Machine Learning Engineer Nanodegree

Capstone Project

Wenzhe (Emma) Ding
September 14th, 2017

I. Definition

Project Overview

Currently, there are over 13,000 licensed taxicabs and over 50,000 taxicab drivers providing transportation for passengers in New York City (NYC) via street hails[1]. The NYC Taxi & Limousine Commission (TLC) has released data with detailed information of each taxi trip from January 2016 through July 2016 and hosted a kaggle competition[2] to challenge data scientists to build a model that predicts the total ride duration of taxi trips in NYC. To understand trip patterns and answer more specific questions such as what is the rush hours of NYC taxi trips, what features are relevant to predict taxi trip duration, what are the most popular taxi regions at different time of the day and etc. we need to investigate such data sets. The goal of this project is to analyze the dataset, craft features for modeling and select a machine learning algorithm which is appropriate to solve the problem.

Problem Statement

Firstly, we have conducted a comprehensive exploratory data analysis (EDA). A few analysis of the data have been done in this process:

- checked whether there are missing values in the features of interest
- conducted both univariate and bivariate analysis to obtain insights from data
- feature engineering: crafted a few more features based on its impact on the target variable and these features are used in model training as well.

According to the published kernels of this competition, many has mentioned an external dataset which is very useful for predicting taxi trip duration and can be accessed at https://www.kaggle.com/oscarleo/new-york-city-taxi-with-osrm. This dataset contains the fastest route for each data point in the train and test set. we used three files in this dataset, which are ‘fastest_routes_train_part_1.csv’, ‘fastest_routes_train_part_2.csv' and ‘fastest_routes_test.csv'.

Afterwards, we implemented both decision tree (DT) and gradient boost tree (GBT) algorithms to predict trip duration. This process includes splitting the training data into train-validation set and train models using training set. Finally, we compared the performance of the best performed DT model (obtained by grid search) and GBT model.

[1] http://www.nyc.gov/html/tlc/html/industry/yellow_taxi.shtml
[2] https://www.kaggle.com/c/nyc-taxi-trip-duration

We also obtained feature importance from the results of GBT, which is used as a measurement to compare the impact among features on the target variable. Since there's no labels in the test set, performance on test data is compared based by the rank on the kaggle competition LeaderBoard. Note that we were still able to submit our prediction and obtain the rank after the competition was ended.

Metrics

The evaluation metric for this project is Root Mean Squared Logarithmic Error (RMSLE), which is defined as follows[3]:

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

Where:

n is the total number of observations in the (public/private) data set,
pi is your prediction of trip duration, and
ai is the actual trip duration for ii.
log(x) is the natural logarithm of x
Basically, lower RMSLE indicates better performance. This metrics is choose due to the following reasons:

- It does not penalize huge differences when both prediction and actual values are huge numbers. Therefore, this metric is less sensitive to outliers compared to Root Mean Squared Error (RMSE), which is defined as:

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

- It penalizes underestimates more than overestimates, which is preferable in this case because if a passenger uses our prediction as a reference and our predicted trip duration turns out to be shorter than the actual one, that passenger can spend extra time at ease. By comparison, if a passenger spends more time on a trip than he/she has expected, this might mess up his/her original schedule.

[3] https://www.kaggle.com/c/nyc-taxi-trip-duration#evaluation

II. Analysis

Data Exploration

The taxi trip data was originally published by the TLC, and is based on the 2016 NYC Yellow Cab trip record data made available in Big Query on Google Cloud Platform[4]. The data was sampled and cleaned for the purposes of the competition. The training set contains 1458644 trip records while the testing set contains 625134 trip records. Data fields in both data set are :

- 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
- 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

There are two features only available in train data, which are:

- dropoff_datetime - date and time when the meter was disengaged
- trip_duration - duration of the trip in seconds

A sample of the train data is shown in the table below

id	id2875421
vendor_id	2
pickup_datetime	2016-03-14 17:24:55
dropoff_datetime	2016-03-14 17:32:30
passenger_count	1

[4] https://cloud.google.com/bigquery/public-data/nyc-tlc-trips

pickup_longitude	-73.9821548462
pickup_latitude	40.7679367065
dropoff_longitude	-73.964630127
dropoff_latitude	40.7656021118
store_and_fwd_flag	0
trip_duration	455

After a few initial checks on the provided data set, we found all the ids are unique, there is no missing value in the training or test data and there are two unique values in column “vendor_id” as well as column “store_and_fwd_flag”.

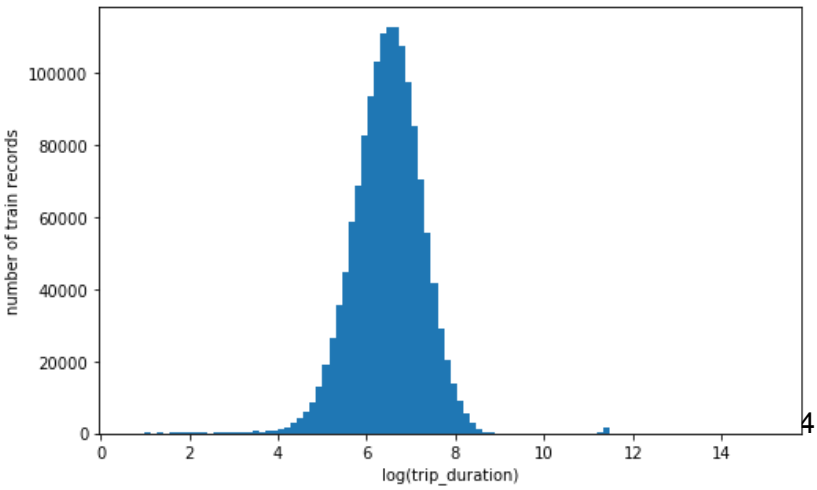
vendor_id	Number of samples
1	678342
2	780302

store_and_fwd_flag	Number of samples
Y	8045
N	1450599

Basically, the provided data set is cleaner than real word data, and there is no need to deal with missing data in this case.

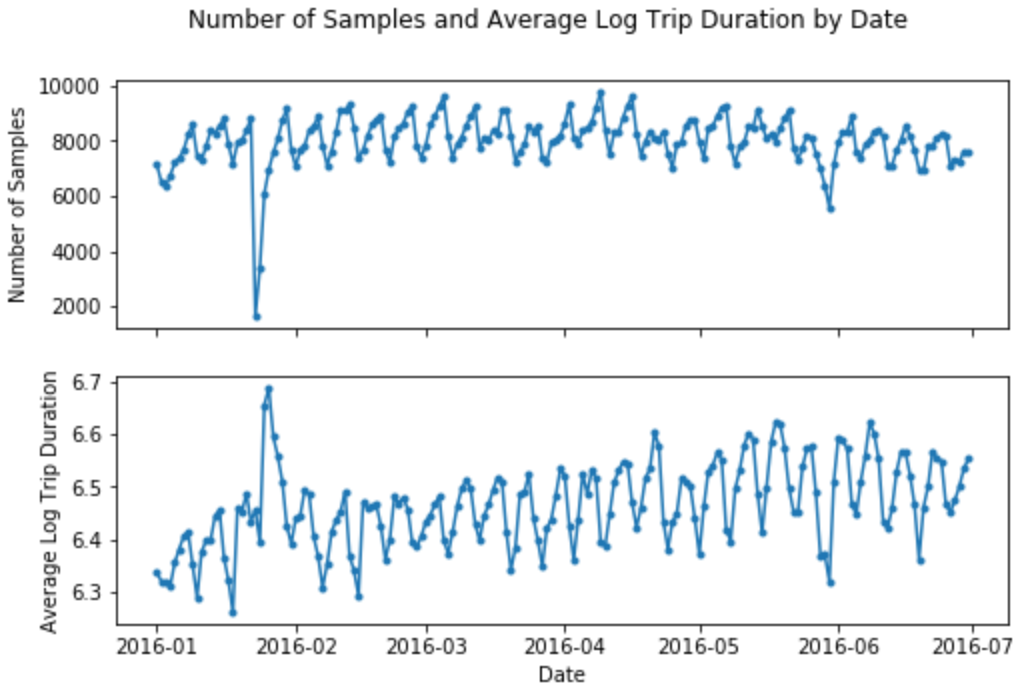
Exploratory Visualization

Since the evaluation metric is RMSLE and not RMSE, we log transformed the target variable and defined it as “log trip duration”(LTD) for the following analysis. The graph below demonstrates the histogram

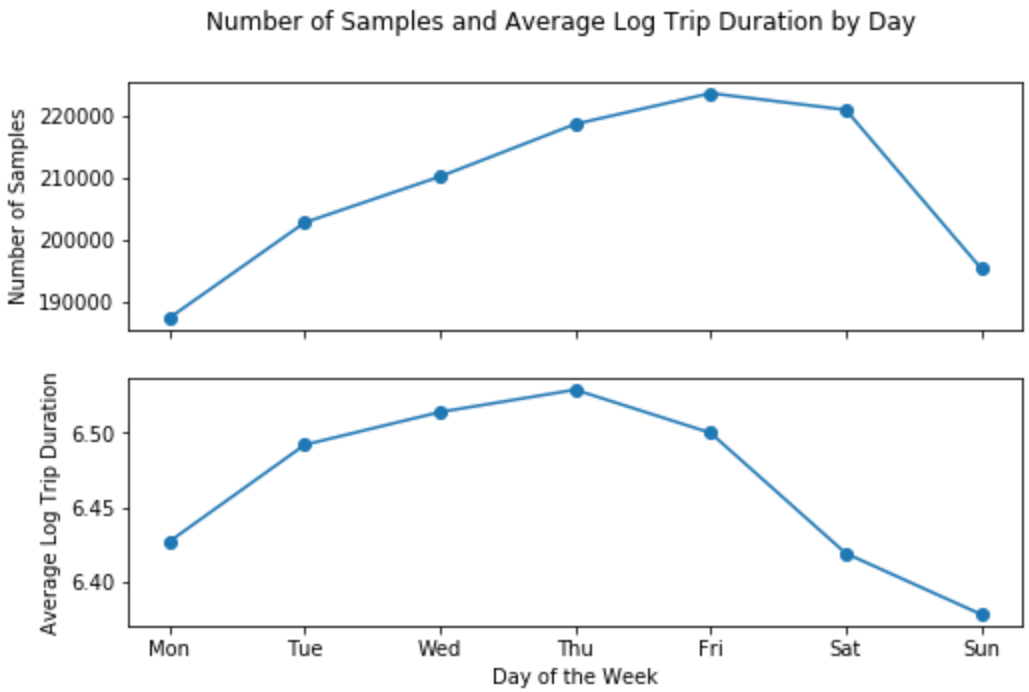


of the LTD in train data. LTD seems to fall along a bell-shaped distribution peaking at around 6.5.

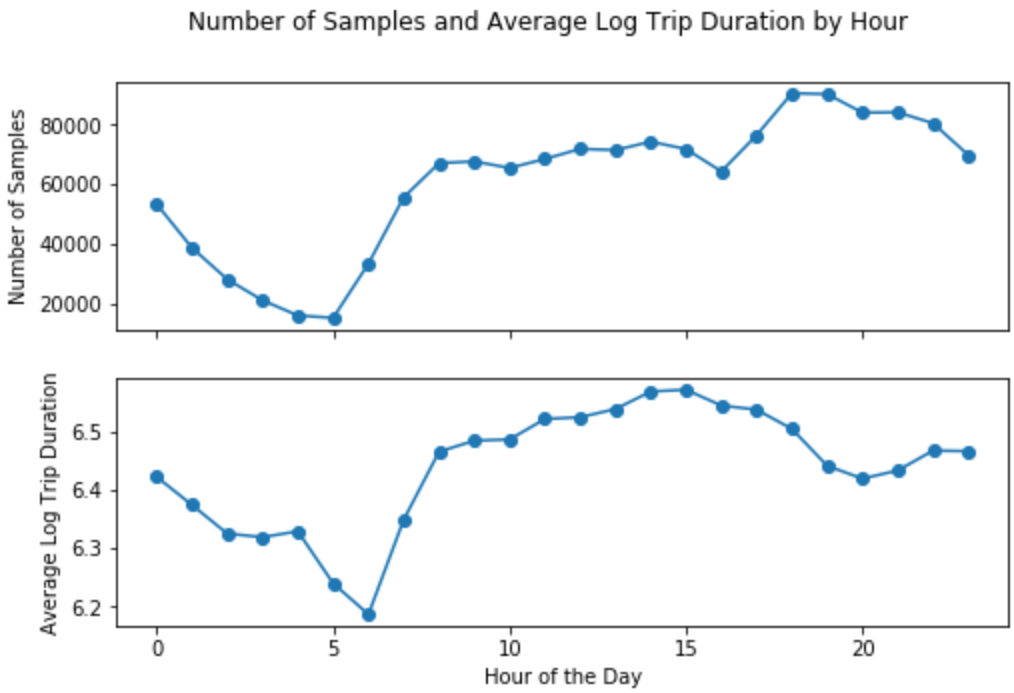
The number of taxi trips fluctuates around 8000, with a dramatic drop on January 23, 2016. The average LTD ranges from 6.26 to 6.69 with a peak on January 23, 2016.



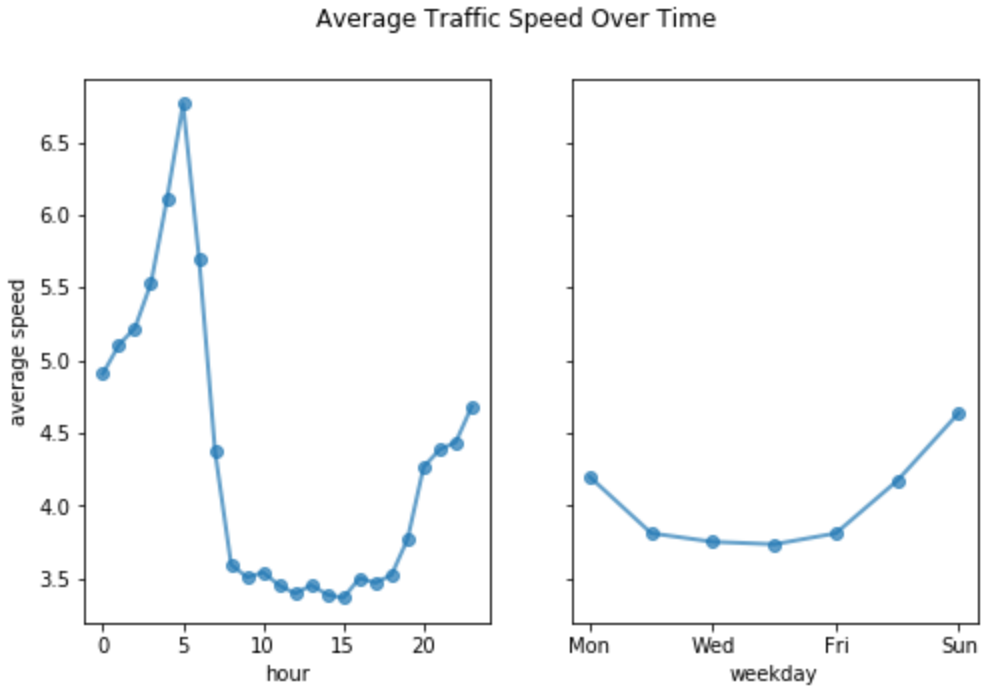
Another insight on taxi trip pattern can be found in the graph below, which illustrates the trip volume and average LTD by day of a week. Taxi trip volume increases from Monday to Friday and decreases from Friday to Sunday. The maximum trip volume, reaches over 220,000, occurs on Friday while the minimum happens on Sunday. Similar patterns appear of average LTD, but the peak is on Thursday. In a nutshell, this plot implies that people take less taxi trips on weekends than weekdays and trip duration, on average, was longer on weekdays than weekends.



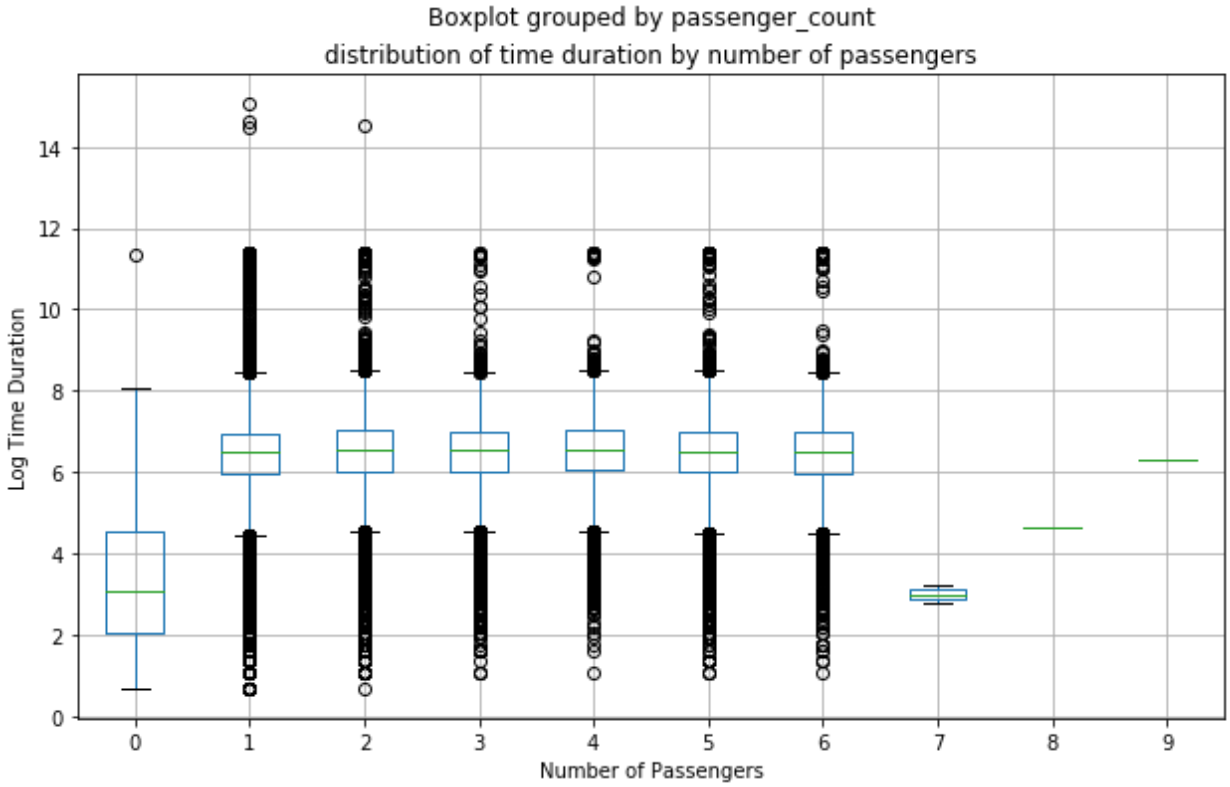
We are also interested in how rides changes by time of day. In the graph below, we see spikes in trip volume at 8am and 6pm. These users must be taking a taxi to get to work and to head home. There doesn't seem to be a lunchtime rise among taxi usage, so users may use taxis throughout the daytime at their leisure. Average LTD starts to increase around 6 am and the rate of increase slows down after 8 am. The change of average LTD seems to follow that of trip volume, meaning that as trip volume increases, average LTD increased.



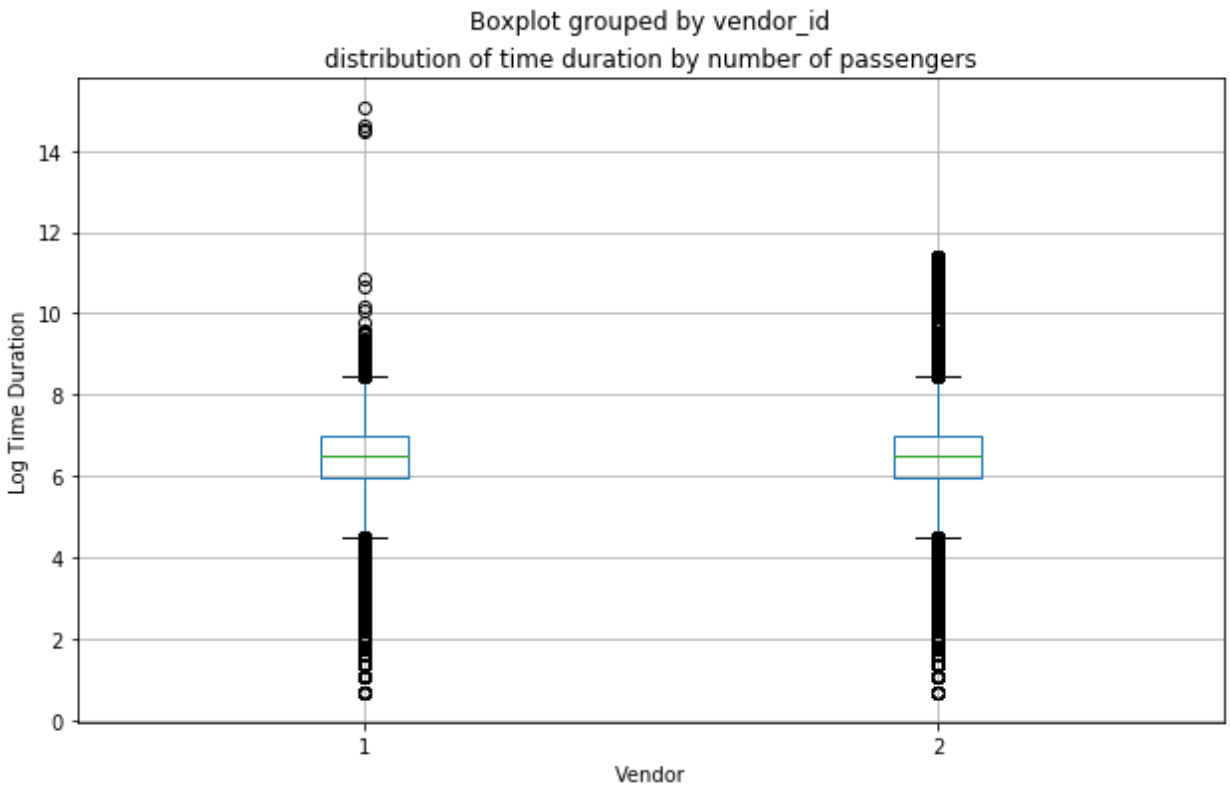
Using aforementioned external distance and duration dataset, we were able to calculate average speed at different time of a day and different day of a week. Basically, the change in average speed is inverse to the change in trip volume, for both daily and weekly change, implying that the more taxi on the road, the lower the speed.



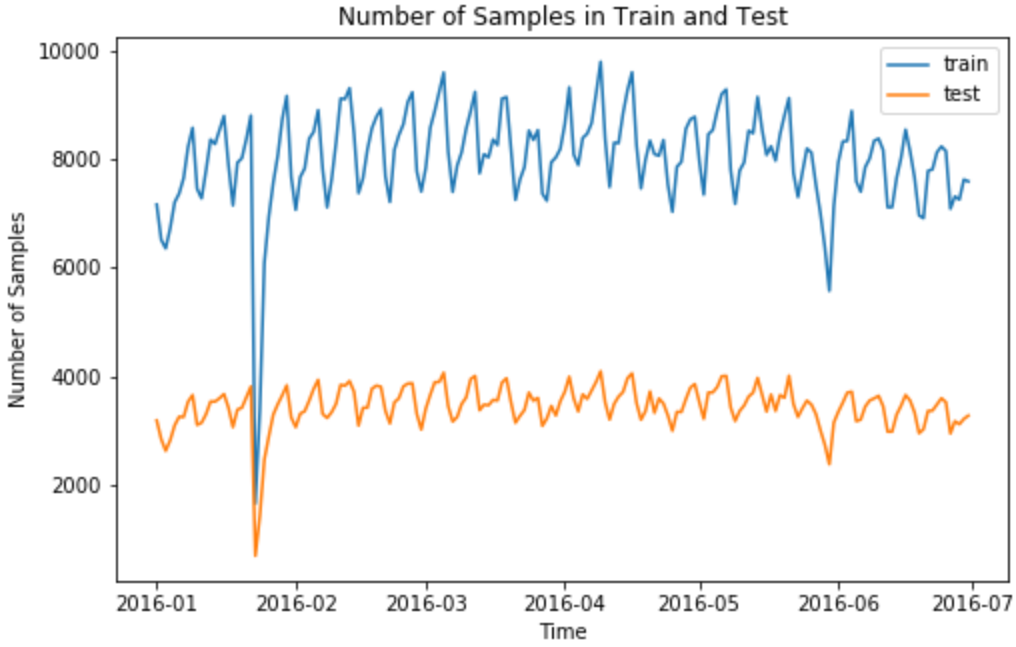
Interestingly, there are certain amount of taxis that have no passengers, the median of LTD of such trips is around 3. As number of passengers increases from 1 to 6, the median of LTD remain constant. However, this number drops dramatically when adding one more passenger to the taxi.



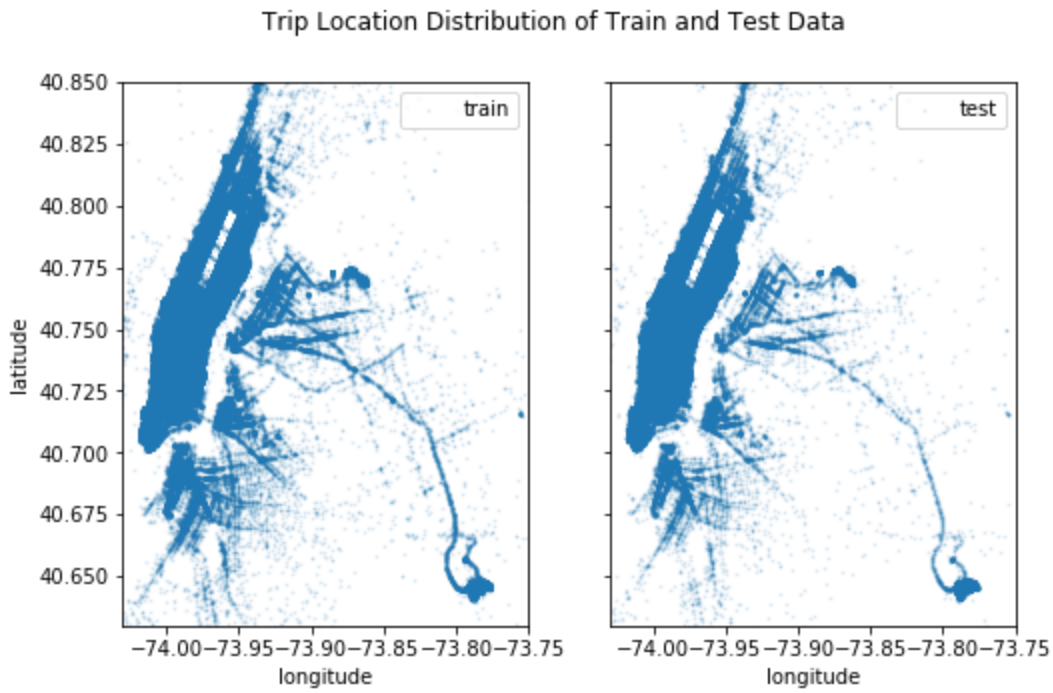
Comparing the LTD among two vendors, we found that the two vendor have similar median, but vendor 1 has more outliers than vendor 2.



Understanding the difference between train and test set is also necessary because if there is huge difference between them then feature extracted from train set would not apply to test set. The graph below shows the number of samples on each day from January 2016 to July 2016.



The plot below compares the geographic distribution of taxi trips in train and test data.



It seems like train and test data are completely overlapped in terms of time and geolocation distribution.

Algorithms and Techniques

There are many possible methods to predict trip duration in this case and a widely used one is GBT regressor, which is one of ensemble methods to deal with supervised learning problems. The main principle behind this method is that a group of “weak learners” can come together to form a “strong learner”. They typically less prone to overfitting and make the model more robust,unlikely to be influenced by small changes in the training data. The input of this algorithm is a set of numerical features, so we converted available features into numerical features or extracted numerical features from existing features. And the output of GBT is a number, which is the prediction of the algorithm.

Benchmark

A benchmark result is from a best-performed DT regression, which is obtained using grid search. The only hyperparameter that was tuned is 'max_depth', and the best value is 9 based on the result of grid search. We used the RMSE of this DT, which is 0.436, as a benchmark result to evaluate the performance of a GBT regressor. Note that we

log transformed the target variable, so RMSE of the model is equivalent to RMSLE of the original target variable.

III. Methodology

Data Preprocessing

There are 9 fields in common between train and test data with two of them - “pickup_datetime” and “dropoff_datetime”- store timestamp object and one of them - “store_and_fwd_flag” - is a character variable. The rest 6 fields are numeric features and we used all of them directly in the model training.
We extracted 4 time-related numerical features from the two timestamp objects and converted the “ store_and_fwd_flag” feature into a binary variable so that all of them could be input to the models. The 4 time related features are:

- 'pickup_month' is the month of a taxi trip
- 'pickup_weekday' is the weekday of a taxi trip
- 'pickup_hour_weekofyear' is the week ordinal of the year of a taxi trip
- 'pickup_hour' is the pickup hour during a day

From **Data Exploration** section, we found that the target variable “log_time_duration” may contain outliers since there are huge numbers showing in the plot, but it falls on a normal distribution and outliers seem to be not significantly affecting the prediction.
As mentioned before, we used three features in external distance data set for model prediction, which are:

- Total_distance: measured between the pickup coordinates and the drop-off coordinates. The unit is meters.
- Total_travel_time: The total travel time for a data point in seconds
- Number_of_steps: The number of steps on that trip. One step consists of some driving and an action the taxi needs to perform. It can be something like a turn or going onto a highway.

We also calculated the great-circle distance between two geo locations using Haversine formula[5]. The total number of features is 15.

Implementation

Two packages are used to implement our model, namely scikit-learn[6] and XGBoost[7]. We used function “train_test_split” in sklearn.model_selection to split data points from

[5] https://en.wikipedia.org/wiki/Haversine_formula
[6] http://scikit-learn.org/stable/
[7] http://xgboost.readthedocs.io/en/latest/model.html

"train.csv" into 80% training set and 20% validation set and only training set was used for model training. Validation set was used for parameter selection and to avoid overfitting of the model being built. DecisionTreeRegressor was imported from sklearn.tree to train DT model and GridSearchCV, imported from sklearn.model_selection, was utilized to select the best hyperparameter of the DT model. In this case, we tried 8 values, ranging from 3 to 10, for the “max_depth” of the tree, and GridSearchCV returned 9 as the best depth.

XGBoost was used to implement GBT regressor. A parameter dictionary and training data set have passed to the function, and the dictionary is specified as follows : {'min_child_weight': 50, 'eta': 0.3, 'colsample_bytree': 0.3, 'max_depth': 9, 'subsample': 0.8, 'lambda': 1., 'nthread': -1, 'booster' : 'gbtree', 'silent': 1, 'eval_metric': 'rmse', 'objective': 'reg:linear'}. We kept using 9 as the “max_depth” because this value is obtained by grid search for the DT model. In this case, we used 60 rounds for model training, meaning that the model is trained until the RMSE of validation data hasn't improved in 50 rounds. Note that the validation data here is not the same as the validation set that was extracted from training data using “train_test_split”. The validation data here is obtained in the model training process by XGBoost and is used to decide when to stop the model training. Finally, we obtained a GBT model with RMSE of train data 0.391 and RMSE of validation data 0.405.

Refinement

We used all 15 features for GBT model training and obtained a RMSE as 0.405 on the validation set (the one extracted from original training data). Since there are three sets of features are strongly correlated, which are “total_distance” and “total_travel_time”, “pickup_month” and “pickup_weekofyear”, “total_distance” and “haversine_distance”, we eliminated one of the correlated features each time and trained the GBT model again. However, none of them has better performance than the initial model with all the available features. Therefore, our final model was the initial model with all 15 features.

We used custom search rather than applying grid search to tune the parameters of the GBT because a single model could take hours if using grid search and obtaining the best GBT is not the focus of this project. The aim is to discover a proper solution for the problem which have better performance than the best-performed DT model. The final hyperparameters of GBT model are specified below:

{'min_child_weight': 50, 'eta': 0.3, 'colsample_bytree': 0.3, 'max_depth': 9, 'subsample': 0.8, 'lambda': 1., 'nthread': -1, 'booster' : 'gbtree', 'silent': 1, 'eval_metric': 'rmse', 'objective': 'reg:linear'}.

IV. Results

Model Evaluation and Validation

Since we log transformed our target variable, we then used RMSE as a measurement during model training process. The final parameters of the model were mostly default values of the function xgboost.train' and 'max_depth' is obtained by grid search in the DT model. Based on the value of rounds we specified, GBT model training did not stop until the RMSE of validation set hasn't improved in 50 rounds. The model is robust enough as when we submit predictions from GBT to the competition, we obtained the same RMSE for the test data, meaning that the performance on the validation set is a reasonable measurement of the performance and the model generalizes well on unseen data.

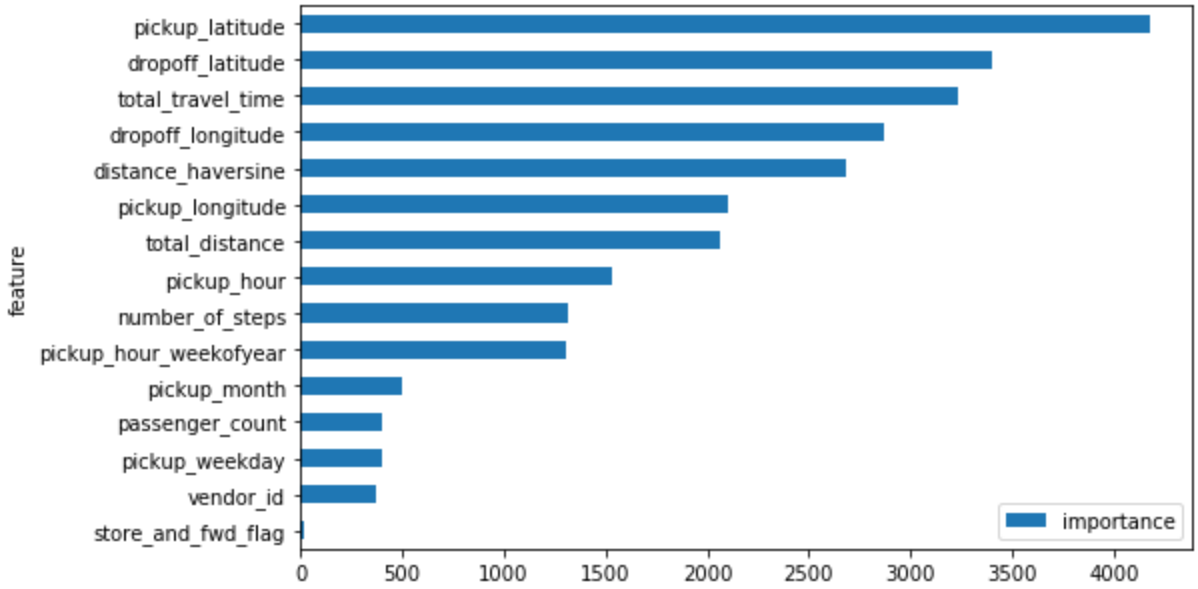
Justification

Comparing with the benchmark model with RMSE 0.436 on validation set, our final GBT model has better performance. In order to compare the RMSE on test data, we submit predictions from both model to the competition, and it turned out GBT has lower RMSE on test data.

V. Conclusion

Free-Form Visualization

The plot below shows the feature importance of each features in the data set. It is clear that location related features - both pickup and drop off location - have high feature importance score. Direction is important from both aspects. It does not have many correlated feature and removing it would hurt the model. Both distances - haversine distance and total distance from external dataset - have significant impact on the target variable. Vendor_id is the second least used feature according to feature importance plot but removing it increases the rmse significantly.



Reflection

In the project, we started from EDA using the given dataset and identified patterns in taxi trips. Then we spent time on crafting feature sets for model training and it turned out the performance is the best when training model with all available features rather than eliminating correlated features. The data we were using have some unique characteristics:

- Train and test data are cleaned by the host of the competition, i.e. we are not dealing with messy data. There was no missing values in either train or test data, the target value is normally distributed.
- Train and test are almost completely overlapped in terms of time and geo locations, therefore it's safe to use features crafted based on train data on test data

Based on these findings, feature engineering is critical in this case, which is also the focuses of many participants of this competition[8]. The final model outperformed the benchmark result, thus it fit my expectations for the problem. And due to the nature of ensemble methods, it could be used in a general setting to solve these types of problems.

This project is challenging in terms of figuring out how to extract useful information and features from available data. For example, based on my previous experience exploring Chicago taxi trips[9], I discovered daily trip patterns of Chicago taxis

[8] https://www.kaggle.com/frednavruzov/nyc-taxi-eda-feature-engineering and https://www.kaggle.com/gaborfodor/from-eda-to-the-top-lb-0-367.
[9] https://wzding.github.io/chicago_taxi.html

which shed light on the assumption that extracting time related features from NYC tax trips may be helpful in trip duration prediction. Also, I hypothesised that great circle distance should be relevant to predict trip duration cause taxi drivers always want to take the shortest path from trip origin to destination, and it turned out such distance is an important feature to predict our target variable.

The most interesting aspect of this project is to learn how data competitions differ from real world problems. One major difference is that real word data is more messy and more data quality issues need to be solved before model training, but the data provided by the competition is clean and tidy. Another difference is that I prefer simple models - ones with less variables - to solve real word problems thus to avoid overfitting on training data. However, it does not seem to be a good practice for data competitions and many participants spent most of their time on feature engineering even if certain features only give marginal improvement.

Improvement

We aimed at finding a reasonable and robust solution to the problem and it turned out the GBT model is an appropriate solution to this problem, so we did not put all our efforts on feature engineering. In order to improve the performance of our model, one area of improvement lie in crafting more features for model training. As proved by some participants of the competition, the model achieved better performance with more features even if they only give marginal improvement. In addition, network analysis[10] and time series analysis[11] could also provide insights of this data set and the findings are useful for feature engineering.

We used custom search rather than applying grid search to train the GBT because a single model could take hours if using such as method and obtaining the best GBT is not the focus of this project. A few hyperparameters, such as min_child_weight, eta, colsample_bytree, max_depth, subsample and Lambda can be tuned to achieve a better performed GBT.

---

[10] https://www.kaggle.com/crailtap/basic-network-analysis-tutorial
[11] https://www.kaggle.com/mmotoki/time-series-including-nyc-s-biggest-snowstorm