CZ4041: Machine Learning

Project: NYC taxi trip duration prediction (Kaggle)

Presented by Group 21

Meet the Team

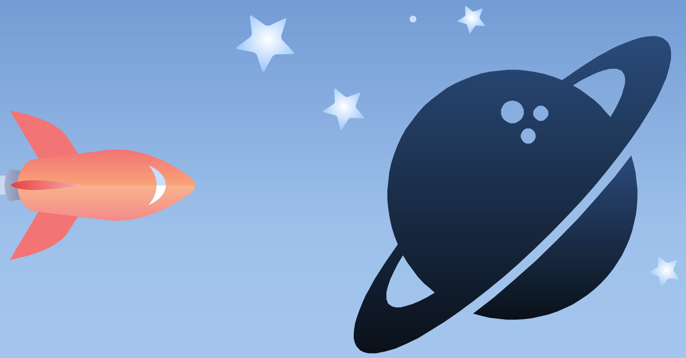
Chang Heen Sunn (Ryan)

Hasan Mohammad Yusuf

Ng Man Chun (Jay)

Padhi Abhinandan

Wong Zhen Yan



Build a model that predicts the ride durations of Taxi Trips in New York City

NYC Taxi Trip Duration Dataset

id

vendor_id - taxi company

pickup_datetime, dropoff_datetime

passenger_count

pickup long/lat, dropoff long/lat

store_and_fwd_flag - trip record submitted manually?

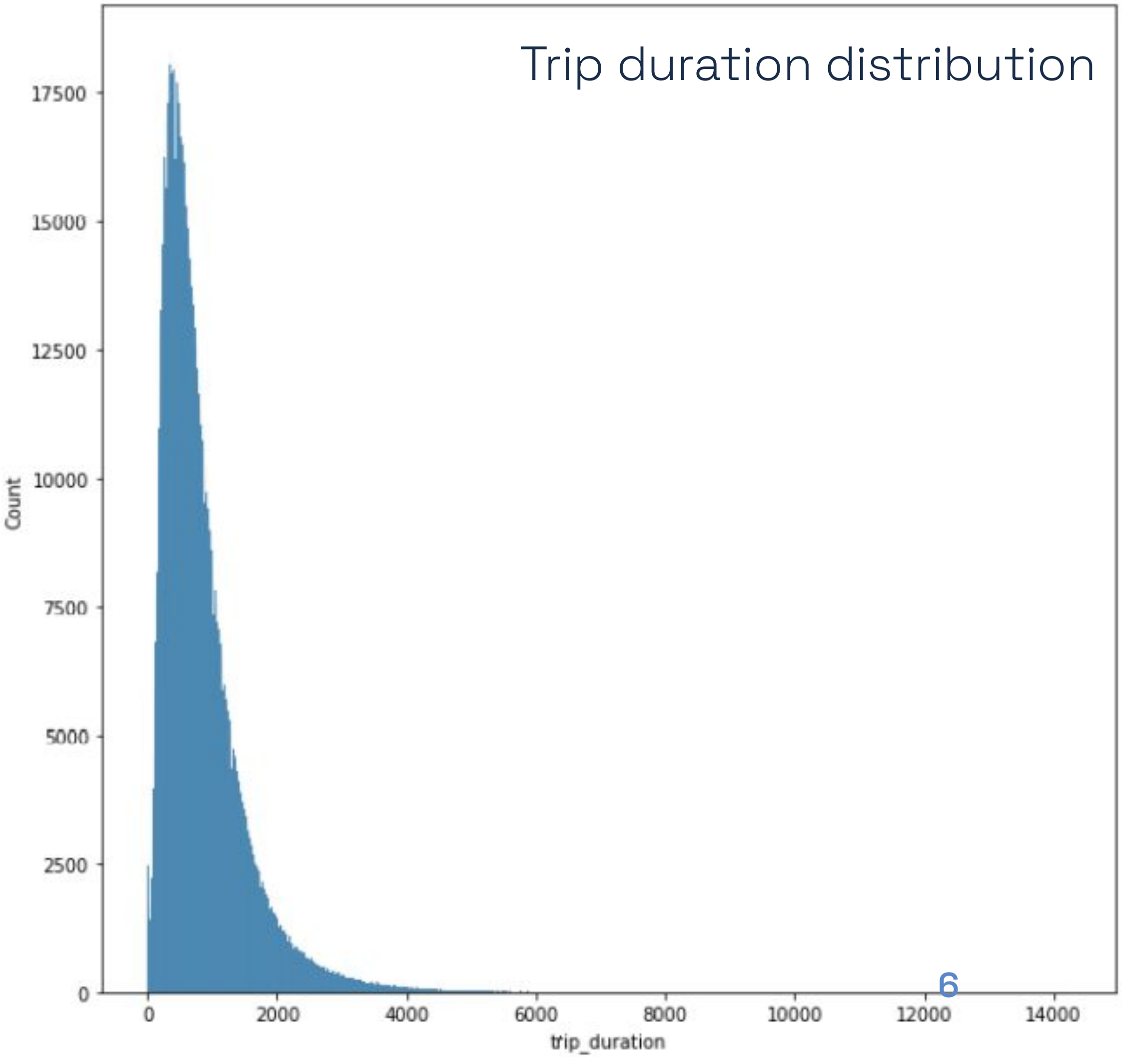
trip_duration - duration of the trip in seconds

NYC Taxi with OSRM

starting_street
end_street
total_distance
total_travel_time
number_of_steps - step consists of some driving and a turn or going on to a highway
street_for_each_step - a list of streets where each step occurs
distance_per_step
travel_time_per_step
step_maneuvers - e.g. taking turn
step_direction
step_location_list - the coordinates for each action

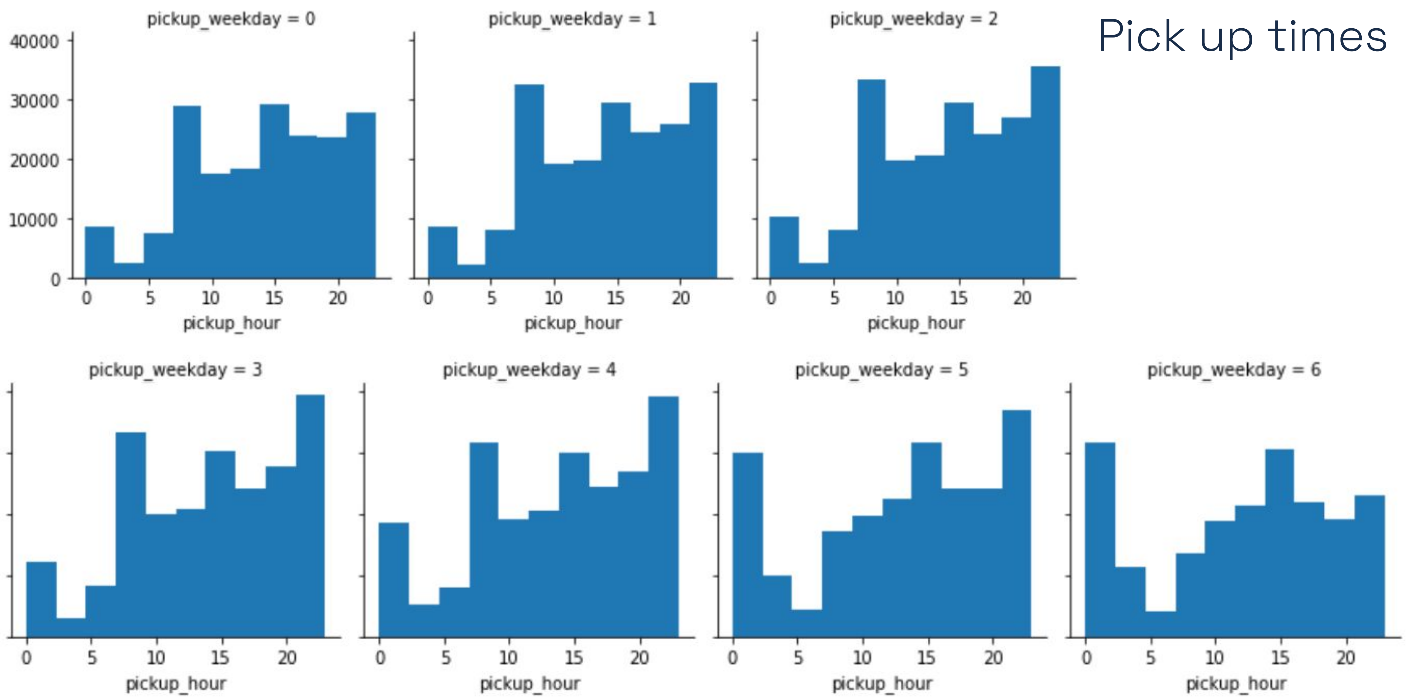
1. Exploratory Data Analysis

Trip duration distribution

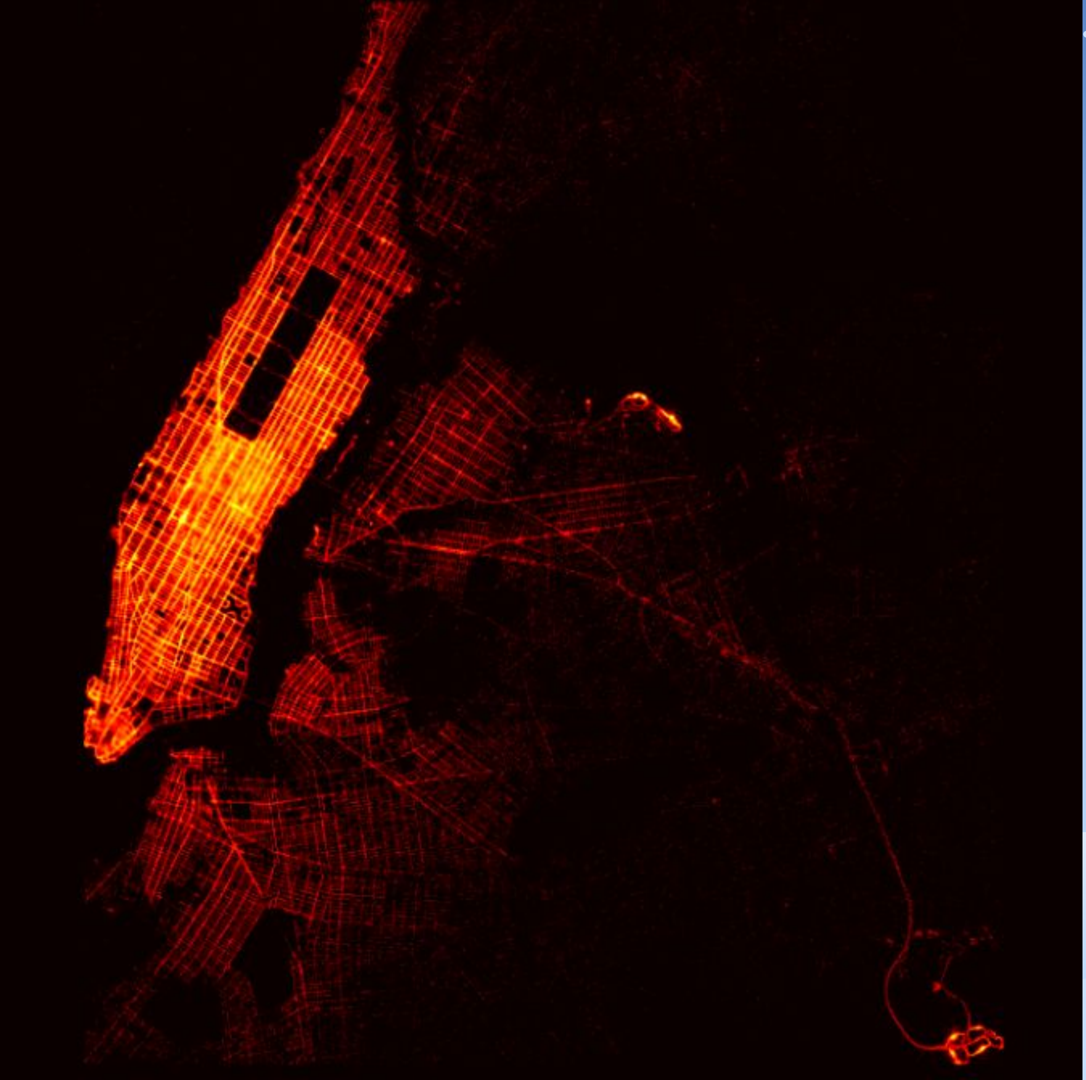


- Very few outliers
- Most <3600 seconds

Pick up times

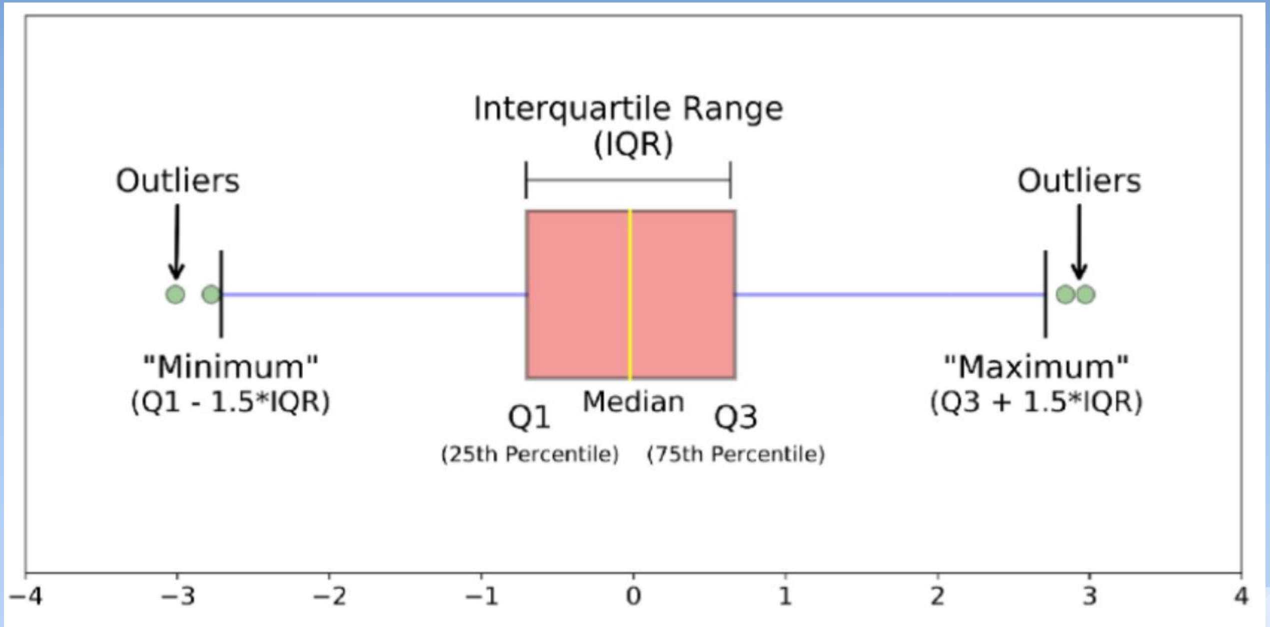


- Peak going to work, lunch, and dinner hours during weekdays
- Peak after midnight during weekends
- Very consistent!



Trip start/endpoint heatmap

- Streets clearly visible
- Mostly around center of Manhattan
- None in Central Park



Outlier elimination

Feature Engineering & Selection

Date time features extracted from raw timestamp

PCA on longitude/latitude

Manhattan-Haversine distance of trip

OSRM - calculate theoretical fastest (shortest) path

2. Extreme Gradient Boosting (XGBoost)

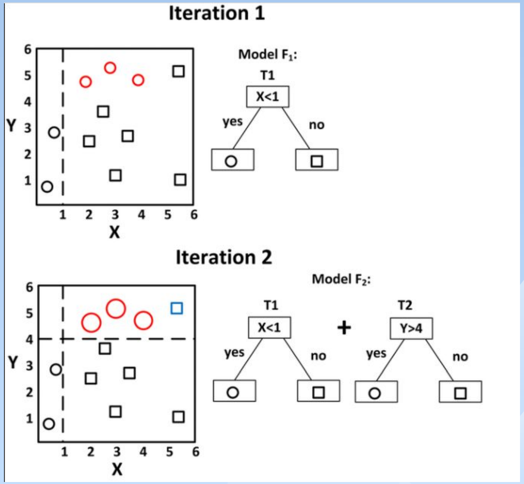
Gradient Boosting

Gradient Descent

$$w = w - \eta \nabla w$$

$$\nabla w = \frac{\partial L}{\partial w} \quad \textit{where } L \textit{ is loss}$$

Ensemble of weak learners -> strong learner



One new decision tree at a time

Address shortcomings of other weak learners

Techniques

Backwards **tree pruning** for each weak learner

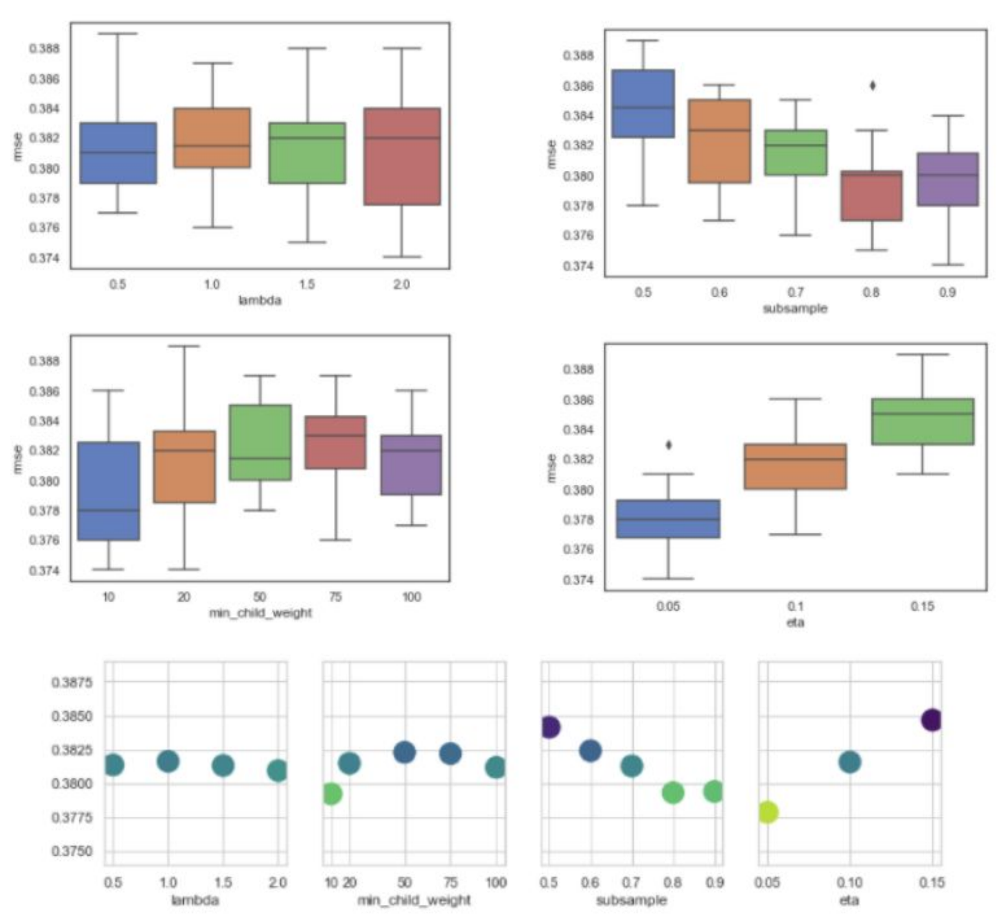
Cross-validation to tune hyperparameters

Parallelisation and cached gradient vectors

Objective Function

RMSE loss + regularization

Hyperparam Optimisation



Final results XGB

Training RMSE: 0.13582

Validation (k-fold): 0.36754

Test (Kaggle): 0.37299

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission_xgb.csv	just now	1 seconds	3 seconds	0.37299

Complete

Jump to your position on the leaderboard

Submission and Description	Private Score	Public Score
submission_xgb.csv a minute ago by lionchang test 10 Apr	0.37039	0.37299

3. Neural Networks

Neural Network Overview & Architecture

- Neural networks are a subset of machine learning and form the basis of deep learning algorithms.
- A feed forward network architecture was created using TensorFlow (Keras) for this project.
- All hidden layers in the network are Dense (Fully Connected).
- Batch normalization (BN) layers are introduced to improve the training stability, allowing for quicker convergence.
- Training data is split into batches, and for each batch of inputs to each BN layer, inputs are normalized with respect to their batch in both batch size and input size dimensions.

Hyperparameters Tuned

- Number of Dense (FC) Layers - Network Depth
  - Between 4 and 64 layers
- Number of Neurons per Dense Layer - Network Breadth:
  - Between 16 and 512 neurons per layer
- Optimizers for Gradient Descent
  - Stochastic Gradient Descent (SGD), Momentum, RMSProp, and Adam.
- Learning Rate (α)
  - Tested α values from the range: 10^-2 < α < 10^-4
- Regularization
  - None vs. L2 Kernel/Bias vs. Dropout Regularization

Neural Network Experiments

- Random Search used for Hyperparameter tuning.
- Model was overfitting quickly so training was usually stopped at 100 epochs.
- Experimented with different batch sizes initially [16, 32, 64, 128, 256], but final batch size was 128.
    - Trade-off between no. of gradient descent updates per epoch & training time per epoch (larger batches => less time).
    - Dataset is large (>1.4 million training examples).
- Used a Train-Validation split of 80:20.

Important Results

Hyperparameters	Train RMSE	Validation RMSE	Leaderboard Score (Public)
16 layers, 256 neurons per layer, SGD, no regularization, learning rate = 1e-2	0.43314	0.43201	0.47360
16 layers, 256 neurons per layer, Adam, no regularization, learning rate = 1e-3	0.42165	0.42784	0.46209
16 layers, 256 neurons per layer, Adam, L2 reg. factor = 1e-5, learning rate = 1e-3	0.41398	0.41239	0.43442
16 layers, 256 neurons per layer, Adam, Dropout w/drop-rate = 0.2, learning rate = 1e-3	0.41887	0.43542	0.45930
32 layers, 256 neurons per layer, Adam, L2 reg. factor = 1e-5, learning rate = 1e-3	0.40480	0.39246	0.41522
40 layers, 400 neurons per layer, Adam, L2 reg. factor = 5e-5, learning rate = 1e-3	**0.39921**	**0.38803**	**0.39963**

Table of Results for Neural Networks with Various Combinations of Hyperparameters

- Table shows main results.
- Best (final) model highlighted in green.
- Best optimizer: Adam (vs. SGD, RMSProp, etc.).
- Error decreased with increasing network depth and breadth.
- Learning rate adjusted based on Adam (α = 1e-3).
- L2 regularization helped improve ability of NN to generalize to new data.

Results of Final Neural Network Model

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission_neural_network.csv	a minute ago	1 seconds	3 seconds	0.39963

Complete

Jump to your position on the leaderboard

Submission and Description	Private Score	Public Score
submission_neural_network.csv a minute ago by Abhinandan Padhi add submission details	0.39758	0.39963

Analysis of Experimental Results

- Clearly, XGBoost performed better than Neural Networks for predicting NYC Trip Duration. Why?
- Possible reasons:
  - XGBoost has the big advantage of Ensemble learning, NN does not. Ensemble Learners are especially popular for NYC dataset.
  - Further hyperparameter tuning could have improved NN performance, but is difficult due to computational and time constraints.
- While XGBoost allows for Top 6% Kaggle LB rank, higher performance might have been possible with Weather data or more external data.
- Some taxis/drivers may be faster or “more aggressive” than others, so predicting trip duration would be complicated further.

4. Conclusions

Summary of The Project

- We have explored various types of Exploratory Data Analysis (EDA) techniques to better understand trends/patterns in the NYC data.
- Applied outlier detection using IQR and Z-score approach.
- Performed feature engineering and selection on our dataset.
- For prediction, we experimented with two types of models: XGBoost and Neural Networks.
- Experiments and analysis shows that XGBoost outperformed the Neural Network models.
- Thus, XGBoost is a great solution to the NYC Taxi Trip Duration prediction problem.

Challenges Faced

1. Data was noisy and contained both outright errors and non-standard behaviour.
2. Trip duration was affected by various factors, e.g. time of the day, speed, and trip distance.
3. NYC data had relatively few features per training example, so external data was required.
4. Training Machine/Deep Learning models was computationally expensive and time-consuming.
5. Division of work was difficult due to dependencies between different aspects of our project.

Our Solutions to the Challenges

1. Discovered ways to filter bad data to improve prediction accuracy.
2. Explored factors and created new features to account for these factors.
3. Used NYC-OSRM dataset from Kaggle, though processing and merging with the original data was challenging.
4. Had to use cloud services for computation, e.g. Google Colaboratory and Kaggle GPUs.
5. We attempted to collaborate on the code by sharing code on a group GitHub repository and using Google Colaboratory.

Possible Future Improvements

1. Different methods for outlier removal and data cleaning.
2. Engineer more (and better) features and select features more carefully.
3. Experiment with other complex models to see if performance is better.

Thanks for watching