
# Project 3 - Taxi Challenge - Report

## Applied Inductive Learning

---

**Alexandre Danthine** (S113236)
**Fabrice Servais** (S111093)

December 14, 2015

# 1 Introduction

We will introduce our different models chronologically with their leaderboard scores. After that we will explain all the changes that we try to implement in our model.

# 2 Simple reference submission

First, we tried the simplest model we could have made in order to have a reference score to compare with.

## 2.1 Preprocessing

*The preprocessing introduced here is used on all our other models since it is the minimum processing to make the data usable.*

The first processing is to replace non-numeric values of some features into numeric values, as shown in `Table` 1.

| Feature(s) | Non-numeric value | Numeric value |
|---|---|---|
| `CALL_TYPE`, `DAY_TYPE` | A | 0 |
| | B | 1 |
| | C | 2 |
| `MISSING_DATA` | True | 0 |
| | False | 1 |

Table 1: Conversion of non-numerical values into numerical values

We then have to change all missing values in `ORIGIN_CALL` and `ORIGIN_STAND`. We decided to put for each missing feature the mean value of the corresponding feature. The corresponding values may not have meaning but they are supposed to be ignored by the model.

The last step is to get $X$ and $y$. We considered that the destination was the last coordinate of `POLYLINE` and $X$ was the rest of the path. Paths of length 0 were ignored and for the paths of length 1, the origin is the destination. Moreover, we decided to remove all the rows that have a missing data since we noticed that there are a small number of them regarding the total size of the learning sample.

## 2.2 Training

We used `KNeighborsRegressor` with $K = 5$ as regressor, with the default parameters. About the features, we test with only the last coordinate of the partial trajectory with and without the others features.

## 2.3 Results

The results are shown in Table 2.

We see that the features leads us to think that the features induces error. We made approximation by taking the mean of parameter and we did not perform features selection.

| | Public leaderboard score | Private leaderboard score |
|---|---|---|
| Only the last coordinate | 0.01924 | 0.02094 |
| Last coordinate + features | 0.02053 | 0.02095 |

Table 2: Simple reference scores

# 3 First 'expand' model

This model was a trial of an idea that we used in all our other models.

## 3.1 Preprocessing

The idea behind this model is that we had to solve a main problem: how could we fit a variable number of features in input into a model with a defined number of features? The first naive, 'brute force' solution was to increase the length of all the paths so that they would have the same length, which is the length of the longest path.

We made the function `expand_list(l, final_size)` which takes a list `l` and expand the list by copying its values so that it returns a list of size `final_size`. For example, `expand_list([1, 2, 3], 8)` gives `[1, 1, 1, 2, 2, 2, 3, 3]`.

## 3.2 Training

We thus applied the same preprocessing as the first model, the method explained above and the `5-Nearest Neighbors` regressor. We used the expanded `POLYLINE` feature to predict the destination.

## 3.3 Result

The results are shown in Table 3.

| Public leaderboard score | Private leaderboard score |
|---|---|
| 0.02008 | 0.02078 |

Table 3: First 'expand' model scores

If we compare with the previous model we see that the score decreases a bit so we taught that we was on the good way to perform a good prediction.

# 4 New reference model

This model served as a base for future models, which are thus variants of this one.

## 4.1 Preprocessing

By looking at the distribution of the occurrences of the lengths, as in Figure 1, we can see that most of the lengths are situated before 200. So we came with the idea of having multiple regressors depending of the length of the path, i.e. we used 4 different regressors depending on if the length

of the path is smaller than 25, smaller than 50, smaller then 100 or bigger than 100.

To know in which regressor we are, we will predict the length of the path with the origin of the partial path and the others features.

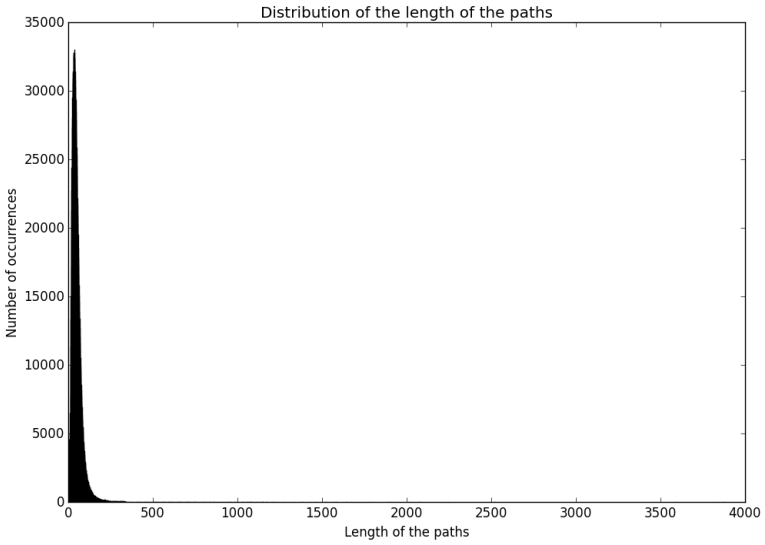


Figure 1: Distribution of the occurrence of the length of the path

## 4.2 Training

In this case, we used a `DecisionTree` regressor. We used the expanded `POLYLINE` as feature to predict the destination and all the others given features (with the origins) to predict the length. The `TIMESTAMP` feature was also transformed to fit in 4 categories: between midnight and 6AM, 6AM and noon, noon and 5PM and 5PM until midnight.

For another case, we removed `ORIGIN_CALL`, `ORIGIN_STAND` and `TAXI_ID` as we thought that using the mean of the feature to replace the missing values will induces error. The values of `TIMESTAMP` were here replaced by the corresponding hour, which is more general than the timestamp and more specific than a time slot.

## 4.3 Results

The results are shown in Table 4.

| | Public leaderboard score | Private leaderboard score |
|---|---|---|
| Path expand + all features | 0.01747 | 0.02080 |
| Path expand + all features except `ORIGIN_CALL`, `ORIGIN_STAND` and `TAXI_ID` | 0.01725 | 0.02072 |

Table 4: Reference scores

These results were quite good as we saw on the public leaderboard, so we thought that it was a good way to start for our further analysis on the regressors to use this configuration.

Afterwards, we saw that our results were still not good at all for our *'reference'*.

# 5 Separate regressors by `CALL_TYPE`

## 5.1 Training

We used a `DecisionTree` regressor. We separated the data with their `CALL_TYPE` and we made 3 regressors by type of length. We did that because we thought that in this way the features `ORIGIN_CALL` and `ORIGIN_STAND` could have more impact on the model.

## 5.2 Results

The results are shown in Table 5.

| Public leaderboard score | Private leaderboard score |
|---|---|
| 0.01757 | 0.02097 |

Table 5: Separate `CALL_TYPE` model scores

As the results were more or less the same score as before. We tripled the number of regressors in this model so we decide to drop this model.

# 6 Variants

## 6.1 Methodology

To test the effect of the different methods and parameters, we used cross-validation. 500.000 lines were read from the train file and we used 3-Fold to have a train set and a testing set to work on. This could have been set to more, however, we needed less computation time for our tests.

To predict the length, we measured the length of the `POLYLINE` and after that we made our cross-validation.

To predict the destination, we took the `POLYLINE` on the `train_data` and after that, to create the test set, we removed a random number of position to find a partial trajectory as in the real test.

We know that this random part can bring variance in our test error but as we do cross-validation we think that this problem is largely reduce.

## 6.2 Variants on our length regressor

### 6.2.1 Features selection

We tried to do a feature selection. We did a feature selection embedded in the regressor as the `DecisionTreeRegressor()` gives us a way to know the importance of the features represented as weights.

| | CALL_TYPE | TAXI_ID | DAY_TYPE | TIMESTAMP | orgx | orgy |
|---|---|---|---|---|---|---|
| **Weight features** | 0.0384 | 0.1170 | 0.0 | 0.0991 | 0.4446 | 0.3006 |

Table 6: Features selection

We tried to remove the less weighted features to see the impact on the error of the prediction of the length.

| **Feature removed** | bias | Variance | Mean absolute error |
|---|---|---|---|
| None | -0.06594 | 344.29765 | 22.51414 |
| DAY_TYPE | -0.00783 | 202.09584 | 22.36994 |
| DAY_TYPE+CALL_TYPE | 0.012228 | 154.30441 | 22.81543 |

Table 7: Feature selection

We saw that we decreased the error by removing the first less weighted feature but it is not the case for the second one. We decided to drop DAY_TYPE and CALL_TYPE as the second one decreases also much the variance.

#### 6.2.2 Change of the TAXI_ID feature

As there is a fixed number of taxis in the city, we decided to try to put a binary feature by TAXI_ID instead of a large number as it is given in the data.

| **Type of taxi feature** | bias | Variance | Mean absolute error |
|---|---|---|---|
| TAXI_ID | -1.1454 | 2663.155 | 32.807 |
| binary TAXI_ID | -0.00783 | 202.09584 | 22.36994 |

Table 8: Feature selection

We see that the mean absolute error decreases much. But we added a huge amount of features to fit in our model. This way of preprocess the data increases a lot the running time of our model, so we decided to drop this idea. We could implement this change of the TAXI_ID in the destination regressors but that would also take too much time.[1]

### 6.3 Variants on our destination regressors

#### 6.3.1 Study of the max depths of the decision tree

| **Max depths** | bias | Variance | Mean absolute error |
|---|---|---|---|
| **5 (Reference)** | 0.0005034 | 0.0008954 | 0.0130758 |
| **10** | 0.0004996 | 0.0009563 | 0.01300926 |
| **20** | 0.0004888 | 0.0009274 | 0.0130165 |
| **50** | 0.0004886 | 0.0009511 | 0.0130272 |

Table 9: Study of the max depths of the decision tree

We see that the mean absolute error does not decrease at all. It increases a bit.

[1] We tested for a small number of data and that lead to bad results on the leaderboard.

We still decided to make a test on "Kaggle" to see if the behavior tested was the same on the public test set.

| Public leaderboard score | Private leaderboard score |
|---|---|
| 0.01815 | 0.02109 |

Table 10: Max depths = 40

We clearly saw that the score does not decrease so we drop the fact of changing the maximum depths in our decision tree.

#### 6.3.2 Separate the prediction of the longitude and the latitude

*The reference is not all the time the same as we tested all these tests on different day. So we changed the number of data taken or others parameters. The goal of this test is to see the trend of the error.*

| | bias | Variance | Mean absolute error |
|---|---|---|---|
| **(Reference)** | 0.0004814 | 0.000894196 | 0.0126402 |
| **Separate lat/long** | 0.000496227 | 0.000892860 | 0.0125918 |

Table 11: Separate the latitude and the longitude

As the mean total error decreases with the separation of the two coordinates, we submitted to "kaggle" with this way of doing.

| Public leaderboard score | Private leaderboard score |
|---|---|
| 0.01950 | 0.02059 |

Table 12: Separate the latitude and longitude

We see that the score increases, we dropped this separation. We were motivate to do that because not all the regressors can have multiple output.

#### 6.3.3 Add the direction in the features

We decided to create a new feature for the model. We created the direction of the car in degrees. To do that, we took the first and the last coordinate of the `POLYLINE` and we computed the angle with a horizontal reference.

That seems to be quite a good idea as it will more characterize the path. So we submitted a test on Kaggle and the score was higher than our reference so we decided to drop the `DIRECTION`.

| Public leaderboard score | Private leaderboard score |
|---|---|
| 0.01768 | 0.02068 |

Table 13: Include the direction of the path

#### 6.3.4 K-Nearest Neighbors

We used K-Nearest Neighbors instead of decision tree. We made a cross validation to know which $k$ was the best and we found $k = 51$.

So we tried with this parameter for our prediction.

| Public leaderboard score | Private leaderboard score |
|---|---|
| 0.01743 | 0.02065 |

Table 14: 51-NN

So we decided to not use the kNN regressor.

# 7 Conclusion

As we saw in this report, we found a good score for the public leaderboard for our model but that led to really poor results in the private leaderboard.

We had a lot of problems to run the model in reasonable time. In fact it tooks often more than 2 hours to do the computations. We made a 'pickle' file to avoid the re-preprocessing of all the data at every run, but we did that the last week so we lost a lot of time running that same code. We know that we could have put less data in input but everytime we did that, it led to a horrible score on the public leaderboard.

The fact that we had a good solution without tuning some parameters to reach a better score leads us to think that our way to write the model was a good choice. So we took this as a reference and we tried to find a better score by making some variants of it. In fact we should have tested more ways to predict the destination before thinking to improve the model with different kind of regressors, feature selection,...