

---

# Final Project Report - Kaggle

---

**Samarth Gupta**  
College of IST  
Penn State University  
State-College, PA 16801  
sxg646@psu.edu

**Shivansh Rao**  
College of IST  
Penn State University  
State-College, PA 16801  
shivanshrao@psu.edu

## Abstract

This report presents our methodology which we used for the Kaggle Competition : "New York City Taxi Trip Duration". Given the primary dataset by the NYC Taxi and Limousine Commission, our goal is to build a model that predicts the total ride duration of taxi trips in New York City. We propose a gradient boosting method optimized on fair loss to solve this problem. At the end of all the checkpoints we finish on 2<sup>nd</sup> position in the leader-board with a RMSLE value of 0.37122. Our final model was a stacked version of 30 LightGbm models.

## 1 Introduction

In this competition, Kaggle puts forward the challenge of building a model which could predict the trip duration of the taxi trips that took place in the New York City. The dataset provided by the competition was based on the NYC Cab trip data of the year 2016 and was made available in Big Query on Google Cloud Platform (GCP). This data was a subset of the original NYC Taxi and Limousine Commission (TLC) dataset. The training data consists of 1458644 trip records across the New York City, and the test data has 625134 trip records in total. The task is to predict the trip duration given a set of taxi trip attributes such as pickup location, drop-off location, pickup date and time, passenger count etc.

This is an important task for most of the taxi trip applications and makes it easier to estimate the arrival time at the drop off location, allowing the customer to plan his trip efficiently. An inefficient algorithm may cause issues with the customer's daily schedule which will lead to lower customer base ultimately leading to a reduction in revenue.

## 2 Method

In our approach we use the Lightgbm model to predict the trip duration values. Lightgbm is also a gradient boosting framework which uses the tree based learning algorithm. The major difference it has from the other gradient boosting algorithms like XGBoost is that it grows vertically whereas other algorithms grow horizontally. This means that it performs a depth-first search unlike the other breadth-first approaches and is faster than all the remaining gradient boosting algorithms. Since the amount of data is increasing every day, it has become really difficult for the classical data science algorithms to give faster results. The word 'Light' in Light GBM stands for high speed. It can handle very large quantities of data also takes up a very less memory to execute. In addition Light GBM also uses GPU for learning.

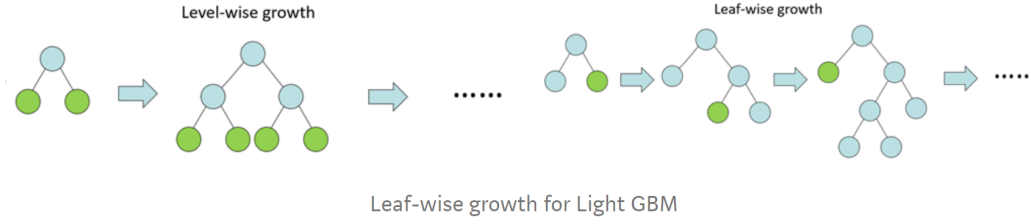


Figure 1: LightGBM architecture

## Data fields

- **id** - a unique identifier for each trip
- **vendor\_id** - a code indicating the provider associated with the trip record
- **pickup\_datetime** - date and time when the meter was engaged
- **dropoff\_datetime** - date and time when the meter was disengaged
- **passenger\_count** - the number of passengers in the vehicle (driver entered value)
- **pickup\_longitude** - the longitude where the meter was engaged
- **pickup\_latitude** - the latitude where the meter was engaged
- **dropoff\_longitude** - the longitude where the meter was disengaged
- **dropoff\_latitude** - the latitude where the meter was disengaged
- **store\_and\_fwd\_flag** - This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server - Y=store and forward; N=not a store and forward trip
- **trip\_duration** - duration of the trip in seconds

Figure 2: Training Data fields

## 2.1 Feature Engineering

In this section, we go through the feature engineering and exploratory data analysis to better understand the dataset and include additional features which may be critical for trip duration prediction. Figure 2 lists the initial features in the dataset.

The steps that we followed in our feature engineering are as follows:

- We first convert the 'pickup\_datetime' by using `pandas.to_datetime`, and then extract 'pickup\_date', 'pickup\_hour', 'pickup\_min', 'pickup\_weekday' as features from 'pickup\_datetime', 'pickup\_dayofyear', 'pickup\_hour\_weekofyear'.
- We then make 'pickup\_week\_hour' by calculating it using 'pickup\_weekday' and 'pickup\_hour'.
- **Bearing angle:** We calculate bearing angle and add it to our feature list.  $\beta = 2 * \alpha * \sin^{-1}(\sqrt{(distance)})$  where  $\beta$  is the bearing direction,  $\alpha$  is the average earth radius which is 6371 kms and distance is equal to  $(\sin(latitude\_difference * 0.5))^2 + \cos(pickup\_latitude) * \cos(dropoff\_latitude) * \sin(longitude\_difference * 0.5)^2$ .
- **Haversine distance:** The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes.  $D_h = 2 * \alpha * \sin^{-1}(\sqrt{(distance)})$  where  $D_h$  is the haversine distance and  $\alpha$  is the average earth radius which is 6371 kms and distance is equal to  $(\sin(latitude\_difference * 0.5))^2 + \cos(pickup\_latitude) * \cos(dropoff\_latitude) * \sin(longitude\_difference * 0.5)^2$ .
- **Manhattan distance:** Manhattan distance is the distance between two points measured along axes at right angles. In a plane with  $p_1$  at  $(x_1, y_1)$  and  $p_2$  at  $(x_2, y_2)$ ,  $D_m = |x_1 - x_2| + |y_1 - y_2|$  where  $D_m$  stand for Manhattan Distance.
- **Center Latitude/Longitude:** It is the average of the pickup and dropoff coordinates.

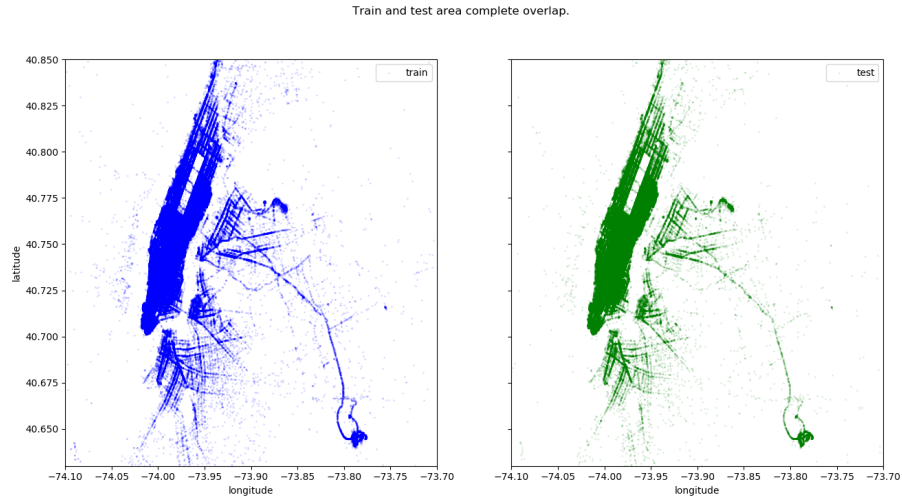


Figure 3: Scatter plot of pickup latitudes and longitudes for training and testing dataset

- Principal Component Analysis: We do a 2D  $\rightarrow$  2D coordinate transformation which could possibly help in better tree splits.
- Clustering: Using a random selection of 500000 data points, we create 100 clusters of pickup/dropoff coordinates and assign each training data point into clusters
- Open Source Routing Machine: We include additional features from OSRM dataset which has the information for the shortest paths in road network given pickup/dropoff points. We merge total\_distance, total\_travel\_time, number\_of\_steps from the OSRM dataset to our training dataset

## 2.2 Exploratory Data Analysis

This section talks about some of the exploratory data analysis that we had done before proceeding with data modeling. Before training a predictive model, we study the data distribution to summarize the main characteristics of the dataset. First, we plot the geographical features of our dataset to see which areas are the busiest in NYC (Figure 3). We perform clustering on the pickup and drop-off coordinates and visualize the results (Figure 4).

Our final set of parameters for the model are as follows:

```
[ 'objective': 'fair',
  'metric': 'rmse',
  'boosting': 'gbdt',
  'fair_c': 1.5,
  'learning_rate': 0.2,
  'num_leaves': 60,
  'bagging_fraction': 0.95,
  'bagging_freq': 1,
  'bagging_seed': seed + n,
  'feature_fraction': 0.6,
  'feature_fraction_seed': seed + n,
  'min_data_in_leaf': 10,
  'max_bin': 255,
  'max_depth': 10,
  'reg_lambda': 20,
  'reg_alpha': 20,
  'lambda_l2': 20,
  'num_threads': 30]
```

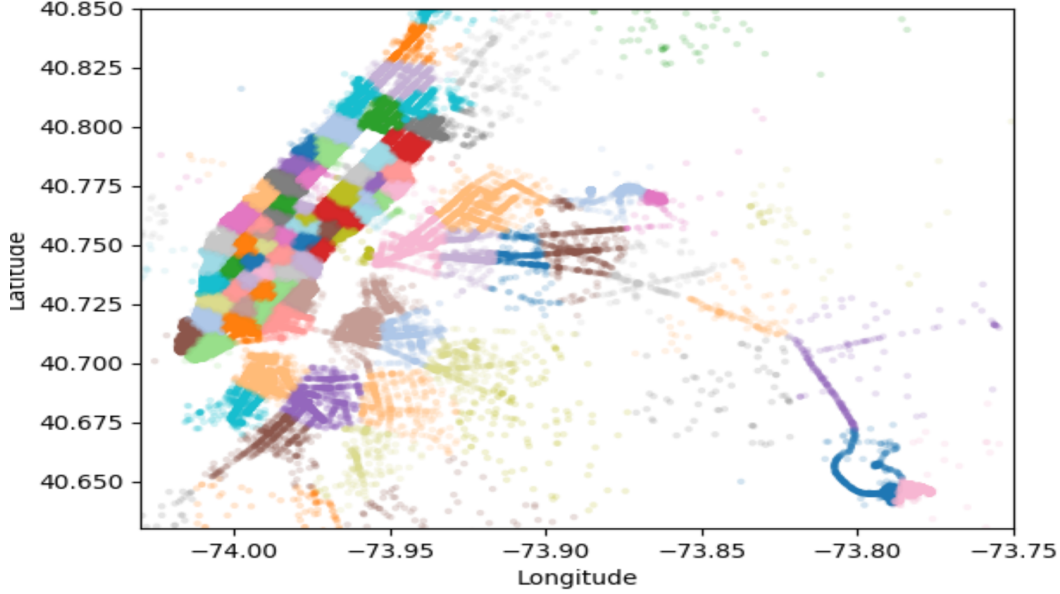


Figure 4: Clustering results using pickup and drop-off coordinates.

### 3 Results

Our results show that LightGBM when optimized on fair loss objective with optimal parameters performs better than any other model. Moreover, it can be trained  $30\times$  faster than XGBoost, another widely used boosting algorithm, which falls just behind LightGBM in terms of RMSLE score. This inference is based on our experiments of implementing XGBoost and attempting to fine tune the model with extra set of features (like OSRM data, Weather Data, Holiday Information Data, Airport distance, cleaning the noisy data etc.) and we found that we were not able to bring down the RMSLE below 0.374. The parameters of XGBoost that we were using were as follows:

`'min_child_weight':10, 'eta' : 0.05, 'cosample_bytree': 0.5, 'max_depth' : 16, 'subsample': 0.8, 'lambda' : 1.5, 'nthread' : -1, 'booster' : 'gbtree', 'eval_metric' : 'rmse', 'silent' :1, 'objective' : 'reg:linear'`

The results of comparison are as follows:

Model	RSMLE	Training time (min)
XGBoost	0.374	420
LightGBM	0.373	10
LightGBM (30 estimators)	0.37122	180

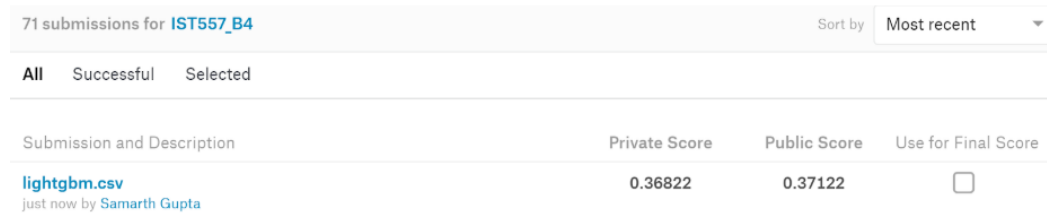
We then compare the performance of four different regression loss functions

Model	L1	L2	Huber	Fair
LightGBM	0.375	0.377	0.37354	0.37349

Furthermore, we experimented with our model by incorporating additional features such as weather data, airport distance, national holidays and other features which we believed may help towards a better prediction. However, to our surprise, the additional features did not help in any sort of improvement in terms of testing error (RMSLE). We compare the results of models after including these features.

Model	RMSLE
XGBoost	0.374
LightGBM with airport distance	0.37353
LightGBM with weather data	0.37359
LightGBM with holiday data	0.37364
LightGBM (single estimator)	0.37349
stacked GBM (30 estimators)	0.37122

### 3.1 Final Result



71 submissions for IST557_B4		Sort by	Most recent
All	Successful	Selected	
Submission and Description	Private Score	Public Score	Use for Final Score
<a href="#">lightgbm.csv</a> just now by Samarth Gupta	0.36822	0.37122	<input type="checkbox"/>

Figure 5: Screenshot from Kaggle Website

## 4 Discussions

This section discusses some of the other methods we tried and compares it with our final model.

**Random forest:** We tried random forest with the following parameters:

(n\_estimators=100, criterion='gini', class\_weight=None, max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features='auto', max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, bootstrap=True, oob\_score=False, n\_jobs=None, random\_state=None, verbose=0, warm\_start=False).

The problem with random forest was that it was taking a long time to train on the dataset whereas lightgbm was very fast. This is why we feel that this is not a good method and preferred using the lightgbm method. We also believe that the accuracy for random forest would not be much better than lightgbm and spending much training time did not seem to be logical.

**Linear Regression:** Tried the following variants from sklearn module.

linear\_model.LinearRegression

Following are the accuracies of linear regressor we tried.

Public: 0.58937

Private Score: 0.58431

linear\_model.Lasso - The validation rmse was 0.99. We did not submit the test file due to such bad performance. linear\_model.Ridge - Public score: 0.58946

Private Score: 0.58441

At this point what was clear to us is that it is XGBoost v/s Lightgbm, and we preferred Lightgbm solely due to the reason that it was 30x faster, hence we could see its results sooner and try different variations of it. The performance was comparable for both the methods.

Furthermore, there were several additional Datasets which we supposed may benefit the final model. Holiday dataset, which tells whether a given day is a national holiday, could affect the traffic and hence, the trip duration. Trip duration could also depend on the weather. For instance, rains or snow could lead to delay in the trips. Another feature which we looked into was distance of pickup or dropoff location from the nearest airport with an intuition that locations near the Airport are crowded and may lead to longer trip durations. However, none of these features gave us an improvement on the test RMSLE score when trained with LightGBM. Hence, we decided to remove these features which simplified our feature matrix and led to a better RMSLE score.

## 5 Summary

- There were many technical concepts which we learnt while working on this problem statement. Some of them were how to perform ensemble techniques, how to do an efficient parameter search, how to handle the trade off between accuracy and training time. It was interesting for us to know how one could downsample the data for training purposes so that the training time of XGboost could be lowered. Another interesting thing to know was about the GPU support of XGboost.
- We were initially training our models on our laptops itself, but then later trained it on new resources such as Google Cloud Platform and Google Colab. Working in a team was really efficient for both of us, as it not only helped us learn from each other but we were also able to speed up the process of working by distributing tasks among ourselves. Although communication seemed to be difficult before the start of the project, but it went quite smooth throughout the project and there was no sort of misunderstanding among our group. One thing which we wish we knew earlier is how to downsample the training data for XGBoost to reduce its training time. If this was known to us then we definitely would have stacked our current best model with a version of XGBoost.
- After working on all the three checkpoints and finishing with a public score of 0.37122 which is 88<sup>th</sup> in the world leaderboard, we feel that we have performed quite well. There were a few things which could have been done such as cleaning of the noisy data, since the data set had some erroneous samples. We tried some preliminary cleaning but it failed to improve the accuracy. Hence we spent most of our time in the feature engineering and model parameter tuning. One new method to learn through this course project was the method called lightgbm. Our future plans are to incorporate the loss function that we used in this task (Fair Loss) for some other application too and see its novelty.

## 6 Team Contribution

Each of us has contributed equally in the project for all three checkpoints as well as for the final project report.

## 7 Acknowledgement

The work was done as a part of Course Project for IST 557 Data Mining: Techniques and Applications at Penn State University. The authors would like to thank the course instructor Zhenhui Li and the teaching assistant Guanjie Zheng. We thank Google Cloud Platform for providing GPU which was used in this work.

## 8 References

- [1] <https://www.kaggle.com/c/nyc-taxi-trip-duration>.
- [2] <https://lightgbm.readthedocs.io/en/latest/>.
- [3] <https://www.kaggle.com/c/allstate-claims-severity/discussion/24520>.
- [4] <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>.
- [5] <https://xgboost.readthedocs.io/en/latest/>.
- [6] <https://www.kaggle.com/headsortails/nyc-taxi-eda-update-the-fast-the-curious>
- [7] <https://www.kaggle.com/gaborfodor/from-eda-to-the-top-lb-0-367>
- [8] <https://www.kaggle.com/c/nyc-taxi-trip-duration/discussion/39545>