Advanced Machine Learning Coursework: Kaggle Competition NYC Taxis

The Unicorn Hunters
Christian Frantzen (29780322)
Guilherme Barreiro Vieira (26286874)
Mortimer Sotom (29785056)
Théo Verhelst (29799031)

May 11, 2018

Abstract

Accurate prediction of taxi trip duration is an important part of efficient customer service for taxi companies. To this end, the New York City Taxi and Limousine Commission organised a competition on the Kaggle website, aiming to stimulate research in taxi trip duration regression. This report presents the work that our team conducted on this problem. A classical machine learning pipeline has been applied: data analysis, data cleaning, feature selection, model evaluation and optimisation. The models that have been tried range from Ridge Regression to Extreme Gradient Boosting. An unlabeled test set is proposed to the competitors in order to accurately evaluate the generalisation performance of their models. The predictions of our best models on this test set gave us a decent ranking in the top 47% of the competition leaderboard.

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 released 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. The next step was preprocessing the data and augmenting it with hourly rainfall, further distance measurements and road related informations from The Open Source Routing Machine (ORSM). Section four explains how the relevance of each feature was evaluated by information gain and recursive feature selection. The following section describes the chosen evaluation metric, Root Mean Squared Logarithmic Error (RMSLE), and the reasoning behind also predicting the trip durations in minutes by classification. 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 pick-up time, the pick-up and drop-off locations, the number of

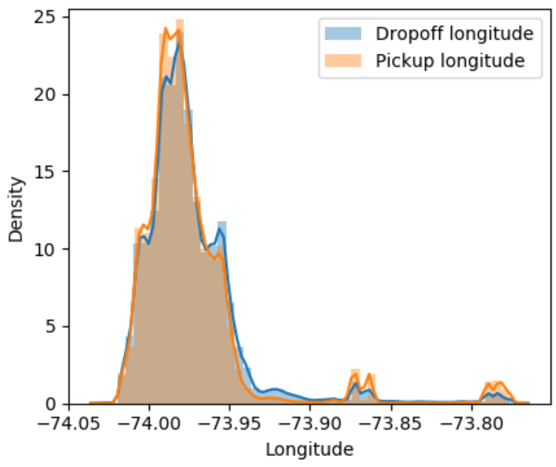

Figure 1: Distribtion of the longitude.

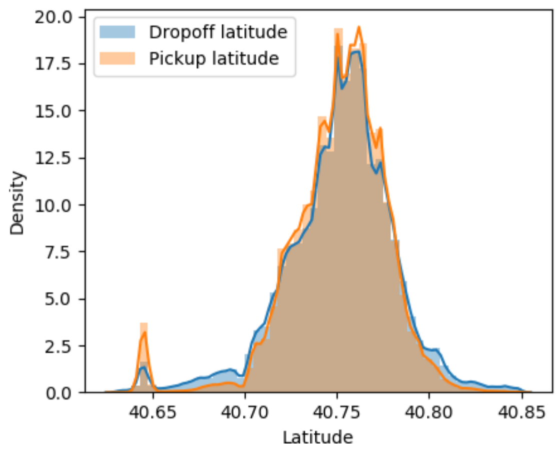

Figure 2: Distribution of the latitude.

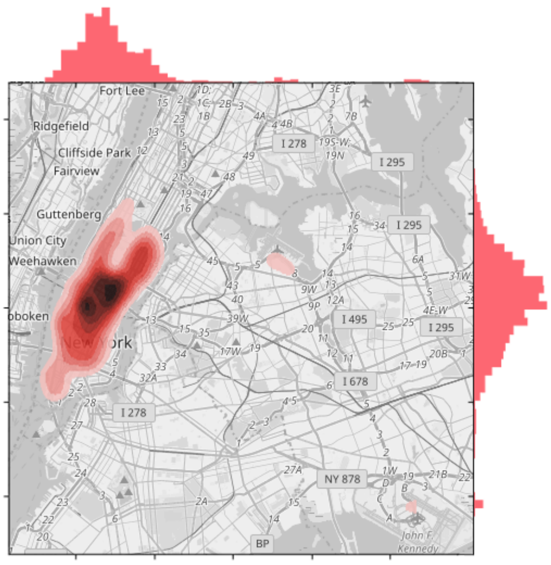

Figure 3: Heatmap of the trip locations on a map (credits: OpenStreetMap).

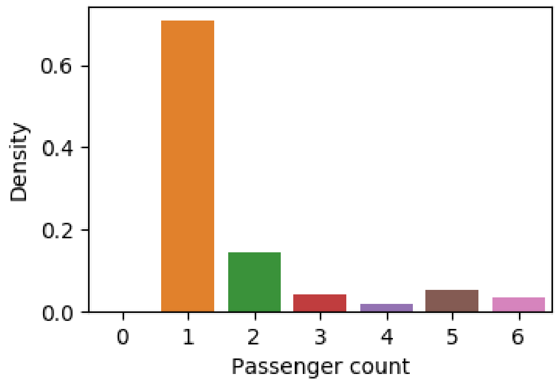

Figure 4: Distribution of the number of passengers.

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. Pick-up month, minute and seconds are not shown, as their graphs are uniformly distributed and therefore not visually informative. Outliers have been removed in order to properly display the pick-up and drop-off location distribution, as well as the trip duration. These outliers will be discussed in the next section. One can see in figure 1 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 two city airports.

It is also important to make sure that the training and test sets are independently and identically distributed. For this, the above distributions have been compared with those from the test set. If the distribution from the training and testing set significantly overlap, then it can be supposed 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, some outliers can be found in the distributions of the GPS coordinates and the trip durations. The mean value for the trip duration

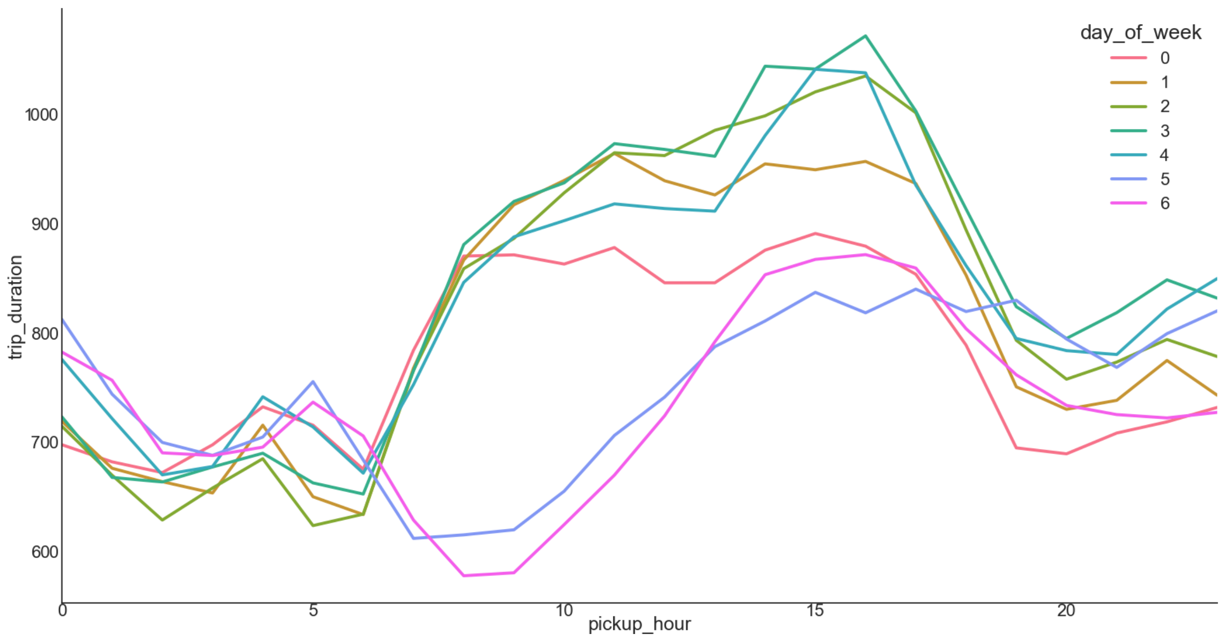

Figure 5: Distribution of the pick-up time, for different days of the week.

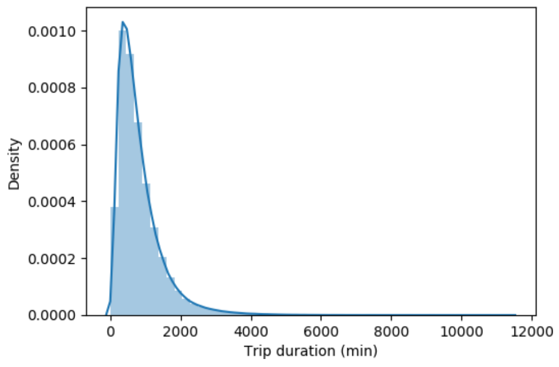

Figure 6: Distribution of the trip duration, in seconds.

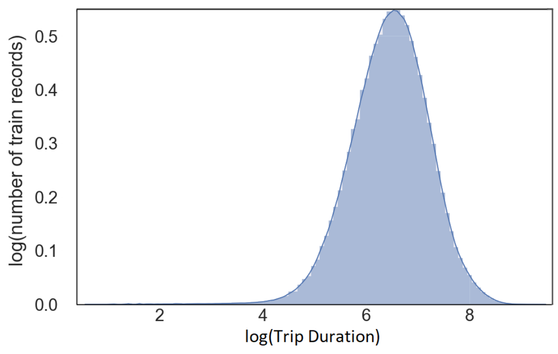

Figure 7: Distribution of the logarithm of the trip duration, in seconds.

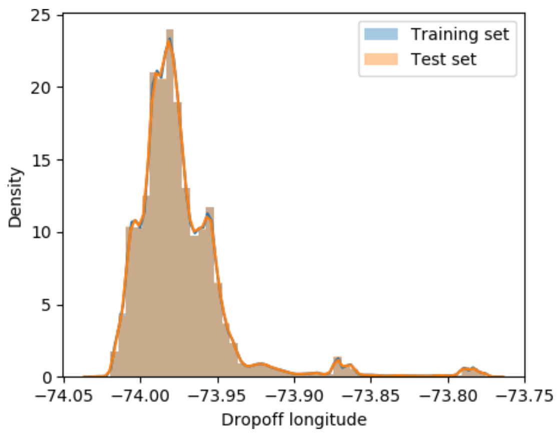

Figure 8: Drop-off longitude in the training and test set.

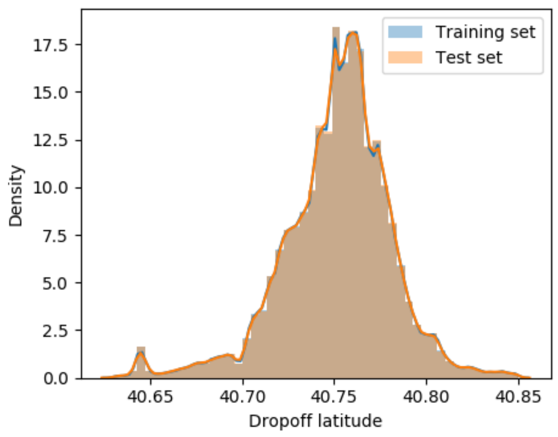

Figure 9: Drop-off latitude in the training and test set.

(961 seconds) and the standard deviation (5247 seconds) have been calculated, and all trips which were longer than the mean plus twice the standard deviation have been removed. Futhemore, all trips which were located far outside of NYC were removed, as it would probably the case for a realistic taxi service. The geographic bounds are $[-74.03; -73.77]$ for the longitude and $[40.63; 40.85]$ for the latitude.

To augment our dataset, the weather data of NYC in 2016 has been downloaded [1] and the current volume of precipitation has been added to each trip. Next, the GPS coordinates of the pick-up and drop-off locations have been used to calculate following properties:

- Distance between the two points curved along the surface of the earth called, Vincenty's distance;
- Pythagorean distance between the two points;
- Manhattan distance;
- The angle between the two points in relation to the north pole, called bearing.

OpenStreetMap Route Machine (OSRM, `http://project-osrm.org/`) has been used to add further information about the taxi ride. OSRM 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, the number of turns to make before reaching the destination has been included, as well as how many intersections one has to pass. Apart from the data provided by Kaggle it was possible to use 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 it was decided not to include them.

Additionally to the data augmentation, each sample is shifted and scaled, so that each column has zero mean and unit variance, as required for most of the models that have been used. This functionality is easily provided by the SKlearn library.

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 Sklearn package recursive feature elimination (RFE) on top of a random forest model. 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 pick-up locations as well as different measures of distance were ranked as very important. In contrast, features that did not contain much information such as pick-up 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, did not rank very low leading to believe that it did not 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 using XGBoost in figure 10 provides a different perspective on which features provide the most information to the model. This information implies the relative contribution of the corresponding feature to the model calculated - calculated by taking the average information gain produced by each feature on each split of the constructed trees. It is expected that the distance features 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 add significantly more value to the model than previous analysis. 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 consistent with the scoring of the Kaggle competition, the same evaluation function

[1]from IEM Computed Hourly Precipitation Totals by Iowa State University (`https://mesonet.agron.iastate.edu/request/asos/hourlyprecip.phtml?network=NY_ASOS`)

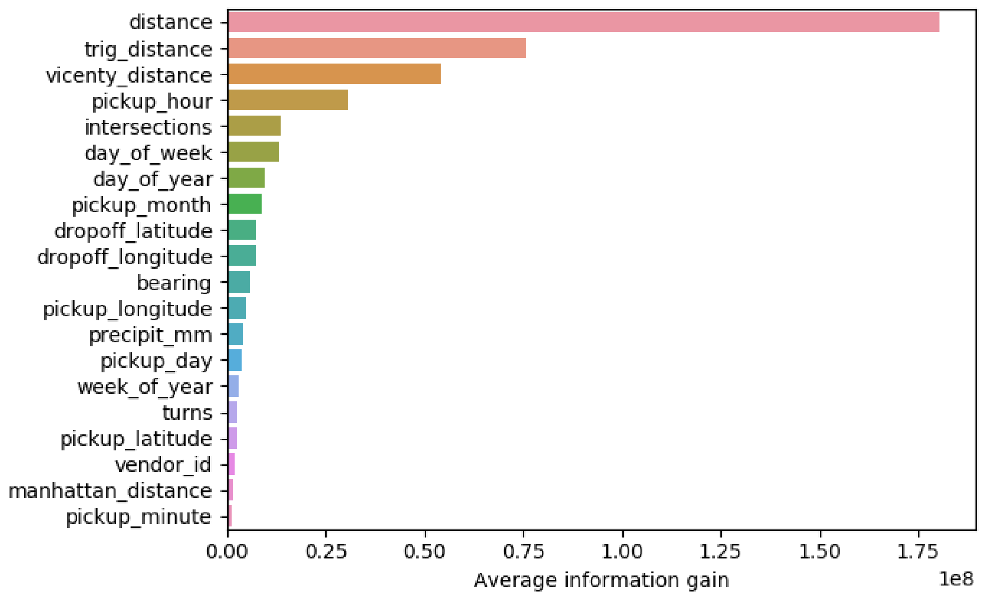

Figure 10: Average information gain of each feature in the XGBoost model.

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 as

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

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 unrealistic to predict a taxi journey to the exact second due to the dynamic environment of the city.

6 Results

As previously mentioned, the results of each model will be presented in terms of their RMSLE score and the predictions in minutes through classification. The final optimised results of all models can observed and compared in table 1. Each model was initially tested by running with the default parameters and it became evident that neural networks, random forest and XGBoost performed much better than the remaining models. Those three models were extensively tested and optimised with all data available in appendix B. A grid search was still applied to the less performant models but no further time was invested in optimising them since it was evident they would not outperform the top three models and hence the lack of testing results in the appendix.

6.1 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 (see appendix B) and removing features suggested by the feature selection elimination, the RMSLE was reduced to 0.346, making XGBoost perform slightly better than NN and becoming the best tested model. l_1 and l_2 regularisation terms were also introduced in an attempt to further improve the prediction accuracy but regardless of the values used, no effect could be observed. While training the model, an internal parameter controls which metric is used to calculate the evaluation for validation data.

Unfortunately, no customised function can be passed so the RMSE was used but it is believed that if this parameter was customisable, using RMSLE as an error metric would marginally improve the results.

6.2 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.

6.3 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.

6.4 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, a SVM regressor has been trained on a random subset of the training set, which would contain up to 20,000 random samples. Past this value, the training time is simply unbearable.

On the other hand, since the evaluation of a trained SVM model is fast, it was possible to validate the 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, a SVM model achieves a RSMLE of 0.359. By using more powerful hardware, it would certainly be possible to achieve a better score by using more training samples, in order to reduce the variance of the learned estimator, and thus its generalisation error.

6.5 Stochastic Gradient Descent

Using the Stochastic Gradient Descent regressor from sklearn, an RMSLE of 0.487 has been obtained without any hyper-parameter tuning. Using grid search however it was possible to achieve an RMSLE of around 0.455. While the features are scaled 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.

6.6 Gaussian Process

Just as the support vector machine, the Gaussian Process (GP) has a time complexity of $O(n^3)$. Therefore, it was unbearable to train a GP model with the whole dataset. After a grid search to find the optimal hyper-parameters, a GP model has been trained with a rational quadratic kernel, and a noise reduction parameter $\alpha = 0.01$, on 1000 random samples and then validated on 100,000 other samples. The RSMLE was calculated to be 0.469, which is most probably due to underfitting, due to the very limited number of training samples.

Model	RMSLE regression	RMSLE classification
XGBoost	0.346	0.286
Neural Network	0.347	0.136
Random Forest	0.350	0.312
SVM	0.359	NA
SGD	0.475	0.400
Gaussian Process	0.512	0.500
Ridge Regression	0.539	NA
Dummy Regressor	0.810	0.802
AdaBoost	0.990	0.800

Table 1: Comparison of the RMLSE of all tried models.

7 Conclusion

As our first real team machine learning project, we tackled this introductory Kaggle competition with a lot of curiosity. Having enriched the provided data significantly with help from OSRM we managed to establish a decent model with Extreme Gradient Boosting, which ranks us 586th (top 47%) on the Kaggle competition leaderboard. While we did improve the data set we could have gone further by including the available files on the NYC TLCs website the increase the number of samples, just as using a sophisticated way of linearizing GPS coordinates like one-hot encoding could have helped the improve the predictions. This could be studied in a next Kaggle competition using this kind of data. For the time being, we can be be happy with what we learned in the process and look forward to our next challenges as machine learners.

A Appendix

A.1 Confusion Matrix

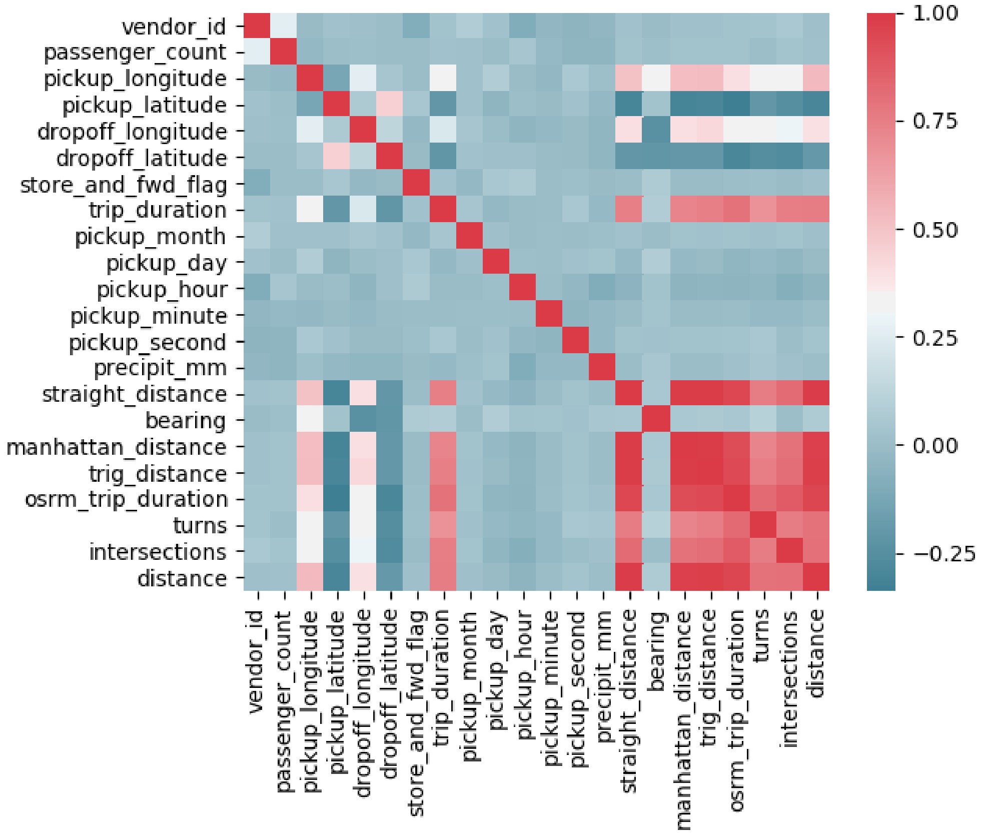

Figure 11: Confusion matrix of the extended dataset.

B Detailled Grid Search results

B.1 XGBoost Grid Search Results

parameters	RMSLE regression
default	0.3940
num-boost-round = 100	0.3869
early-stopping-round = 5	0.3878
early-stopping-round = 10	0.3896
early-stopping-round = 3	0.3877
early-stopping-round = 1	0.3875
model = gblinear	0.4877
model = dart	0.3881
depth = 3	0.4045
depth = 6	0.3869
depth = 10	0.3804
depth = 20	0.4108
depth = 15	0.3900
depth = 9	0.3764
depth = 8	0.3813
eta = 0.3	0.3672
eta = 0.4	0.3711
eta = 0.2	0.3640
eta = 0.1	0.3614
eta = 0.05	0.3597
eta = 0.001	0.5396
eta = 0.01	0.3592
alpha = 0.5	0.3701
alpha = 1.5	0.3674
alpha = 0.25	0.3667
alpha = 1.1	0.3688
alpha = 1.2	0.3662
subsample = 0.7	0.3732
subsample = 0.95	0.3680
subsample = 0.85	0.3692
min-child-weight = 5	0.3637
min-child-weight = 10	0.3603
min-child-weight = 8	0.3599
min-child-weight = 2	0.3627
min-child-weight = 9	0.3588
l1-gamma = 5	0.3588
l1-gamma = 100	0.3588
l2-alpha = 1	0.3607
l2-alpha = 1	0.3600
l2-alpha = 1	0.3599

Table 2: XGBoost grid search results.

B.2 Random Forest Grid Search Results

parameters	RMSLE regression
default	0.3676
n-estimators = 100, max-depth = +inf	0.358
n-estimators = 125, max-depth = 10	0.362
n-estimators = 130, max-depth = 20	0.361
n-estimators = 125, max-depth = none	0.350
max-features = 10, min-samples-leaf/split = 3	0.3483
20 out of 24 features	0.3473
15 out of 24 features	0.3496
17 out of 24 features	0.3485

Table 3: Random forest grid search results.

B.3 Neural Networks Grid Search Results

Parameters	RMSLE regression
default (2 layers of 512 nodes)	0.3691
3 layers of 512 nodes	0.3614
3 layers of 256 nodes	0.3640
3 layers of 64 nodes	0.3492
3 layers of 32 nodes	0.3560
3 layers of 64, 32 and 16 nodes	0.3562
3 layers of 64, 64 and 32 nodes	0.3745
3 layers of 64, 128 and 64 nodes	0.3509
3 layers of 64, 256 and 64 nodes	0.3550
The following entries are for 3 layers of 64 nodes	
relu as activation function	0.3511
tanh as activation function	0.3552
relu, alpha =1	0.3465
alpha = 10, 200 iterations	0.3466
alpha = 10, 300 iterations	0.3505
alpha = 100, no early stopping	0.3486
tol = 0.00001	0.3512
batch size = 100	0.3486
batch size = 200	0.3586
batch size = 500	0.3538
batch size = 1000	0.3644
batch size = 10000	0.3651

Table 4: Neural Networks grid search results.