Tharid Lerskiatiphanich

July 20th, 2018,

Capstone Proposal for machine learning nanodegree

New York City Taxi Trip Duration

Domain Background

Since the invention of the wheel, human civilization has been rising rapidly due to transportation of goods and led to various other innovative technologies such as car, airplane that play an important role in transportation. [1]

Taxi is one of the main transportation for humans. They are fast, comfortable, and have an affordable price for many people. Fortunately, with the rise of computing powers, collecting sensors, and modern algorithms in computer science, it is possible to improve the taxi service by predicting the total ride duration of taxi trips. If we are able to predict the ride duration at any specific time in this large scale system, taxi companies are able to use the prediction time as the input feature for their optimal price model - this is better than naive guess which might lead to overpriced tag. Passengers could also plan and estimate the destination time more accurately; more productive in their life. Lastly, drivers also are able to compare their time and fare to evaluate their returns for riding in the particular trip. This gives valuable benefits to all drivers, taxi companies, and customers.

[1] "The revolutionary invention of the wheel - ḎḤWTY", 2 JUNE, 2014, https://www.ancient-origins.net/ancient-technology/revolutionary-invention-wheel-001713

Kaggle, well-known platform using data science to solve the problems, has a dataset and challenges you to build a model that predicts the total ride duration of taxi trips in New York City. The dataset is based on the 2016 NYC Yellow Cab trip record data made available in Big Query on Google Cloud Platform. The data was originally published by the NYC Taxi and Limousine Commission (TLC). The data was sampled and cleaned for the purposes of kaggle competition. Based on individual trip attributes, participants should predict the duration of each trip in the test set. [2]

Problem Statement

The objective is to predict the total ride duration of taxi trips and also minimize the difference between the actual predictions and real duration trips as much as possible. Searching for the best algorithms and hyperparameters which minimize the errors of the evaluation metric, Root Mean Squared Logarithmic Error [3] is the solution. As I have stated in the previous paragraph, If we are able to predict the ride duration at any specific time in this large scale system, all drivers, taxi companies, and customers will have a lot of benefits especially more productive life and more reasonable taxi fare than ever.

[2] "Kaggle data source", https://www.kaggle.com/c/nyc-taxi-trip-duration/data
[3] "Root Mean Squared Logarithmic Error ", https://www.kaggle.com/wiki/RootMeanSquaredLogarithmicError

Datasets and Inputs

In this dataset from kaggle, we have 2 files: the training set containing 1458644 trip records, and the testing set containing 625134 trip records. We will focus mostly on the training set since this is our input for machine learning model to learn.

The dataset is based on the 2016 NYC Yellow Cab trip record data made available in Big Query on Google Cloud Platform. The data was originally published by the NYC Taxi and Limousine Commission (TLC). The data was sampled and cleaned for the purposes of kaggle competition. Based on individual trip attributes, participants should predict the duration of each trip in the test set.

Data field	Description [4]
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

[4] "Kaggle NY city taxi trip duration data", https://www.kaggle.com/c/nyc-taxi-trip-duration/data

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

Solution Statement

“Garbage in garbage out” - if the quality of the input data or features are bad, no matter how good your model is, the output is going to be bad, is the useful concept that has to be followed when tackling the machine learning problems. Thus, data wrangling, data cleansing, feature engineering with the expert domain knowledge are important. In this problem, I will focus mainly on feature engineering by creating new features such as distance, weather conditions, day of week, and use the machine learning model to learn and minimize the errors based on the evaluation metric, Root Mean Squared Logarithmic Error.

Benchmark Model

At first glance, I was tempted to use mean prediction as the baseline model because of its simplicity. However, it is not a good baseline in this setting because the duration time has a high variance and the model would predict too naively which make it not suitable to use as a baseline model to beat. Using plain linear regression might be more suitable for this domain problem because most of the regression problems have been using this model since early 20th century and its performance is also great for numeric problems - it is the foundation of most modern machine learning algorithms even deep learning. Another reason to use linear regression as baseline model is easily measurable; RMSLE with linear regression in this case is very easy to compare with our solution.

Evaluation Metrics [5]

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

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

[5] "Kaggle evaluation", https://www.kaggle.com/c/nyc-taxi-trip-duration#evaluation

ϵ is the RMSLE value (score), n is the total number of observations in the (public/private) data set, p_i is your prediction of trip duration, a_i is the actual trip duration for i, and $\log(x)$ is the natural logarithm of x.

The reasons I choose RMSLE for this project are: Official evaluation metric for kaggle competition and it is the standard for most regression problems.

Firstly, using the same evaluation metric that kaggle used in the competition benefits hugely to us because we can now submit our test set predictions to kaggle private leaderboard evaluating how good our model is.

Secondly, RMSLE, in fact, is the alternative version of RMSE which takes the log of both actual and predicted values before calculating. This is very helpful in this project because it does not penalize both huge actual and prediction values the same way as RMSE. For example, if we have 2 scenarios: A) actual trip duration 10 mins, predicted trip duration 15 mins. B) actual trip duration 60 mins, predicted trip duration 65 mins. If the evaluation metric is RMSE, it will yield the exact same error. In contrast, RMSLE would penalize the scenario A more because it has a smaller value; and that is more suitable for our domain because in scenario A it will surely upset passengers more than scenario B.

Project Design

I am going to start this project with exploratory data analysis. It is the mandatory for every machine learning projects in the world. Looking for descriptive statistics along with basic data visualization such as histogram or bar plot is always a good idea. We need to understand the characteristic nature of each features - basic questions should be asked such as is it skewed, what is the data distribution, is there any outliers, should we do any data transformation, is this feature relevant to our objective, is there any weird values because of mistakes during collecting data, how to treat nan values in dataset. After enough data exploration, we will execute the reasonable actions derived from that exploratory data analysis.

As I mentioned in "solution statement" part, this problem will be solved by properly feature engineering technique combining with specific domain knowledge. I think pickup_datetime, pickup_longitude, and pickup_latitude are very important because we can construct various new useful features with these. For instance, I have an idea to measure the distance between pickup_longitude/latitude with dropoff_longitude/latitude - it could possibly be an euclidean distance. Datetime features can also be extracted and create many new features such as is it holiday or not, day of week, time of day. Location is also helpful if we can create new features like zipcode or implement some unsupervised-learning method such as K-means to clustering and add that back to dataset. In addition, Weather conditions at that specific time are important. Driving a car while raining is slower than driving a car in sunny weather.

After creating new features from the raw features, we will drop all irrelevant features such as id which does not give any useful information to our model. I will use K fold cross-validation technique - splitting training set into train and validation set K times. Various regression models such as Ridge regression, Random Regressor tree, and Gradient boosting tree will be used and I will pick the best one based on the validation set performance. After that, I will search the best combinations of hyperparameters for that model by the bayesian optimization along with K-fold cross-validation technique and use that hyperparameters to train our training set again before using that model to predict test set and submit on kaggle leaderboard. The performance of local validation set and test set on kaggle leaderboard should be consistent - if the local validation set is improved, the test set should be improved too.

Iterating all of previous steps until we have found the best score on kaggle leaderboard. I expect to spend most of my time with feature engineering and just use rough hyperparameters to yield the results until I am satisfied with the features because searching best hyperparameters before constructing good features, in my experience, does not return much great performance.