DS-GA 3001 Final Project Presentation

NYC Uber Pickup Analysis and Prediction

Group11
Chuan Chen
Yafu Ruan
Shuwen Shen
Yihang Zhang

Introduction

- The demand for ride-sharing services in NYC can vary in different areas, time periods, or under different weather conditions.
- In this project, we aimed to analyze and predict the demand (measured and represented by the number of pickups) and to analyze the relationships between the number of pickups and variables including weather, location, and several time factors.
- We try to produce results that could provide insights to help ride-sharing companies like Uber improve the user experience through optimizations in planning.

Data Overview

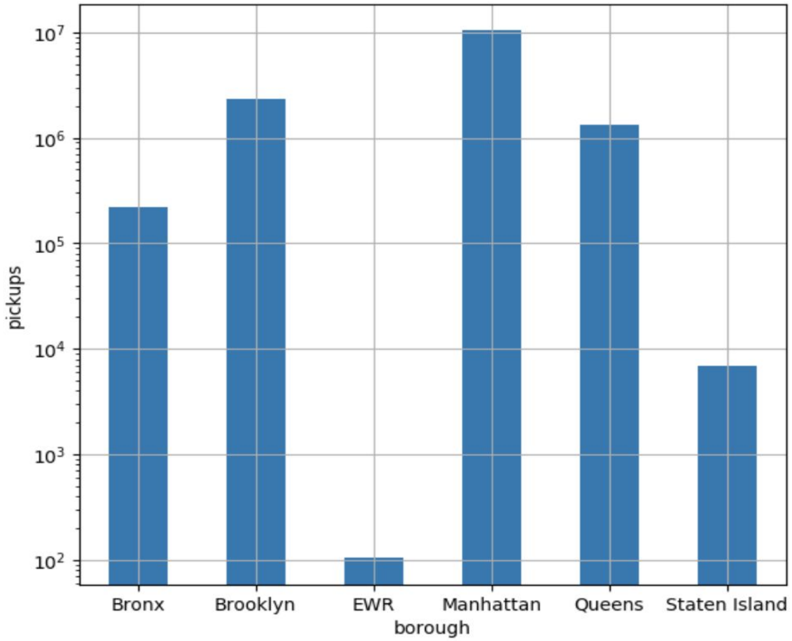
- **Dataset**
  - Uber_nyc_enriched.csv
  - “NYC Uber Pickups with Weather and Holidays” comes from Kaggle [1].
- **Dataset Information**
  - Uber Pickups in New York City, from 01/01/2015 to 30/06/2015
  - Weather data from National Centers for Environmental Information
  - LocationID to Borough mapping by FiveThirtyEight
  - NYC public holidays

Data Overview
(Cont’d)

- The dataset contains 29,101 instances, each of which has the following 13 variables:
  - pickup_dt: Time period of the observations.
  - borough: NYC’s borough.
  - pickups: Number of pickups for the period.
  - spd: Wind speed in miles/hour.
  - vsb: Visibility in Miles to nearest tenth.
  - temp: temperature in Fahrenheit.
  - dewp: Dew point in Fahrenheit.
  - slp: Sea level pressure.
  - pcp01: 1-hour liquid precipitation.
  - pcp06: 6-hour liquid precipitation.
  - pcp24: 24-hour liquid precipitation.
  - sd: Snow depth in inches.
  - hday: Being a holiday (Y) or not (N).

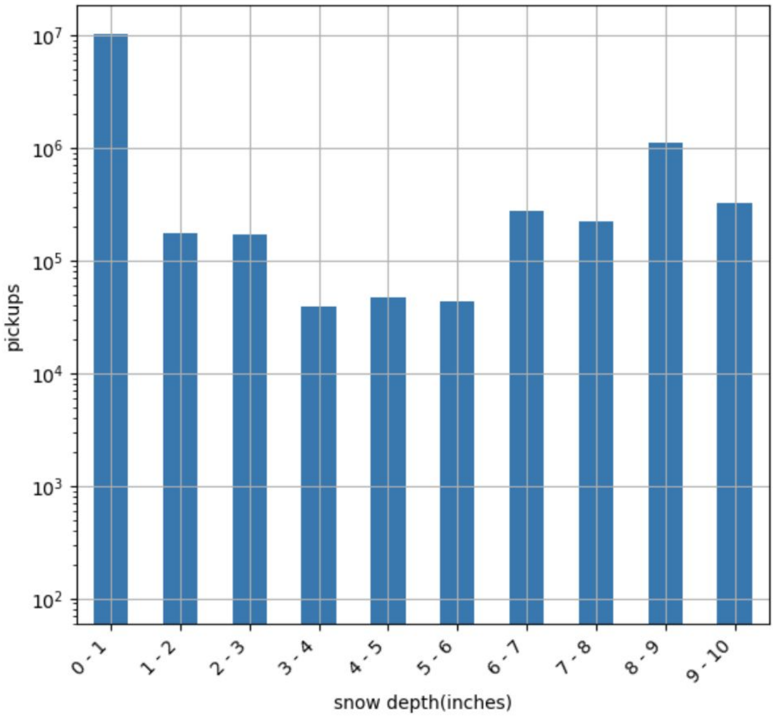
Data Overview
(Cont'd)

- Exploratory Data Analysis
  - borough



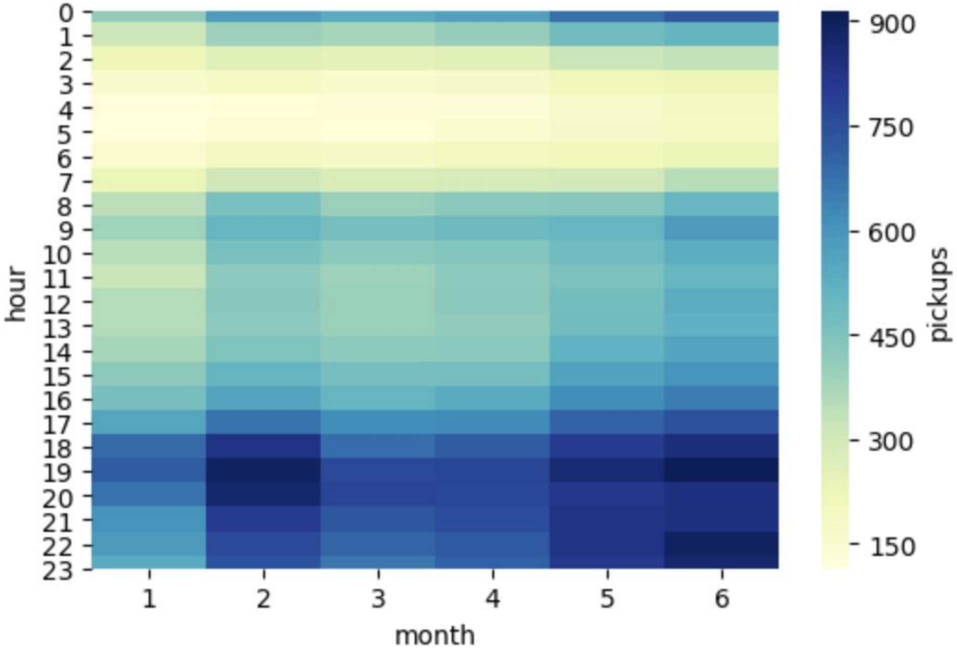
Data Overview
(Cont'd)

- Exploratory Data Analysis
  - snow depth



Data Overview
(Cont'd)

- Exploratory Data Analysis
  - heatmap



Linear Regression

- Deleting Missing Values
- Data Normalization
- One-hot Encoding on Borough
  - Queens 1
  - Bronx 0
  - Manhattan 0
- Reformat Date features
  - Day
  - Month
  - Hour range

Linear Regression

- R2: 0.701
- RMSE: 0.0833

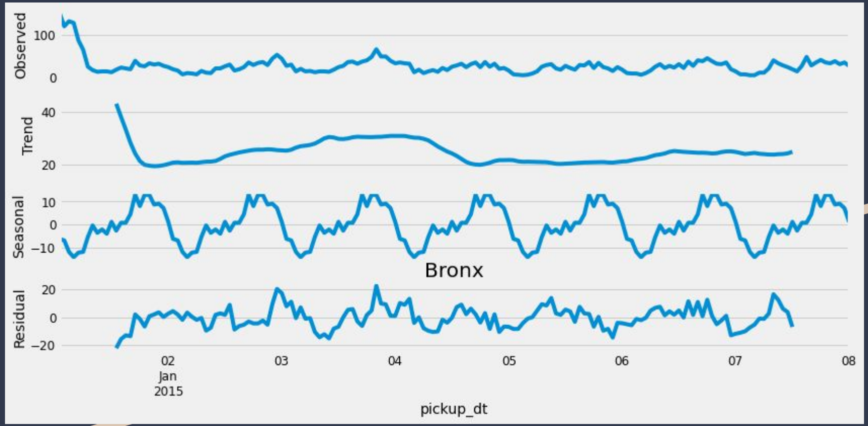
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0433	0.002	22.887	0.000	0.040	0.047
spd	0.0248	0.004	6.886	0.000	0.018	0.032
temp	0.0327	0.003	10.309	0.000	0.026	0.039
pcp06	0.0215	0.009	2.326	0.020	0.003	0.040
pcp24	-0.0477	0.006	-7.387	0.000	-0.060	-0.035
Bronx	-0.0585	0.001	-40.702	0.000	-0.061	-0.056
Brooklyn	0.0040	0.001	2.763	0.006	0.001	0.007
EWR	-0.0646	0.001	-45.124	0.000	-0.067	-0.062
Manhattan	0.2522	0.001	175.853	0.000	0.249	0.255
Queens	-0.0252	0.001	-17.587	0.000	-0.028	-0.022
Staten Island	-0.0645	0.001	-44.890	0.000	-0.067	-0.062

Net Elastic

- Grid Searching with MPI
- R^2: 0.7246, RMSE: 0.0836

```
{'spd': 0.0022735242086995l,
 'vsb': 0.0,
 'temp': -0.0,
 'dewp': -0.008916970789O9031,
 'slp': -0.0,
 'pcp01': 0.0,
 'pcp06': 0.0,
 'pcp24': -0.023964404904429866,
 'sd': 0.008425801795421935,
 'hday': -0.0,
 'Bronx': -0.0325857506162292 9,
 'Brooklyn': 0.028111867915252612,
 'EWR': -0.03887475740313736,
 'Manhattan': 0.27480260095218745,
 'Queens': -0.0,
 'Staten Island': -0.03848110202495794,
 'day': 0.00039186413757 44425,
 'month': 0.005109977583957104,
 'range': 0.012231658456 29436}
```

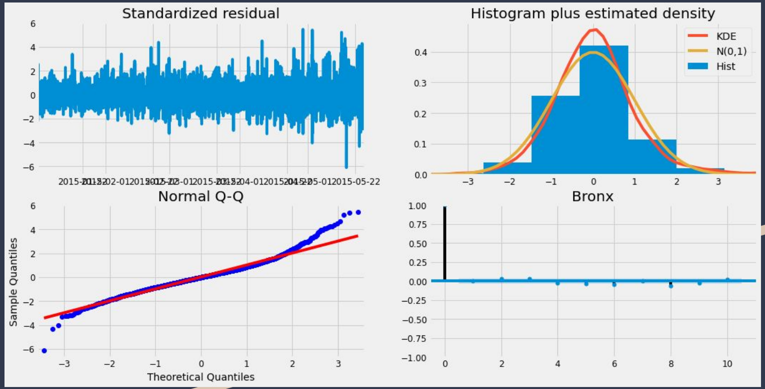
ARIMA model

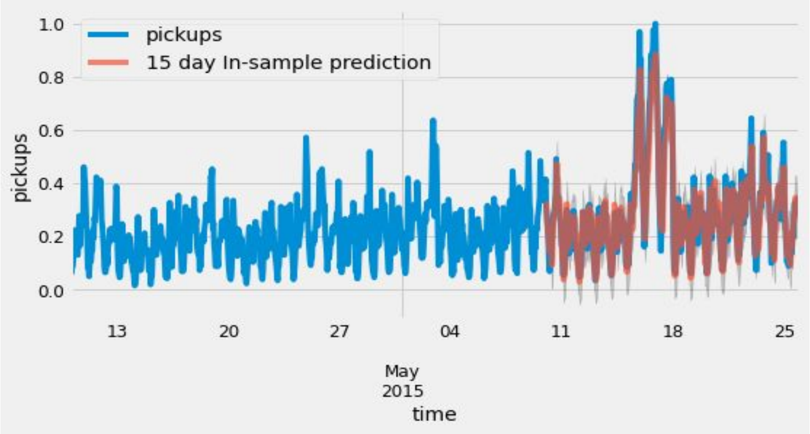


- Data preparation
- Grid_search
  - 64 different parameter combinations
  - normalized data : negative AIC

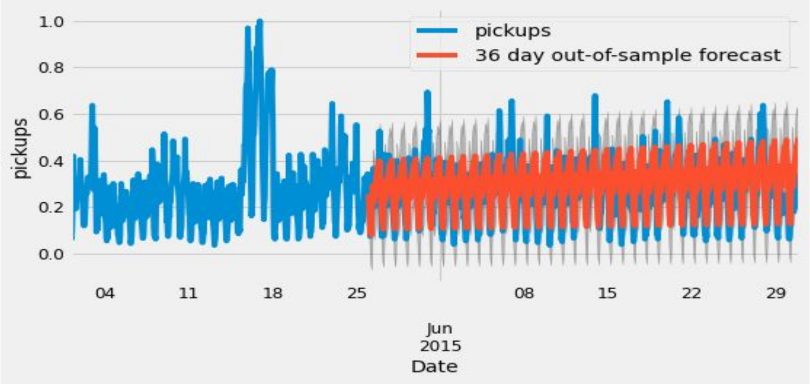
```
the best parameters are: ARIMA(1, 0, 1)x(1, 0, 1, 24) - AIC:-11749.16332826469
the best parameters are: ARIMA(1, 0, 1)x(1, 1, 1, 24) - AIC:-13940.69232958924
the best parameters are: ARIMA(1, 0, 1)x(0, 0, 0, 24) - AIC:-8304.604605601418
the best parameters are: ARIMA(1, 0, 1)x(1, 1, 1, 24) - AIC:-13171.74264575796
the best parameters are: ARIMA(1, 0, 1)x(1, 0, 1, 24) - AIC:-10321.197330620835
the best parameters are: ARIMA(1, 0, 1)x(1, 0, 1, 24) - AIC:-5859.308267172053
total time spent on grid search: 3452.0252158641815
```

ARIMA prediction and forecast





RMSE: 0.06



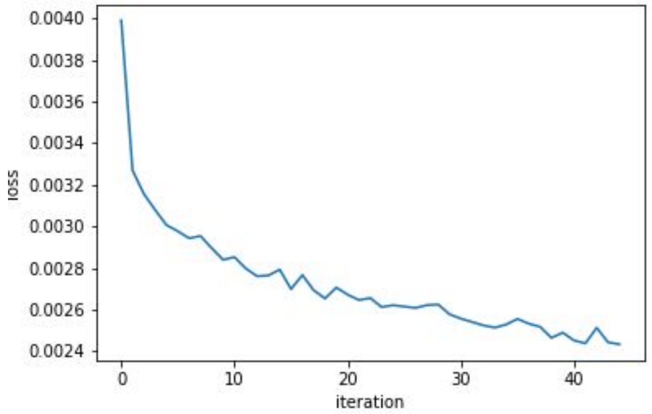
RMSE: 0.1

Neural Network

Find nonlinear relationship between data.

Use early stopping to avoid overfitting.

Reach RMSE 0.0703.



Optimization

- line_profiler

In this project we use this technique to find out how much time is spent on each function.

After that, we are able to know where we need to improve.

Optimization

- Itertools

It is faster than operations on list since it reduce the swag between data.

13.46% performance increase

```
activation = ['tanh','relu']
solver = ['lbfgs','sgd','adam']
alpha=[0.001,0.01,0.1,1]
learning_rate = [0.001,0.0001]
hidden = [(100,50,25),(200,100,100,50),(100,75,50,50)]
para = list(itertools.product(activation,solver,alpha,learning_rat
```

Optimization

- Parallel Computing

Message passing interface(MPI)

It is used in parameter tuning since this problem is easy to parallelize.

It saves 36% of time.

MPI	For-loop
258s	408s

Optimization

- Python Concurrency
  - multiprocessor
  - 4 chunks
  - Parallel parameter tuning

improved efficiency: 21.22%

Conclusion

- NN is the best performing model
- Improved 27.35% overal
- Introduce more features
- Possible future models: RNN, LSTM