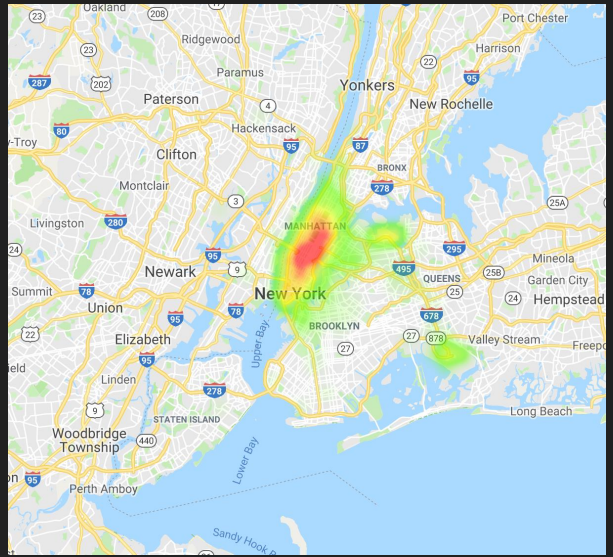
Heatmap of NYC taxi pickup At 12:00pm-1:00pm

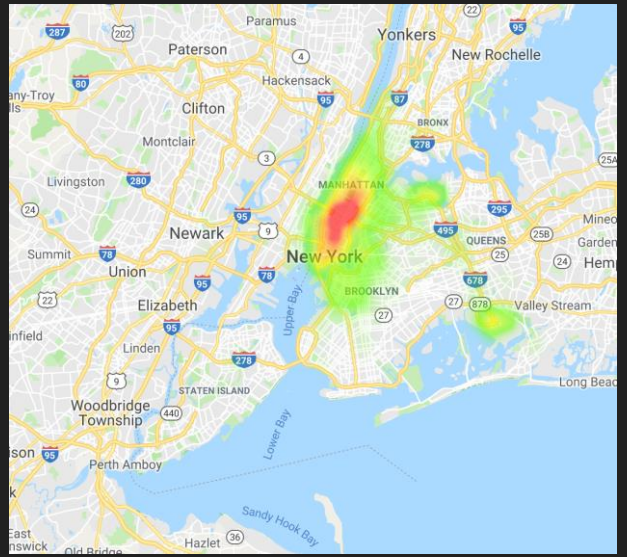




Comparison of Taxi Pickups at Different Times







6am-7am 10pm-11pm 12pm-1pm

Summary

- Ultimately achieved our realistic goal of Level 2, which includes additional predictors such as weather data.
- Successfully aggregated millions of rows of trip data using Amazon Athena and combined with Weather.com API data.
- Ended up not using Uber data because there were months missing and the data has the same structure as the taxi data and therefore the concept is applicable to both.
- Compared numerous algorithms for their accuracy and running time and came to the conclusion that a random forest model is our best bet.
- Hyperparameter tuned (100 iterations with 10 cross validations each) to gauge the expected accuracies and running time across different parameters for Random Forest.
- Experimented with different visualization methods, including Python and React to interact with Google Map API.
- Built a prediction API based on AWS Lambda and accessible through API Gateway

Lessons Learned

- Realized how important the prediction speed can be for specific applications of machine learning. Our grid required 900 predictions and the API was built to predict on demand. Therefore we had to make a trade off between accuracy and speed.
- Learned pros and cons of using proprietary and open-source libraries and packages. The Google Maps Heatmap limited our ability to visualize our predictions to their full potential by not allowing us to easily normalize the data across different hours.
- Cloud services such as Amazon Athena are extremely powerful and convenient. We were able to process a relatively large dataset without much issue and at a low cost using SQL.
- Preparing and cleaning the data for training the model is a lengthy process which has a big impact on the predictive quality of the model.