# Advanced Machine Learning Coursework: Kaggle Competition NYC Taxis

The Unicorn Hunters
(Christian Frantzen, Guilherme Barreiro Vieira,
Mortimer Sotom and Théo Verhelst )

May 10, 2018

**Abstract**

This is the abstract

## 1 Introduction

New York City (NYC) is one of the busiest cities in the world with pedestrians constantly rushing to arrive to their destination on time. To avoid searching for parking and the stress of driving during rush hour, most citizens of NYC use public transport, and especially taxis. The most important piece of information is undoubtedly the estimated trip duration for customers to plan their schedule, for the driver to find the shortest route and for deriving the estimated price of the journey. This is exactly the aim of the New York City Taxi Trip Duration Kaggle competition, build a model able to predict the total ride duration of taxi trips given a set of features such as geo-coordinates, number of passengers and others. The dataset provided is one release by the NYC Taxi and Limousine Commission (TLC) and contains over 1.45 million trip records over the entire year of 2016. Participants must then predict the trip duration in seconds for each entry of the test set containing 625,134 trip records.

Before diving into building predictive models, the data must be carefully analysed to gain an understanding of the features provided and the problem at hand while comparing the distributions between the train and test sets. The next stage is preprocessing to identify invalid data and eliminate outliers. The training set was also augmented with hourly rainfall, further distance measurements and road related informations such as number of intersections to reach the destination from The Open Source Routing Machine (ORSM). The relevance of each feature was then evaluated through Principal Component Analysis (PCA) and recursive feature selection. The evaluation metric of the Kaggle competition, Root Mean Squared Logarithmic Error (RMSLE), was used to evaluate the performance of the models models. It was also decided to predict the trip durations in minutes by classification for a more logical result as it is virtually impossible to predict this type of problem down to the exact second in a dynamic environment. Finally, a total of seven models were implemented: Adaboost, Support Vector Machine (SVM), Gaussian Process, Stochastic Gradient Descent, Random Forest, Extreme Gradient Boosting (XGBoost) and Neural Network (NN). The results of each model is presented and discussed in the last section of this report.

## 2 Data Analysis

The first step of a machine learning project is to explore and analyse the data, in order to better understand the problem. Our dataset is composed of 11 features: a unique identifier for each row, the identifier of the taxi company, the pickup time, the pickup and drop-off locations, the number of passengers, and a boolean flag indicated if the trip data has been stored on-board or directly sent to the data server.

Figures 1 to 5 show the distribution of some of these features. We did not show the pickup month, minute and seconds, as their graphs are uniformly distributed and therefore not visually informative. Outliers have been removed in order to properly display the pickup and drop-off location distribution, as well as the trip duration. These outliers will be discussed in the next section. We can see in figure 1

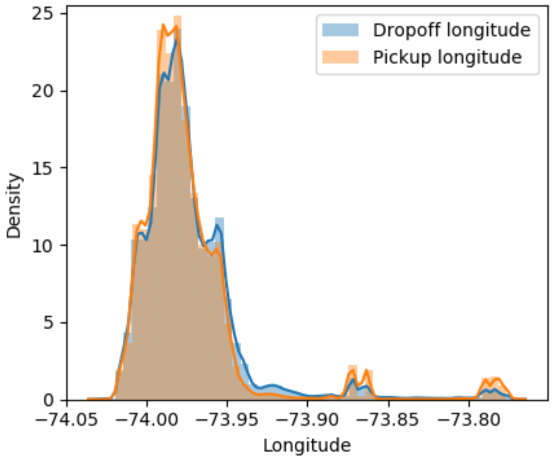


Figure 1: Distribtion of the longitude.

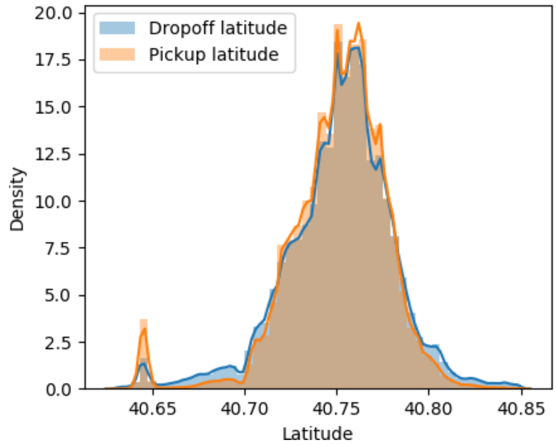


Figure 2: Distribution of the latitude.

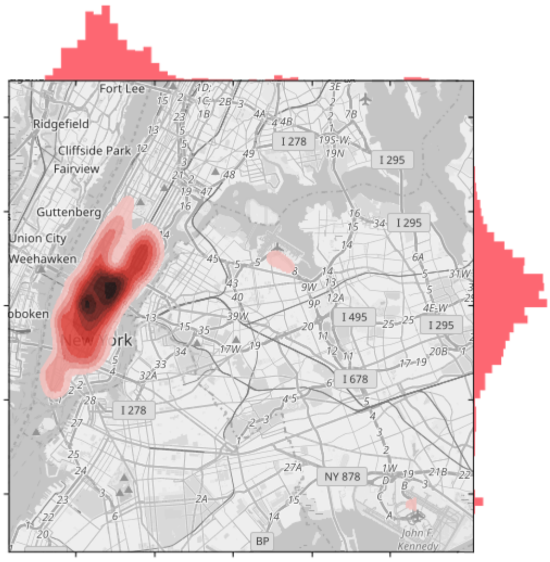


Figure 3: Heatmap of the trip locations on a map (credits: OpenStreetMap).

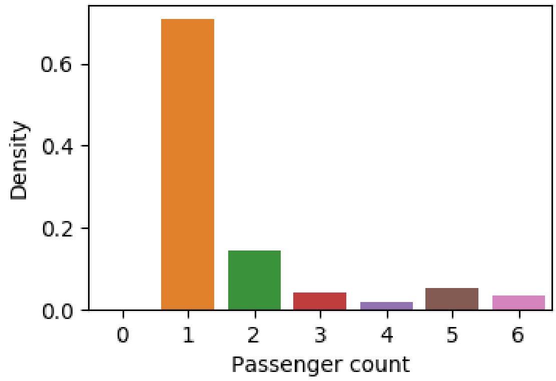


Figure 4: Distribution of the number of passengers.

and 2 that the location seems to be roughly normally distributed, and the logarithm of the trip duration also appears to be normally distributed in figure 7. The smaller bumps outside of the main bell in the location curves correspond to trips to the city airport.

It is also important to make sure that the training and test sets are independently and identically distributed. For this, we compared the above distributions with those from the test set. If the distribution from the training and testing set significantly overlap, then we can consider that the I.I.D. assumption is verified. As shown for example in figure 8 and 9, it is indeed the case.

# 3 Preprocessing

Since the data is provided by Kaggle, it is already very clean. However, when looking at the distributions of the GPS coordinates and the trip durations, we observed that some trips were outliers. We calculated the mean value for the trip duration (961 seconds) and the standard deviation (5247 seconds) and then removed all trips which were shorter than the mean minus twice the standard deviation or longer than the mean + twice the standard deviation. We also removed all trips which were not in the following bounds for their coordinates : longitude $[-74.03; -73.77]$ and latitude : $[40.63; 40.85]$.

To augment our dataset we downloaded weather data from 2016 from XX and extracted the volume of precipitation for each of our trips. Next, we used the GPS coordinates of the pickup and drop-off

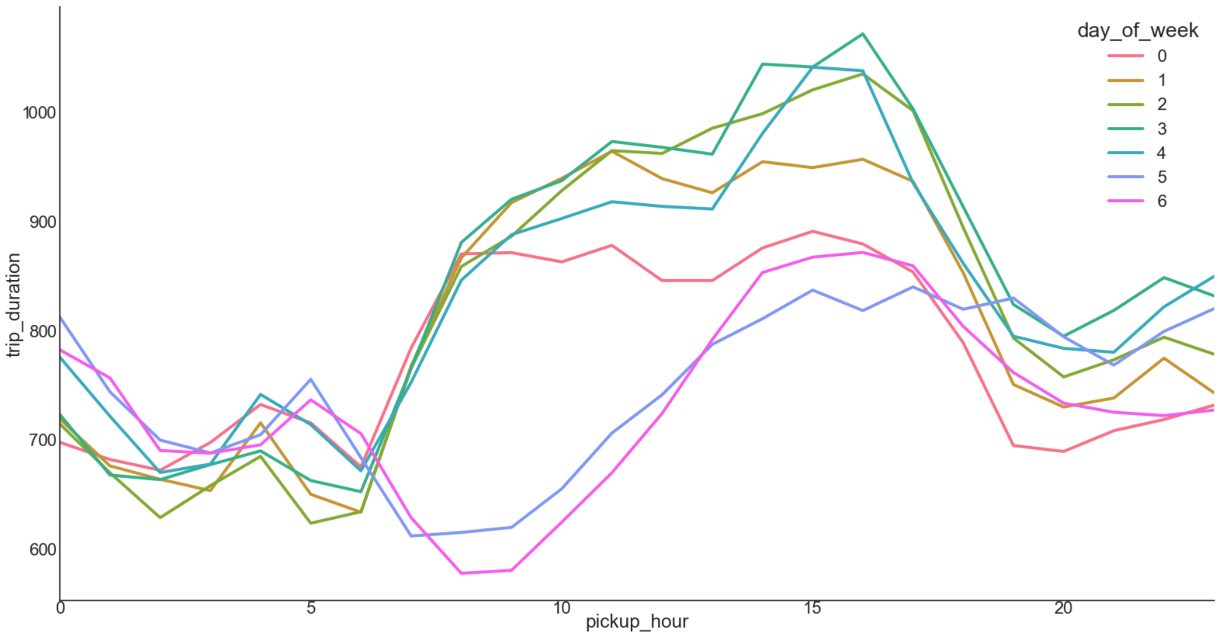


Figure 5: Distribution of the pickup time, for different days of the week.

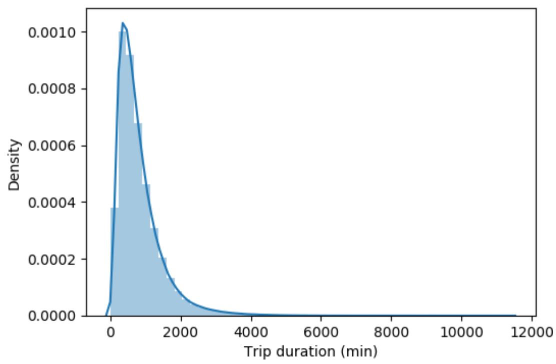


Figure 6: Distribution of the trip duration, in seconds.

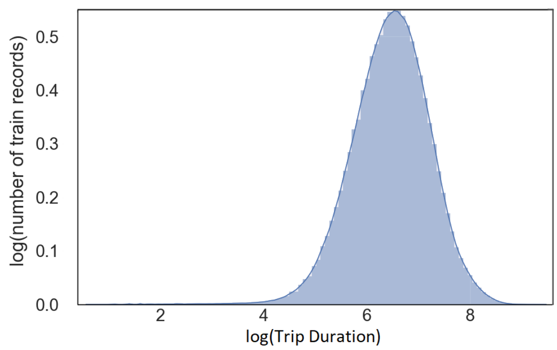


Figure 7: Distribution of the logarithm of the trip duration, in seconds.

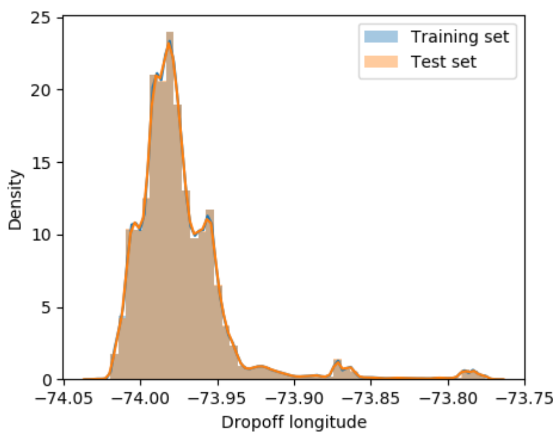


Figure 8: Drop-off longitude in the training and test set.

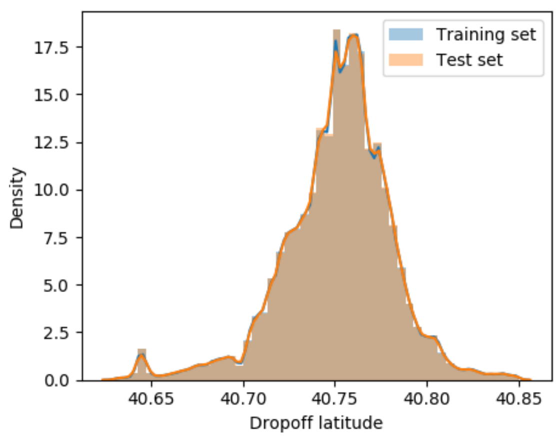


Figure 9: Drop-off latitude in the training and test set.

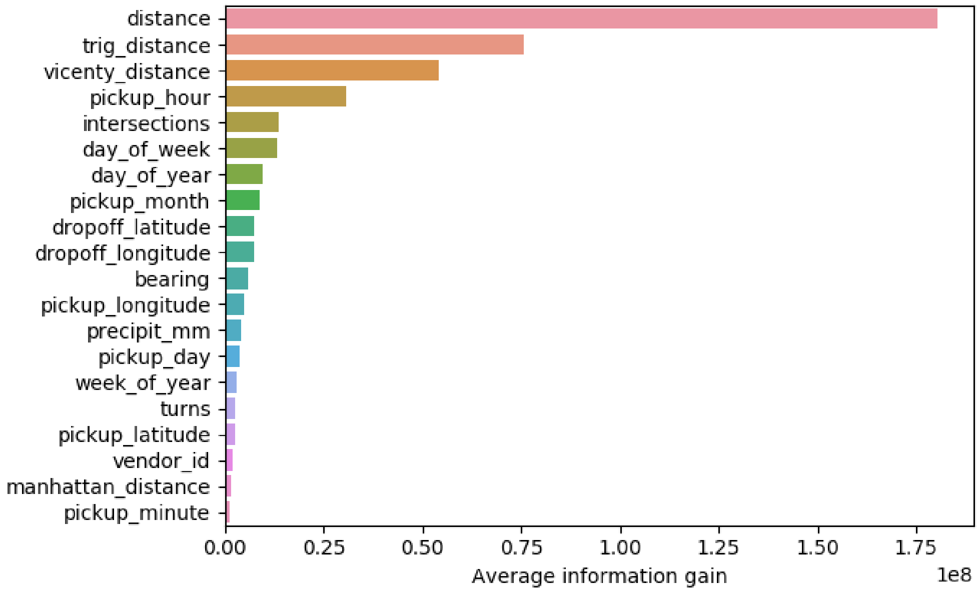


Figure 10: Information gain of each feature.

locations to calculate following properties:

- Straight distance between the two points along the surface of the earth called Vincentys distance;
- Pythagorean distance between the two points;
- Manhattan distance;
- The angle between the two points in relation to the north pole.

We also used OSRM (`http://project-osrm.org/`) to give us information about the cab ride. OSRM stands for open source routing machine and makes it possible to download maps from all around the world and calculate the street distance between two points as well as other information. Besides the street distance we included the number of turns to make before reaching the destination, how many intersections one has to pass. Apart from the data provided by Kaggle we could also have used more trip records available on the website of the NYC Taxi and Limousine Commission. However the files available are of the size of a few gigabytes per month, which is why decided not to include them.

Additionally to the data augmentation we also always scale it before passing it to a model. Sklearn provides the possibility to center every column to the mean and scale to unit variance.

# 4 Feature Selection

In order to ensure that the model was not being fed useless features, a feature selection elimination procedure was undertaken. The results are shown in the above figure - the lower the ranking the better the feature is - by using the scikit-learn package recursive feature elimination. This function attempted to find a ranking of the features by doing a prediction while removing one feature at the time. As expected the most important features such as drop-off and pickup locations as well as different measures of distance were ranked as very important. In contrast, features that didnt contain much information such as `pickup_year`, `store_and_fwd_flag`, `vendor_id` and `pickup_month`, were ranked as very low importance and hence were removed from the model. Interestingly enough, the precipitation feature providing an idea of the weather conditions, didnt rank very low leading to believe that it didnt have a significant effect in the duration of the trips within New York and specifically Manhattan which is where most of the journeys were recorded.

On the other hand an analysis of feature importance in figure 10 provides a different perspective on which features provide more information gain to the model. It is expected that the distances add the most significant amount of information with exception of the Manhattan distance which apparently adds little information - contradicting the RFE analysis. Furthermore, the amount of intersections, day of the week and day of the year have significant amount more value to the model than previously. This is a good indication of the different amount of congestion at different times of week and year whereas in this analysis it seems that the coordinate locations add less value as they change less due to being majorly close by within the city. Finally, note that some of the features within this information gain analysis have been removed from the last one, however, by removing any more features within this selection, the models error increased, hence, this selection of features was chosen as the final one.

# 5 Methods for Model Selection

In order to assess the performance of each model to be tested and compare the results, an error metric was selected. To remain constant with the scoring of the Kaggle competition, the same evaluation function was chosen: Root Mean Squared Logarithmic Error (RMSLE). This allows to estimate the leaderboard ranking of the results and since it is logarithmic, it ensures that the prediction for short trips have the same weighting as the predictions for long trips. The RMSLE is calculated by:

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

Where $n$ is the total number of entries in the data set, $p_i$ is the predicted trip duration and $a_i$ is the true trip duration. The dummy regressor was implemented to use as a benchmark since its predictions employ basic rules, to further compare the regression models. It achieved a high RMSLE of 0.810 for the prediction in seconds and 0.802 for the prediction in minutes. Additionally to predicting the trip duration in seconds through regression, each model was designed to predict the trip duration in minutes through classification for a more logical and interpretable estimate. In real life scenarios, it would be impossible to predict a taxi journey to the exact second due to the dynamic environment of the roads.

# 6 Results

## 6.1 Neural Networks

Neural networks (NN) performed well in this data set with some hyper-parameters tuning. At first, due to the nonlinearity of the data, models with up to 3 hidden layers and 512 nodes were experimented on, however, the results were far from satisfactory. As one tried to reduce the complexity of the model, in which the optimum point was found to be with three layers of 64 nodes - errors reduced down to around 0.35 RMSLE. Following this, hyper-parameters grid search such as the type of solver, activation function, batch size and amount of regularisation were conducted which led to finding our optimal model. The Adam solver was the most efficient which was expected as it is a good solver for large datasets, with a relu activation function, some regularisation and small batch sizes of 200, which brought the error down to 0.346.

## 6.2 Random Forest

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. This algorithm proved to be a powerful algorithm for this dataset as it brings an extra aspect of randomness into the model, as when it is growing the trees, it searches for the best feature among a random subset of features instead of searching for the best feature while splitting a node. This increases the diversity of the model which in this case resulted in better results. The key hyper-parameters in this model were the number of estimators (number of trees in the forest), the minimum number of samples required to split a node and the minimum number of samples required to be at a leaf node. The prediction error was always reduced by the number of estimators of which it was increased up to the hardware memory capability - 125 estimators which led to an error of 0.36. On the other hand, the other parameters were more finely tuned to reach the conclusion that the lower the number of minimum samples split and minimum samples leaf, the higher the overfitting likelihood within the model. Hence, by increasing these parameters, an optimum error value was found to be at 0.346 thereby reaching one of the most performant models within the trials conducted.

### 6.3 Support Vector Machine

The power of Support Vector Machine (SVM) is appealing, given its low generalisation error. However, the time complexity of the training phase of the SVM is proportional to the cube of the number of training examples, and as a result it is impossible to train on our entire dataset, which is composed of more than 1,400,000 samples. Therefore, we trained a SVM regressor on a random subset of the training set, and the size of this subset could range up to about 20,000 samples. Past this value, the training time is simply unbearable.

On the other hand, since the evaluation of a trained SVM model is fast, we could validate our model on a larger portion of the training set, thus giving a strong estimation of the generalisation performance. It turns out that by training a SVM on 20,000 samples, testing it on 100,000 samples and with a suitable hyper-parameter optimisation, we were able to achieve a RSMLE of 0.359.

### 6.4 Gaussian Process

Just as the support vector machine, the Gaussian Process (GP) has a time complexity of $O(n^3)$. Therefore, we were not able to train a GP model with the whole dataset. After a grid search to find the optimal hyper-parameters, we trained a model with a rational quadratic kernel, and a noise reduction parameter $\alpha = 0.01$. This model has been trained on 1000 random samples, and validated on 100,000 other samples. The RSMLE was calculated to be 0.469685, which is most probably due to underfitting, due to the very limited number of training samples.

### 6.5 Extreme Gradient Boosting

Since this problem uses a relatively large dataset, it was expected that Extreme Gradient Boosting (XGBoost) would perform well. Using the default parameter values and all of the available features, it produced a baseline score of 0.394. After applying grid search to optimise the hyper-parameters and removing features suggested by the feature selection elimination, the RMSLE was reduced to 0.346, making XGBoost perform ever so slightly better than NN and becoming the best tested model.

### 6.6 Stochastic Gradient Descent

Using the Stochastic Gradient Descent regressor from sklearn, we got an RMSLE of 0.487 without any hyper-parameter tuning. Using grid search however we managed to achieve an RMSLE of around 0.455. While we do scale our features before running the model, it does not seem to get closer to the true objective function. This may be due to the nature of GPS coordinates which are difficult to linearise.

## 7 Conclusion

## References

# A Additional Figures