# COMP9417 Assignment Report

Topic 1.1: Machine learning and data mining competitions

Yousheng Sui

Chang Ge

Shichao Zhang

# Contents

## Introduction

Taxi usage information such as trip duration and trip cost become a popular issue that people consider these years. New York, as one of the most famous cities in the world, attracts lots of attention every day. A considerable amount of companies set their main sites there. In the meantime, people come to New York to spend their holidays. Taking all these aspects into account, taxi trip duration prediction tends to be useful and valuable. A competition published by Kaggle called New York City Taxi Trip Duration provides a platform for people to predict total trip duration in New York City.

We choose to complete this competition as our 9417 assignment. In this report, we are going to demonstrate and discuss the steps and technologies that we used to complete this competition. In general, we are given some training datasets and test datasets contain some features like pickup time and pickup location. We are asked to predict the time duration according to the datasets.

In this assignment, we first explore the data, look for additional datasets which can support the prediction. Secondly, we apply the feature engineering technique to clean and select features. For training model, we choose to use XGBoost as our training framework. Finally, as an additional feature, we employ Bayesian Optimization which is an automated parameter tuning method to tune our parameters.

# Implementation

## Data Exploration and Visualization

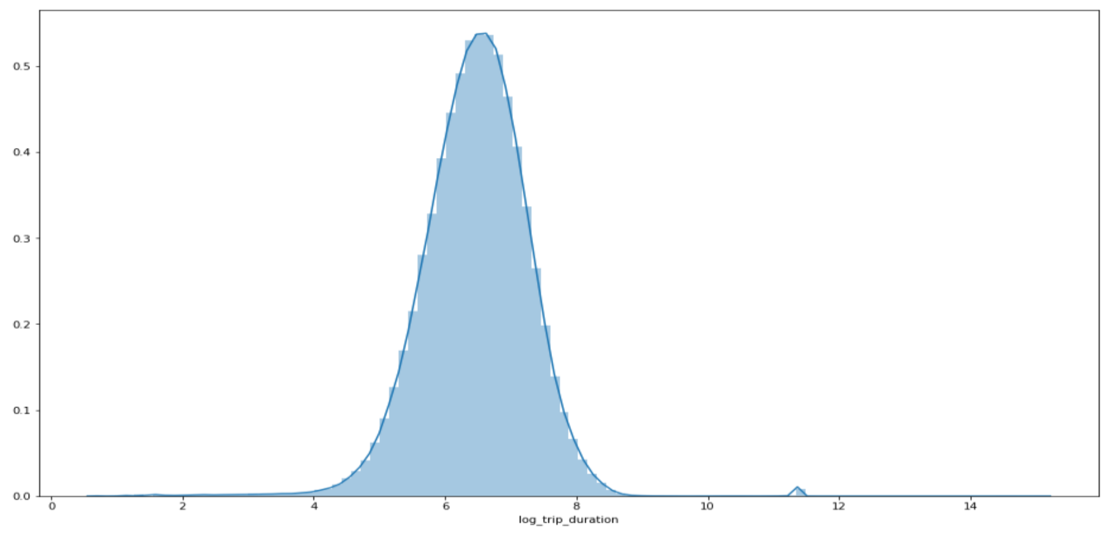
To predict the trip duration, we need to find the most influential features. According to this policy, we discover these three main aspects after discussion: time, location and weather. These three aspects may play a decisive role in this competition. Then all the manipulations we do to our data are based on it.

In the beginning, we found that there are 11 features in the original training dataset, as shown below.

```
id                     1458644 non-null object
vendor_id              1458644 non-null int64
pickup_datetime        1458644 non-null object
dropoff_datetime       1458644 non-null object
passenger_count        1458644 non-null int64
pickup_longitude       1458644 non-null float64
pickup_latitude        1458644 non-null float64
dropoff_longitude      1458644 non-null float64
dropoff_latitude       1458644 non-null float64
store_and_fwd_flag     1458644 non-null object
trip_duration          1458644 non-null int64
dtypes: float64(4), int64(3), object(4)
```

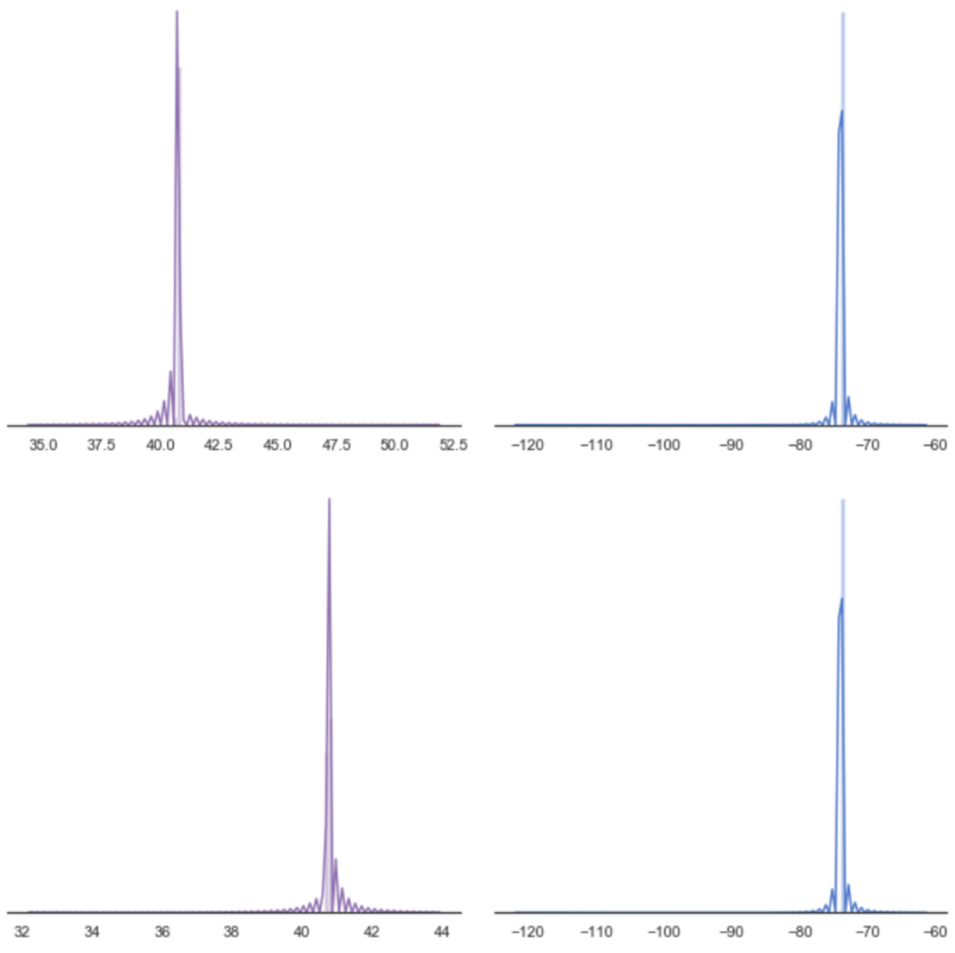
During data exploration, we find that some publicly available resources use additional datasets to assist the prediction. We notice that using additional datasets to train the model is highly encouraged in the problem description as well. Thus, to give as much support as we can to our model, we use additional datasets as following. OSRM datasets, the weather during the period and the holiday dataset are used. We use three features in OSRM dataset which are total_distance, total_travel_time and number_of_steps since these features could be a useful supplementary to original data. The reason why we use weather and holiday data is that people tend to take taxis when the weather condition is bad or during their holidays in common sense. In other words, traffic quality may become worse under these circumstances. Thus, these features may affect trip durations.

Our goal is to predict the trip duration, so the first thing is to have a sense of its' distribution. Since the competition uses RMSLE as the evaluation metric, it's better to convert trip duration to logarithmic form.



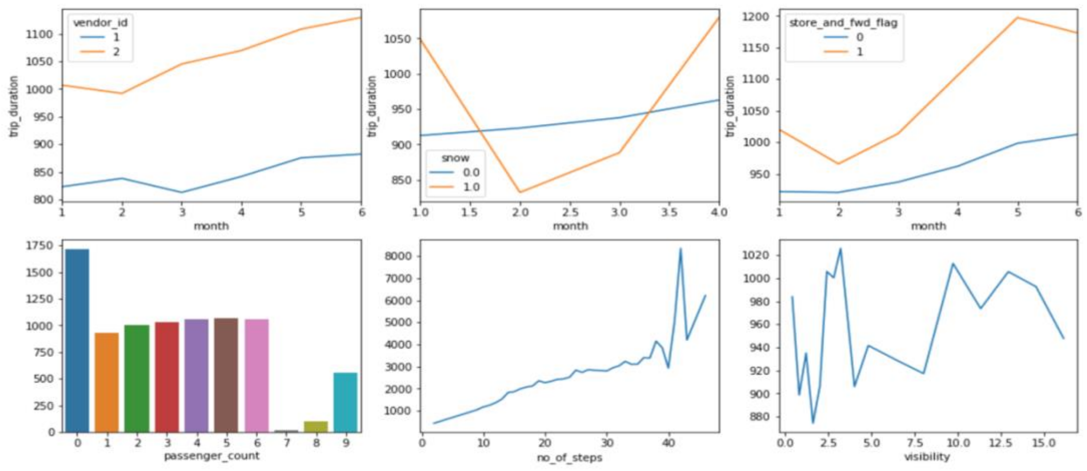
As we can see from the figure above, most of the trip duration follows the normal distribution. However, there are some outliers that need to be considered while doing feature engineering.

Same situation happens in location too.



As we can see from the figure above, 4 location features (pickup_latitude, pickup_longtitude, dropoff_latitude, dropoff_longtitude) also have a lot of outliers need to be done.

As for other features, we use some plots to see if they vary through time. Some figures are listed here:

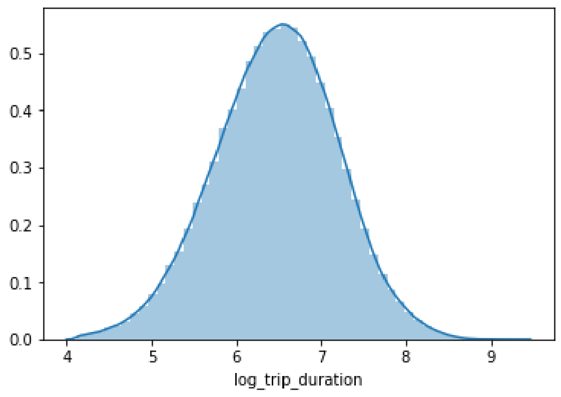


Briefly, the figures show us that features like vendor_id, snow, store_and_fwd_flag, passenger_count, no_of_steps and visibility play an influential role in trip duration. For example, taxis of vendor 1 can send passengers to their destination faster than those of vendor 2. By plotting these features, we can find if we need to keep the feature easily.

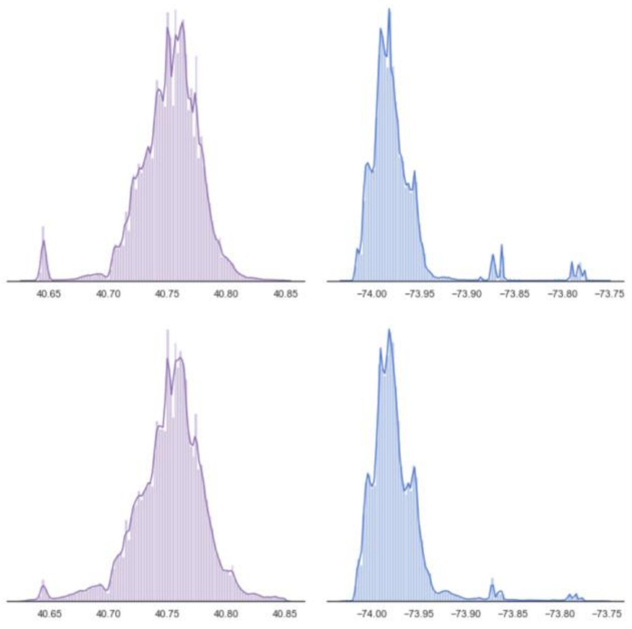
## Feature Engineering

### 1. Outlier processing

As we can see in Data Exploration and Visualization part, we need to extract valid trip duration. In the beginning, we processed the data by using empirical rule, however, we found that trip duration time" 1s" is still in our data set, which is not make any sense since no one would take a 1s trip. so, we manually set the lowest trip duration time to 60s.



Since we are processing the data in New York, a natural idea is to limit the coordinates of all locations within the limits of New York City.



In contrast to the figure in the previous section, we can see that the outliers are basically processed completely.

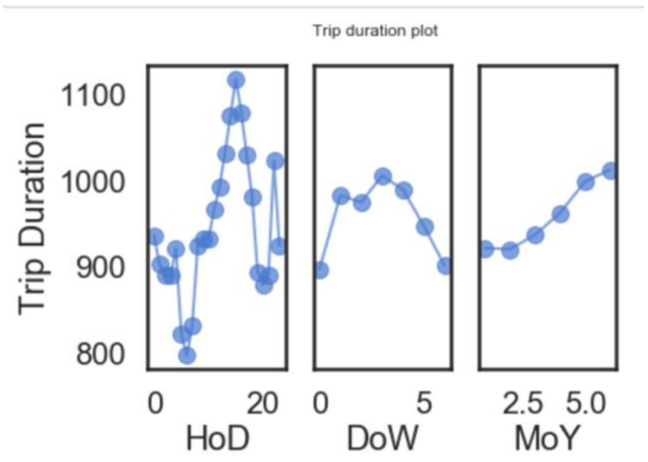
## 2. Time feature processing.

As we said, time is an important feature to trip duration. Here is our thought about what we can do to time.

Holidays could affect the trip duration, since there will be more people taking taxis for travel.

Different weekday could affect the trip duration, since there will be different traffic situation on weekdays and weekends.

Different time in a day could affect the trip duration, since it is clear that morning peak and evening peak could have more traffic jam.



As we can see from this figure, different hour of day, day of week and month of year have big difference at trip duration time.

Follow these thoughts, we:

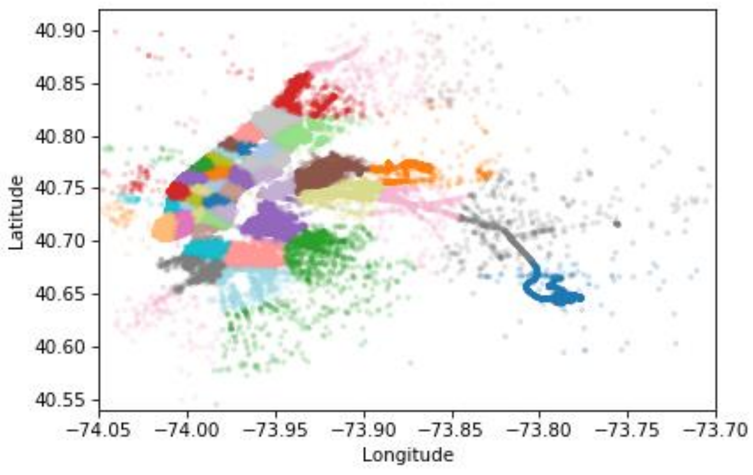
Create 2 features called is_weekend (for weekend and holidays) and is_rest(the rest of them).

Split pickup_time to dayofweek and time (e.g. 17.5 represents 17:30).

## 3. Location feature processing

We believe, as in real life, the trip duration of two trips, which have the same pick-up location and drop-off location, would be similar. Thus, we try to divide the bundle of pick-up and drop-off location into different districts. In this case, it might be hard to divide those locations into real administrative areas and points being clustered looks like buildings located in the same administrative areas. For those reasons, we use K-means to cluster the data points into their own neighborhoods.

we combine the pick_up and drop_off location as a unique identifier and use K-means method to divide New York to 100 clusters. A1fter this process, we graph the pickup location clusters.



Since the clusters are not continuous data, we handle those data via one-hot encoding.

## 4. Weather feature processing

Weather feature is quite important, for example, on snowy and rainy days, the driver must drive at a slower speed because the road is slippery, so this will increase the duration time.

We add 4 new features called Temp (temperature), Precip (precipitation), snow and visibility to our training dataset.

Weather data have many missing values; however, we will use the XGBoost model, which is a tree model, which is less sensitive to missing values and can be used most of the time when data is missing.

## 5. Final features

| | |
|---|---|
| passenger_count | Precip |
| distance_haversine | snow |
| distance_dummy_manhattan | Visibility |
| direction | is_weekend |
| total_distance | is_rest |
| total_travel_time | pick_* |

| | |
|---|---|
| number_of_steps | drop_* |
| dayofweek | vi_* |
| time | sf_* |
| Temp | |

## Model Training

The XGBoost library[1] implements the gradient boosting decision tree algorithm. This algorithm also has other names such as gradient boosting and additive regression trees. Boosting is a bundle of technique where new models are added to correct the errors made by old models until there is no further improvement. Generally, it's a high-efficiency implementation of gradient boosting due to parallelization of tree construction using all CPU cores during training.

In the process of model training, the waiting time before train finishing is very long when we train the model on our laptops. For this reason, we use Google's AI Platform to speed up model training. This means that Google cloud platform offers users creating Jupyter Notebook for running their code and store their data set in Cloud Datastore which allow users to store and access in different regions. Because of the high-performance computing platform help, we reduce the waiting time, we could increase the exploration time in Bayesian optimization without increasing the total waiting time and achieve a more suitable model.

Machine learning algorithms highly rely on parameters normally. Parameters control the learning rate or tree depth in learning algorithms. We employed an automated parameter tuning method called Bayesian Optimization (BO) to tune our parameters. To our best knowledge, almost all the resources or methods shared online use random search optimization. Furthermore, some of them use grid search optimization. These two search methods have their own benefits while they have their weakness. One is that the exact values of parameters need to be provided. Knowing the exact values requires experience. However, as first time “kagglers”, we don’t have enough experience in this area. The other one is that these methods are time-consuming and have high time complexity. For example, if there are 4 parameters and each of them has 4 values, the optimal solution requires 4*4*4*4 = 256 times of running to get.

Bayesian optimization[2], outperforms other state-of-art algorithms on its higher performance and ease of use. By using Gaussian Process, the prior result is used to optimize the next evaluation which gives better performance than other algorithms typically. In addition, only the bounds of parameters need to be provided. Even we are not highly experienced with parameter tuning, we could find the bounds that most fit the model.

We use a python package called Bayesian optimization to implement the feature. Since this package only supports finding the max value of a function, we have to set the score to a negative value. By running the function, the package automatically

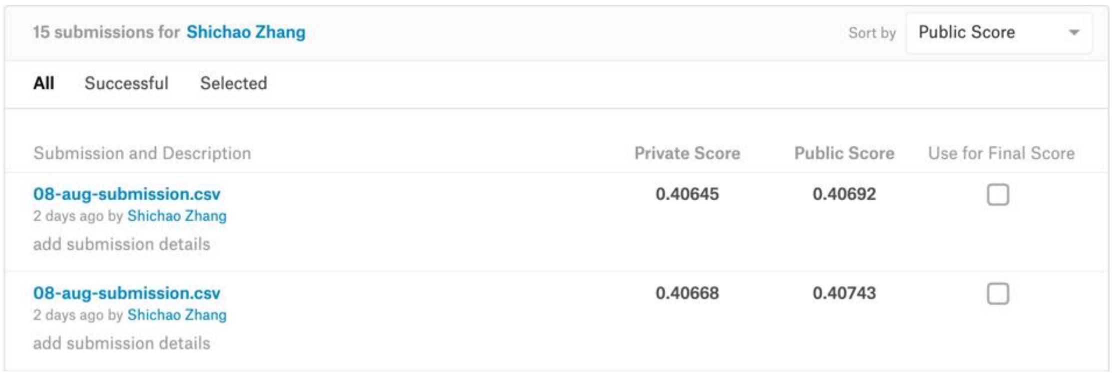
# Results

## Evaluation on Kaggle

The evaluation metric for this competition is Root Mean Squared Logarithmic Error.

$$e = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

The best score we reached is 0.40692 which ranked 523th in 1257 teams.



15 submissions for Shichao Zhang

Sort by Public Score

All Successful Selected

| Submission and Description | Private Score | Public Score | Use for Final Score |
|---|---|---|---|
| 08-aug-submission.csv<br>2 days ago by Shichao Zhang<br>add submission details | 0.40645 | 0.40692 | ☐ |
| 08-aug-submission.csv<br>2 days ago by Shichao Zhang<br>add submission details | 0.40668 | 0.40743 | ☐ |

The final submission file could be downloaded via this URL.

## The best model of parameters

By using Bayesian optimization, we achieve the best results using such parameters in XGBoost.

Max_depth: 15 (maximum depth of a tree)
Objective: reg: squarederror
Eta: 0.15 (Step size shrinkage used in update to prevents overfitting)
Subsample: 0.85 (Subsample ratio of the training instances)
Lambda: 3 (L2 regularization term on weights)
Colsample_bytree: 0.9 (the subsample ratio of columns for each tree)
Colsample_bylevel: 1 (the subsample ratio of columns for each level)
Min_child_weight: 15 (Minimum sum of instance weight needed in a child)

## Running time

In this assignment, we use Compute Engine of Google Cloud Platform for training model. The training time on the virtual machine with 16 vCPU, 60GB RAM and 1

NVIDIA Tesla P100 is 15 minutes. The cost of computing resource is 1.55 USD per hour which means each running included feature engineering, model training and prediction costs 0.55 USD. In our opinion, it’s a worthwhile price.

## Conclusion

In conclusion, we achieved the goal of this competition successfully by doing data exploration and visualization, feature engineering, model training and parameter tuning. Moreover, we used Bayesian optimization as an additional feature which makes our parameter tuning stage effectively and efficiently. We got 523th in 1257 teams by using our final model. In this assignment, we practiced our skills on time organization and teamwork. We applied what we leant in 9417 so far as well. For future work, some other features could be created to improve performance and the parameters could be tuned to get better results.

## References

[1] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016.

[2] Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms." Advances in neural information processing systems. 2012.