Analyzing 1 Billion+ NYC Yellow Taxi and Uber Rides for Taxi Drivers

Jialei Zheng
Ruilin Zhong
Shengjia Zhang

Project Overview

1. **Uber v.s. Yellow Taxi in Manhattan?**

Analyze and compare relationship between ride amount of yellow taxi and Uber, at different locations and time

2. Predict demands for Yellow Taxi in specified area at specific time

Given pickup location, time, weather condition, etc., apply machine learning algorithms to predict:

- Ride amount requested by passengers
- Average fare amount
- Average tip amount

3. Develop a web app to inform Yellow Taxi drivers of real-time predicted ride amount, distance, and tip, so that riders could make better pick-up decisions

Business Value and Social Value

1. Helps NYC yellow taxi drivers - underprivileged population
2. Modeled on “Big Data” (with 20G + size, ~ 24 million data-points), can potentially scale up to predict Uber’s ride requests, distance & fare amount., and other parameters of interest
3. Can scale up to other cities out of NYC

$$$HUGE BUSINESS VALUE$$$ for Drivers, Passengers, Car-sharing companies, and governments!

Data Source

- The TLC Yellow Taxi dataset
  - Pickup & drop-off locations, trip time & date, fare, tips
  - *10+ million* rides per month
  - From *Jul 2015 to Jun 2016* and *Apr 2014 to Sep 2014*

- Uber trip data
  - Date/time, pickup coordinates
  - Incomplete: Apr 2014 to Sep 2014

- Historical weather data for each day
  - Fetched from http://weathersource.com/
  - Collect each day's weather data into a dictionary

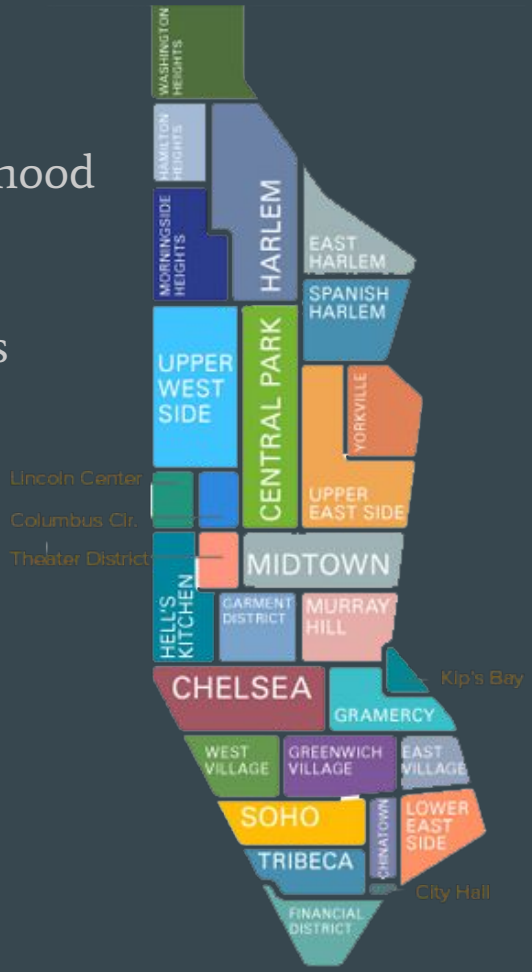
Data Processing

- Location: from coordinate (latitude, longitude) ---> neighborhood
  e.g. (40.762344, -73.982364) ---> Midtown

  Using **Ray-casting algorithm** that finds out if a given point lies within a predetermined area defined by a 2-D polygon

- Pickup time: divided into hours.
- Weather: collect historical weather data for each day.
- Business day or not
- Temperature in Fahrenheit

For each unique combination of:
(neighborhood, hour, is_business_day, weather, temperature)
Count the DAILY total rides, average fare and tips.



Data Processing

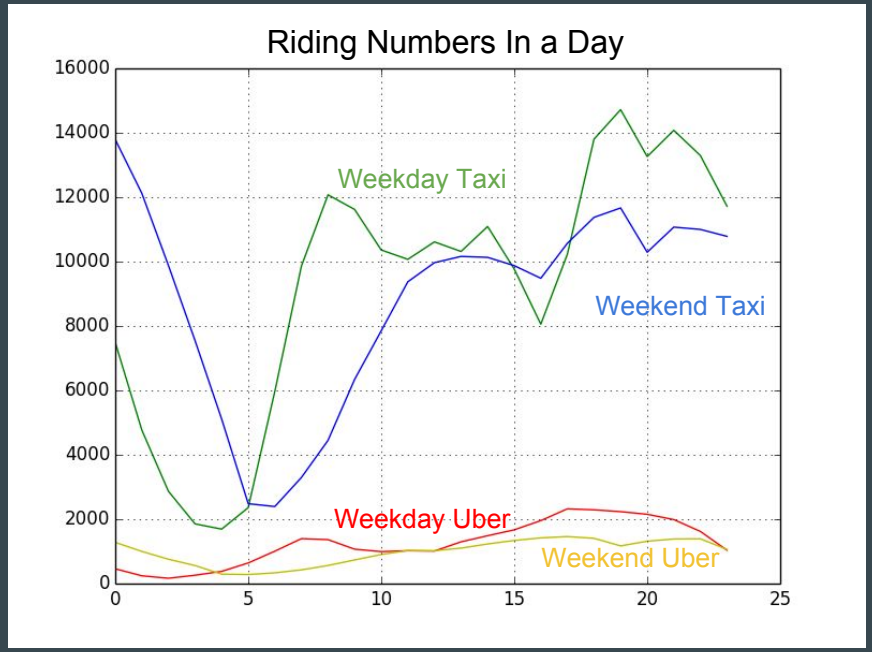
VendorID	tpep_pickup_datetim	tpep_dropoff_datet	passenge	trip_di	ckup_lo	pickup_la	Ratecode	store_and_fw	dropoff_	dropoff_	payment_t
2	2015/12/1 0:00	2015/12/1 0:05	5	0.96	-73.9799	40.76538	1	N	-73.9663	40.76309	1

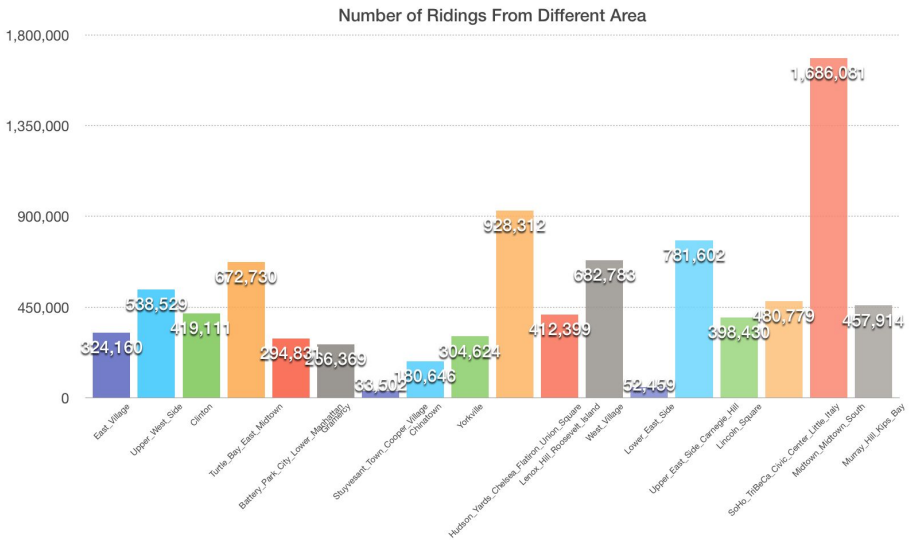
pickup_area	pickup_hour	trip_distance	fare_amount	tip_amount	is_business_day	weather	temperature
Clinton	6	0.1	456.78	0	1	sunny	49.3
Hudson_Yards_Chelsea_Flatiron_Union_Square	19	82.91	415	10	1	rain	46
West_Village	3	0.5	390	0	1	sunny	49.3
West_Village	23	37.41	350	133	1	rain	46
Turtle_Bay_East_Midtown	13	0.4	349.7	0	1	sunny	49.3
Murray_Hill_Kips_Bay	23	72.94	339.5	0	1	sunny	45.5
Hudson_Yards_Chelsea_Flatiron_Union_Square	22	59.3	300	90.09	1	rain	46
Hudson_Yards_Chelsea_Flatiron_Union_Square	0	0	285	0	0	sunny	43.5

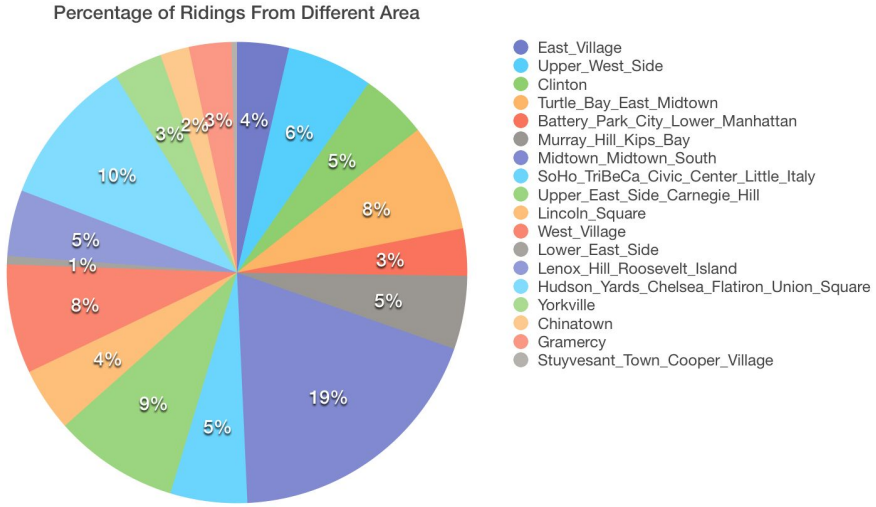
2	2015/12/1 0:00	2015/12/1 0:10	6	2.08	-73.9827	40.73131	1	N	-74.006	40.74523	2
1	2015/12/1 0:00	2015/12/1 0:00	1		3.9902	40.7562	5	Y	0	0	2
1	2015/12/1 0:00	2015/12/1 0:05	1		3.9958	40.74379	1	N	-74.0026	40.73055	1
2	2015/12/1 0:00	2015/12/2 0:00	1	0.9	4.0054	40.72728	1	N	-73.997	40.72539	1
2	2015/12/1 0:00	2015/12/1 0:09	1	1.73	-73.9993	40.72832	1	N	-73.9809	40.73778	1

area	hour	is_business_day	weather	temp_cat	trip_count	avg_fare	avg_tips
Upper_East_Side_Carnegie_Hill	19	1	sunny	80	5724	9.141	1.345
Chinatown	5	1	sunny	50	349	16.431	1.918
Lenox_Hill_Roosevelt_Island	1	1	sunny	50	1148	12.053	1.527
Clinton	2	0	sunny	40	1818	13.065	1.702
Hudson_Yards_Chelsea_Flatiron_Union_Square	8	0	sunny	60	918	9.542	1.275
Gramercy	9	0	sunny	70	827	9.57	1.415
Lenox_Hill_Roosevelt_Island	22	0	rain	70	334	10.092	1.414
Battery_Park_City_Lower_Manhattan	19	1	sunny	40	719	15.471	2.742
Murray_Hill_Kips_Bay	2	1	rain	60	103	13.932	1.591
Lenox_Hill_Roosevelt_Island	13	1	sunny	50	5458	10.981	1.282
Lenox_Hill_Roosevelt_Island	4	0	sunny	40	399	14.238	1.736
Battery_Park_City_Lower_Manhattan	6	0	sunny	70	251	20.165	2.786

Visualization: Exploratory Data Analysis







ML Algorithms: Features and outcome variables

Objectives: predict *3 continuous* outcome variables

1. Total number pickup rides (pickup_counts)
2. Expected (average) trip fare (after-tax but before-tip)
3. Expected (average) tip fare

Based on *3 categorical and 2 continuous (numerical)* features:

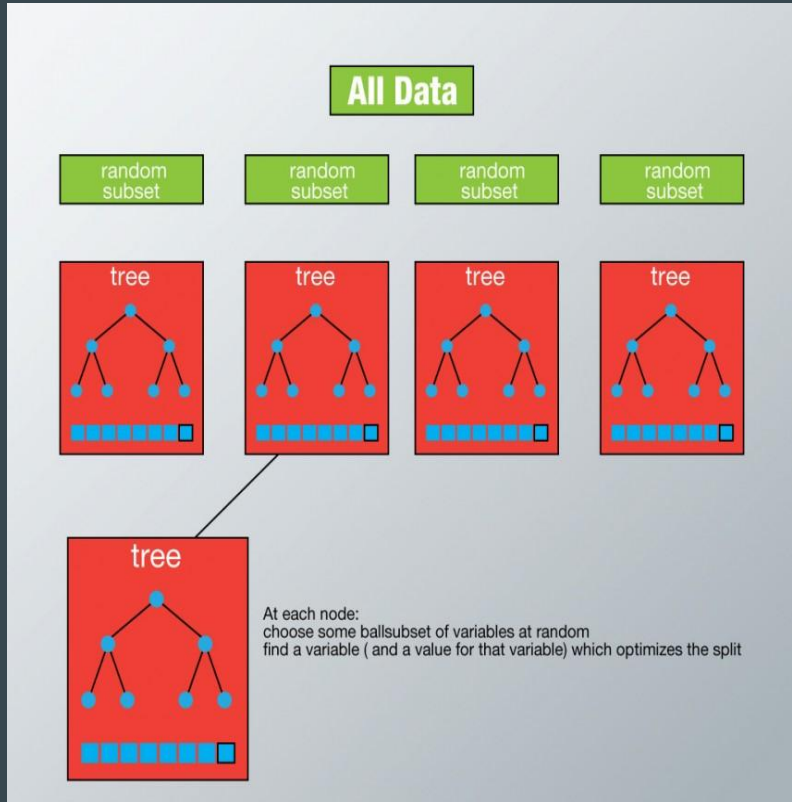
1. Neighborhood of pickup location (categorical; 18 neighborhoods in Manhattan below West 110th Street and East 95th Street)
2. Pickup time (numerical; grouped hourly into 24 categories)
3. Weather (categorical; 3 categories: sunny, rain, or snow)
4. Business day or weekend/federal holiday (categorical; binary)
5. Temperature in Fahrenheit (numerical; with 10F interval i.e. 10, 20, ..., 90)

ML Algorithms: Random Forest

Given several categorical features , we use ***Random Forest***, a “panacea” for data scientists!

Algorithm Pros:

1. Can handle both categorical & numerical features
2. Can rank importance of features
3. Non-parametric → no assumptions on raw data
4. Remains accuracy when a lot of data missing
5. Well-handles unbalanced data & non-linearity
6. As a decision-tree algorithm, easy to interpret



Implementing & Tuning RF in PySpark

PySpark MLlib → implement Random Forest regression in *MapReduce*

```
model = RandomForest.trainRegressor(trainingData, categoricalFeaturesInfo=cat_var,
                                    numTrees=n_tree, featureSubsetStrategy=mode_feature_strat,
                                    impurity='variance', maxDepth=max_deep, maxBins=max_bin)
```

Tune model parameters (number of trees, max tree depth, split strategy, etc.) with extensive experiments on year-long dataset (2015Jul - 2016Jun) on Jupyter Notebook

jupyter randomForest Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help

Code CellToolbar

```
                    performance.append(testRMSE)


best_score = min(performance)
print('minimum test error is ' + str(best_score))
index = performance.index(min(performance))
opt_param = param[index]
print('best model parameter is ' + str(opt_param))
```

```
1
Test Root Mean Squared Error on final_tip_all.txt = 0.25365415617
2
Test Root Mean Squared Error on final_tip_all.txt = 0.252700698118
3
Test Root Mean Squared Error on final_tip_all.txt = 0.254118123889
4
```

Model Test Performance

Metrics:

Root-mean-square deviation (RMSD) on test data (20% randomly sampled)

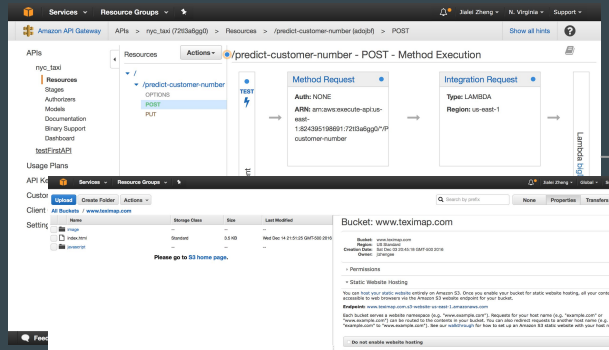
Best Model so far:

500 trees, 10 as max tree depth, 64 as maximum number of bins

	Average Fare	Average Tip	Pickup Count
RMSE on test	$1.26	$0.25	630

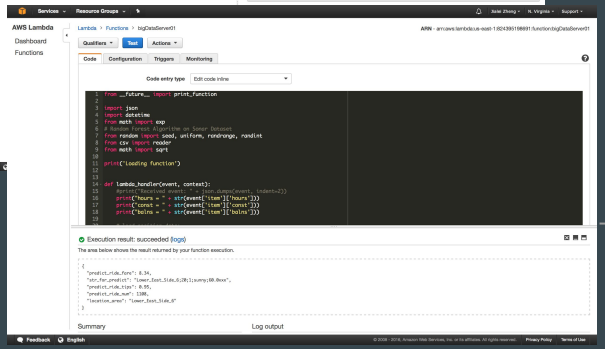
Web Application

Use AWS S3, API Gateway and AWS Lambda Function to build a Web App.

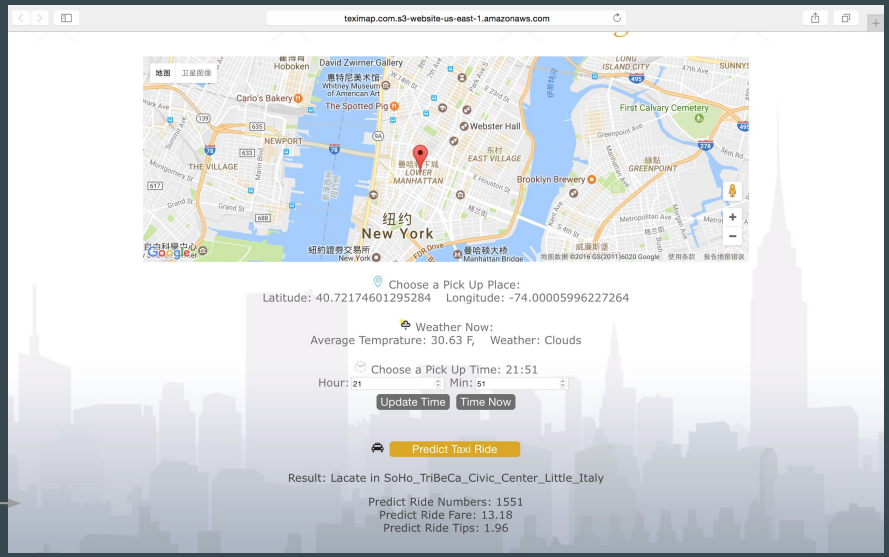


URL Mapping

Front End



Back End



http://www.teximap.com.s3-website-us-east-1.amazonaws.com/

Thank You! Q&A