ELEN0062 - Introduction to Machine Learning
Project 3 - Competition

ALEXIA DONATI (S200742) and LEI YANG (S201670)

1 PRE-PROCESSING

First of all, we decided on dropping trips with missing data and trips with an empty trajectory since these two situations wouldn't help us in our prediction and are fairly uncommon, as shown in table 1 and 2. Following that, since the MISSING_DATA feature now has a unique value, False, we can drop it completely.

We also decided on dropping the attribute DAY_TYPE since it has only one unique value, 'A', for all trips, as shown in table 3.

To make it easier to use, we also decided on replacing the values of CALL_TYPE, 'A', 'B', or 'C', with a numerical representation, 0, 1, or 2.

Then, we looked at the situation of the trips with duplicate TRIP_ID and we found that they all also share the same TAXI_ID and TIMESTAMP. Because of that we decided on only keeping the first occurrence of such trip.

MISSING_DATA	
False	1.710.660
True	10

Table 1. Value count for MISSING_DATA

POLYLINE	
[..]	1.704.769
[]	5.901

Table 2. Value count for POLYLINE

DAY_TYPE	
unique	1
array	['A']

Table 3. Unique of DAY_TYPE

Then, we realized that there are many null values in the ORIGIN_CALL and ORIGIN_STAND features, as shown in figure 1. Because of their high frequency, we can't delete all rows with null values. This is why we decided to replace the null values with 0.

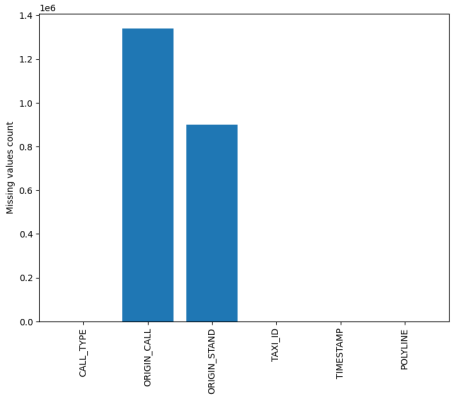

Fig. 1. null values count for each feature

Then, we decided to replace the TIMESTAMP feature with two features that we can extract from it, the day of the week and the hour the trip was started on.

These new, more precise, features allow us to group and make connections between the trips more easily. For example, we can see from figure 3 that the peak hours for taxi trips correspond to the start of a typical work or school day. This could indicate that the destination of a taxi trip has some correlation to the time it was started. The same could be said for the day of the week the taxi trip was started. We can see from figure 2 that the peak day for taxi trips is Friday. We can also argue from the intuition that a taxi trip on the weekend might not have the same goal as a taxi trip on a working weekday.

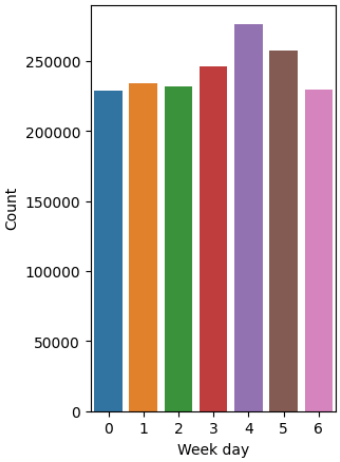

Fig. 2. Count of trips per week day

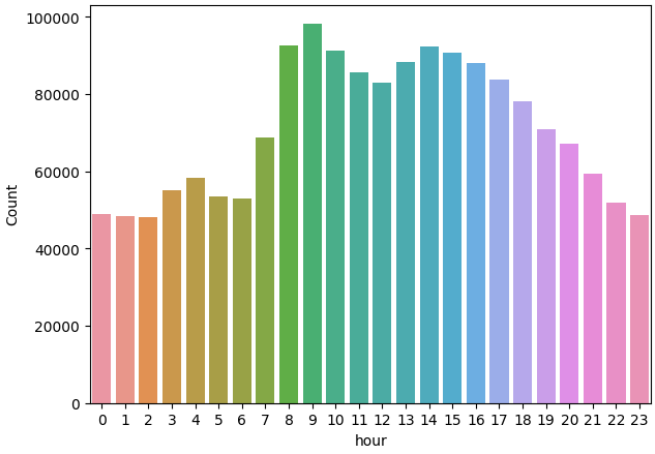

Fig. 3. Count of trips per hour

We also considered extracting the total length and duration of the trip from the POLYLINE feature but we decided against it since the trajectories of the test data are cut off and we fear that using them will lead to inaccuracies or inconsistencies.

This leaves us with 7 features, 1 which is the trip itself, and 6 which are metadata of the trip.

Feature	CALL_TYPE	ORIGIN_CALL	ORIGIN_STAND	TAXI_ID	POLYLINE	WEEKDAY	HOUR
Type	Integer	Integer	Integer	Integer	List as a string	Integer	Integer
Values	{1, 2, 3}					{1,...,7}	{1,...,24}

Table 4. Features of the data after pre-processing

For the training data, we also extracted the values END_Long and END_Lat, which represents the destinations of the trip, from its POLYLINE feature to form our training targets.

2 ASSESSMENT OF THE MODELS

Firstly, we divided the given training data into a training set and a testing set with a 9-1 ratio. When selecting the best parameters and training the model we only used the training set.

The, to evaluate the performance of our models with different parameters we used the k-fold cross-validation technique with 10 folds over the training set. It helped us obtain a more accurate estimate of a model performance by having each fold of the training set be a test set once. We then used the average mean haversine distance of the predictions on each test fold iteration as our evaluation of the model.

Lastly, to assess the model, and later be able to compare them to each other, we used the mean haversine distance of the predictions on the testing set which stayed unseen to the model during its training.

After all of that, to make predictions on the given test data and to write our submission we re-trained the chosen model on the full training data.

3 APPROACHES WE INVESTIGATED

3.1 Non-Sequential

We first tried to implement many of the approaches we saw in the course by replacing the POLYLINE feature by two underlying features, START_Long and START_Lat, which corresponds to the coordinate of the starting point of the trip.

(1) Decision Tree: For this model, we need to find the best value for the hyper-parameter max_depth, the maximal depth that we want to consider when predicting a new data point. We need to be careful when choosing that hyper-parameter to avoid underfitting or overfitting. We used the k-fold cross-validation technique mentioned in section 2 and we reported the results we obtained for different values of max_depth on the figure 4. We found that the max_depth with minimum mean haversine distance is 9 with a mean of 3.669. We can also clearly see from the graph that after that point the model starts to overfit the training set and arrives at a maximum around max_depth 30 which probably corresponds to a fully grown tree on the training set.
The haversine score of the test data predictions resulting from the model using a max_depth of 9 can be found in the table 5 below.

Public	3.67857389
Private	3.61272713

Table 5. Haversine score of the test data predictions

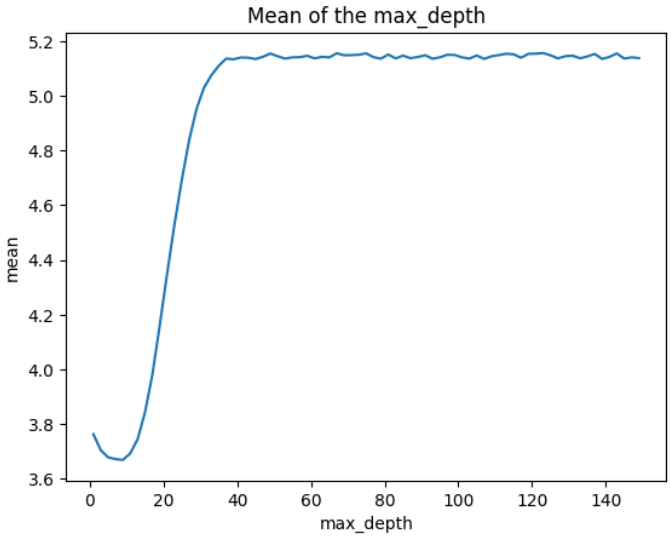

Fig. 4. Evolution of the haversine score with the max_depth parameter

(2) K-Nearest Neighbours: For this model, we need to decide on k, the number of nearest neighbors that we want to consider when predicting a new data point. We need to be careful when choosing that hyper-parameter to avoid underfitting or overfitting. We used the k-fold cross-validation technique mentioned in section 2 and we reported the results we obtained for different values of k on the figure 5. We found that the k with the minimum mean haversine distance is 66 with a mean of 3.6478. We can see from the graph that the model is underfitting when the value of k is under 20.
The mean haversine distance over our test set predictions resulting from the model using a k of 66 is 3.6286 and the haversine score of the test data predictions resulting from that model can be found in the table 6 below.

Public	3.53701552
Private	3.63792529

Table 6. Haversine score of the test data predictions

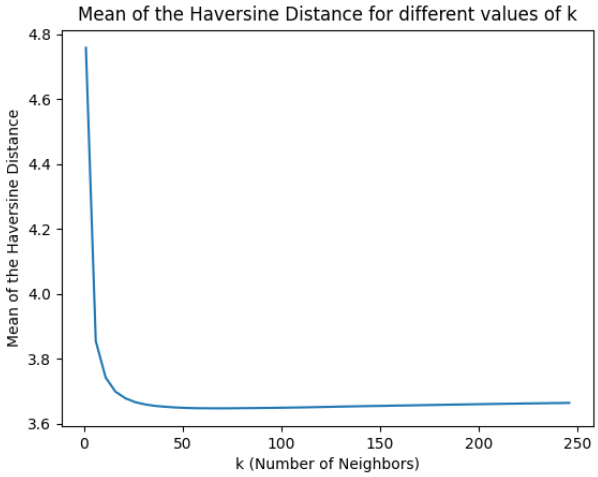

Fig. 5. Evolution of the haversine score with the k parameter

(3) Ridge: For this model, we need to optimize the formula $||y - Xw||_2^2 + alpha * ||w||_2^2$. We need to choose the appropriate value for alpha to minimize the mean of the haversine distance. We used the k-fold cross-validation technique mentioned in section 2 and we reported the results we obtained for different values of alpha on the figure 6. We found that the best value for alpha is 2.5 with a mean of 3.67
The haversine score of the test data predictions resulting from that model can be found in the table 7 below.

Public	3.74035384
Private	3.63206021

Table 7. Haversine score of the test data predictions

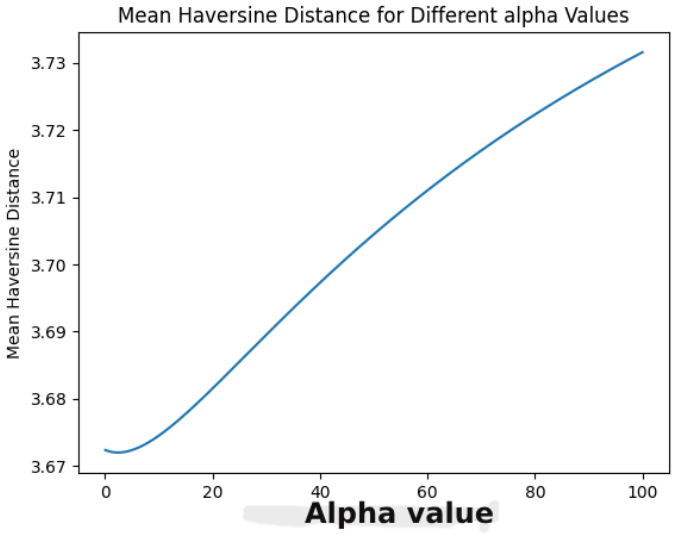

Fig. 6. Evolution of the haversine score with the alpha parameter

(4) Lasso: For this model, we need to optimize the formula $(1/(2 * n\_samples)) * ||y - Xw||_2^2 + alpha * ||w||_1$. Just like with the ridge formula, we need to choose the appropriate value for alpha to minimize the mean of the haversine distance. We used the k-fold cross-validation technique mentioned in section 2 and we reported the results we obtained for different values of alpha on the figure 7. We found that the best value for alpha is 2.5 with a mean of 3.82.
The haversine score of the test data predictions resulting from that model can be found in the table 8 below.

Public	3.70666575
Private	3.63495641

Table 8. Haversine score of the test data predictions

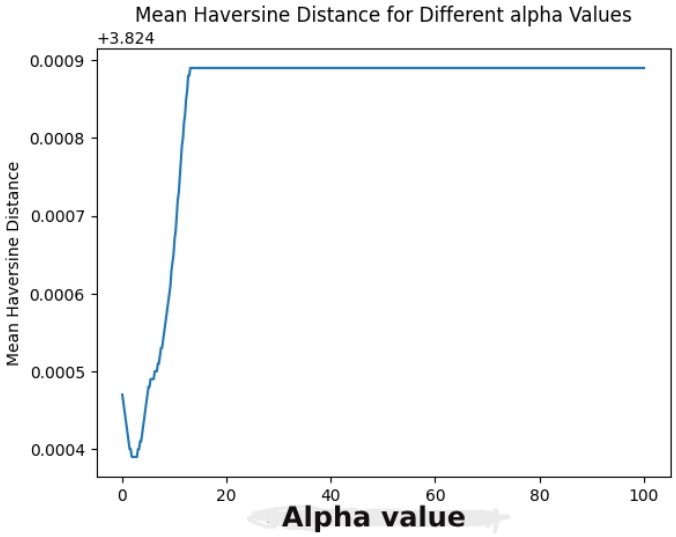

Fig. 7. Evolution of the haversine score with the alpha parameter

We found that the alpha value which minimize the mean of the haversine distance is the same for both the Ridge and Lasso approach. This can be explained by the fact that the two algorithms are very similar. Their difference lies in the fact that the Ridge method is used to reduce overfitting while the Lasso method is used to perform feature selection.

(5) Neural-Network: For this model, we first had to choose which activation function to use. There exist many possibilities, such as relu, logistic, or tanh. We decided on using logistic activation. Then, we had to decide on the size of the hidden layer. We used the k-fold cross-validation technique mentioned in section 2 and we reported the results we obtained for different values of alpha on the figure 8. We found that the most optimal layer size is (30, 30) with a mean of 3.822.
The haversine score of the test data predictions resulting from that model can be found in the table 9 below.

Public	3.71221870
Private	3.62845755

Table 9. Haversine score of the test data predictions

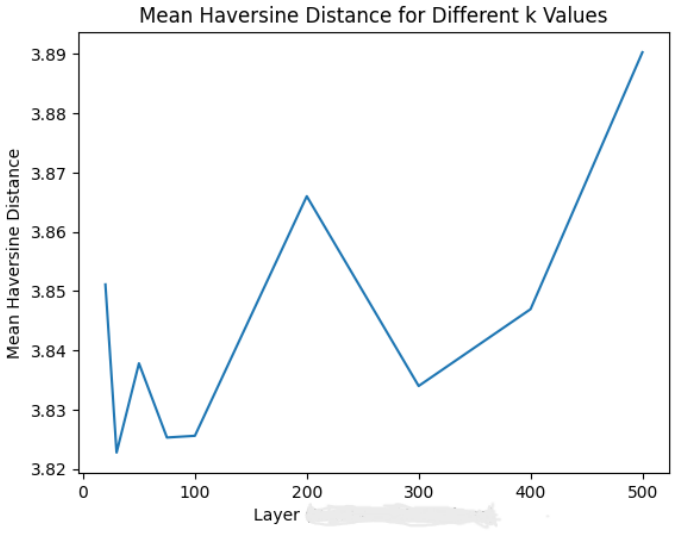

Fig. 8. Evolution of the haversine score with the number of layer

With this method, we observed a major problem, the coordinates of the destination predicted by the model is the same for all input sample.

(6) Stacking Ensemble: For the last model in this non-sequential section, we tried to combine all the advantages of the models we implemented before to create meta-data to predict an outcome for our problem. We used these 5 algorithms to compute the first part and used another model, RandomForest, for the final prediction. We had to divide our training set into 2 sets, the first part for the stacking model and the second part for the final prediction. We had to add some features to the new X for the different outputs in the first part of the stacking process. The average haversine mean we obtained by using this technique is 4.00. It is bigger than the mean of any of the previous model used alone, but we can assume that this model will avoid overfitting.
The haversine score of the test data predictions resulting from that model can be found in the table 10 below.

Public	4.40019364
Private	4.20490545

Table 10. Haversine score of the test data predictions

We think we can attribute the subpar score we obtained in this section to the fact that we only consider the starting coordinate of each trip.

3.2 Sequential

Another approach we considered was using an RNN since the input data has an important sequential and temporal nature. The approach we implemented was using the 5 first coordinates of the POLYLINE feature as input to an LSTM.

For the pre-processing of the polyline sequences, we padded the sequences that were smaller than 5 coordinates with starting 0 and we scaled the longitude and latitude of all sequences.

We used the Mean Squared Error as our loss function and trained our model on 20 epochs with callbacks for checkpoint and early stopping.

To prevent overfitting we added a dropout layer between the LSTM layer and the output layer. This means that, during each training iteration, a set number of units of the output of the LSTM layer will be randomly set to zero.

We also added a learning rate schedule to dynamically adjust the learning rate during the training process. This can accelerate convergence in the early stages of training when larger steps may be beneficial, and reduce the learning rate later when fine-tuning is required.

We tried multiple combinations of dropout rate and learning rate schedules. The results of those tests can be found in the table 11 below with the epoch at which the best model was saved by the checkpoint callback.

	0	0.1	0.2
None	2.526 on the 10th epoch	1.151 on the 6th epoch	1.059 on the 4th epoch
$\times$ 0.1 after 5 epoch	3.374 on the 9th epoch	0.188 on the 12th epoch	0.337 on the 11th epoch

Table 11. Mean haversine distance of our test set predictions with different fine-tuning parameters

Those results need to be approached carefully as every sample of the provided training data has its complete trajectory in its polyline feature but the samples in the testing data of the challenge have incomplete trajectory. However, we do still think that these results can be useful when comparing the different versions of the model to each other.

We can from see from the table 11 above that the model with a dropout rate of 0.5 and a learning rate schedule which is at 0.01 for the 5 first epoch but then is multiplied by 0.1 for each following epoch is the most adapted to our problem. The haversine score of the test data predictions resulting from that model can be found in the table 13 below.

Public	2.77653905
Private	3.15537859

Table 12. Haversine score of the test data predictions

We also tried training this same model with polyline sequences of length 10 but the results were really similar to the previous version with a mean haversine distance of our test set predictions of 0.346. The haversine score of the test data predictions resulting from that model can be found in the table 13 below.

Public	2.89989943
Private	3.13413488

Table 13. Haversine score of the test data predictions

4 SUMMARY OF THE RESULTS OF OUR DIFFERENT APPROACHES

We gathered all of the results we obtained from the different model we trained for different approaches in the table 14 below. From it, we can see that, except for the stacking approach, most non-sequential approaches gave us similar results. On the other hand, we can see that the sequential LSTM approach has significantly better result.

	Decision Tree	KNN	Ridge	Lasso	Neural Network	Stacking	LSTM
Public	3.67857389	3.53701552	3.74035384	3.71221870	3.74650223	4.40019364	2.77653905
Private	3.61272713	3.63792529	3.63206021	3.63495641	3.62845755	4.20490545	3.15537859

Table 14. Haversine score on the test data predictions for our different approaches