Case 3

Improving last mile logistics with Machine Learning

Applied Machine Learning and Optimization
17th June 2022
Team 4
Marc Fernández i (U159912)
Fernando Cámara (U162271)
Carla Torruella (U15988)

Table of contents

Executive summary

There are three main questions related to the engine-off time that the company faces:
(i) Is the actual engine-off time organization optimal, using a fixed time of three minutes, or should Beanie Limited change this engine-off time estimation into a new one in order to have more accurate schedules? (ii) Can machine learning help with this problem? (iii) How can it be applied to this specific case?

These questions arise from the need to optimize the engine-off time, the interval that goes through since the delivery man turns off the truck engine until he completes the delivery and turns off the engine of the truck again. As can be seen in the report it is possible to reduce this problem significantly with machine learning:

(i) With the information that has been provided we can do an exploratory data analysis of the different variables, regardless of whether they are categorical or numerical. First of all data must be checked to see if it's correct and complete, after that, you have to determine which is the variable on which the model is going to be based and how it will be measured.This variable will be the engine-off time in seconds which will be called “final_time”.
Then you have to check whether the other variables have correlation with the independent variable or not and measure it to determine if the engine off time prediction can be improved or not.

(ii) The fact that there are variables that have correlation with the independent variable means that we can estimate a model which is a better estimator of the real engine-off time.
The categorical variables that will be considered are :”truck_size”, “driver_id”, “floor”and “is_fresh_client”. On the other hand, the numerical variables are:” total_weight”, “box_count” and “delivery_timestamp”.

(iii) Since there are variables that can explain the engine-off time and that show linear and non-linear relationships we can use machine learning to create a model that can predict the the dependent variable and with that get to the main goal: having a better schedule for the final_time

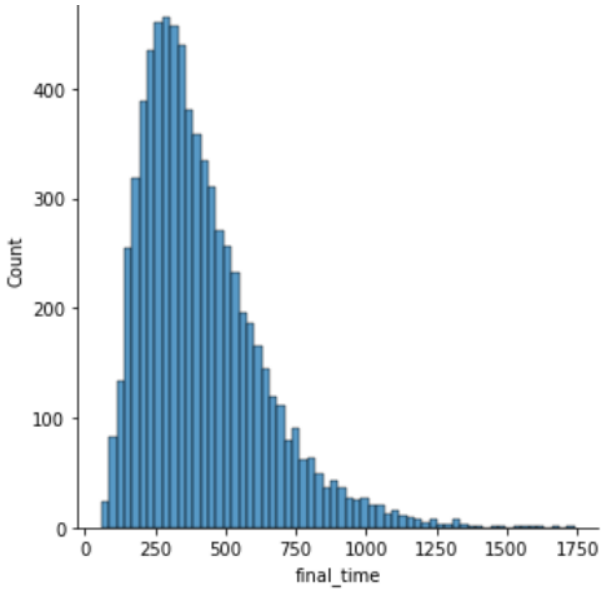
One important thing that must be considered is how the efficiency of the model is going to be measured since there are several ways to calculate this aspect and depending on the characteristics of our model, and actual data, there will be some ways of calculating the error which are more optimal.

Everything mentioned above will be explained in the methodology in each of the levels that we will get deep into.

Methodology

Level 1

The very first thing we did to understand the data is to analyze the distribution of the engine-off time. As you can see there distribution is positively skewed, meaning that it has a long tail to the right side. This will be taken into account to determine the method to measure how precise the model is, and when doing transformation with the data.



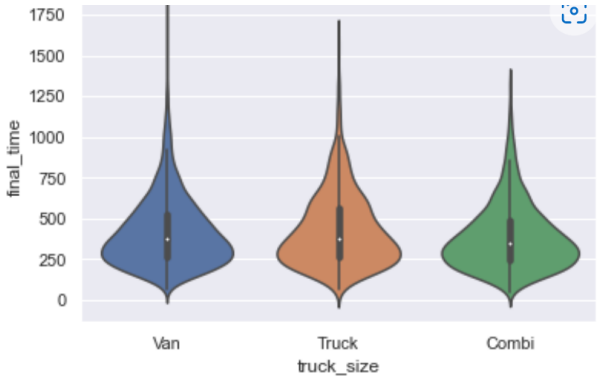
After that we had to check whether the data was clean for use or not. We did that by checking if there is missing data, which is not the case, since there is information for all the observations; and also by checking for outliers. Although the distribution has some very large values, we decided to leave them in, as we didn't consider them to be outliers, and a logarithmic transformation was done to better adjust to a normal distribution. After doing some analysis it can be stated that the information is correct and that it can be used.

After these first steps we have figured out if there is any relationship between the variable that we want to estimate (final_time) and the explanatory variables provided in the data.

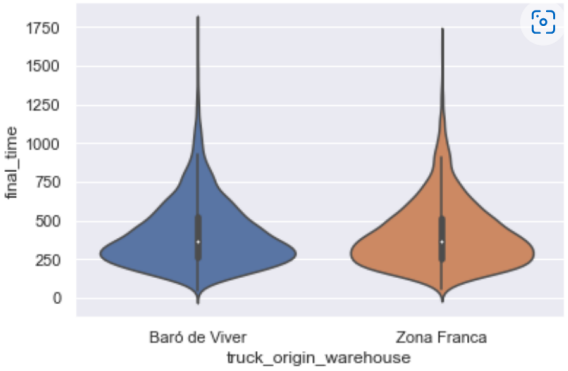
In order to do this analysis we have to make a clear distinction between numerical and categorical variables:

1) Categorical variables:

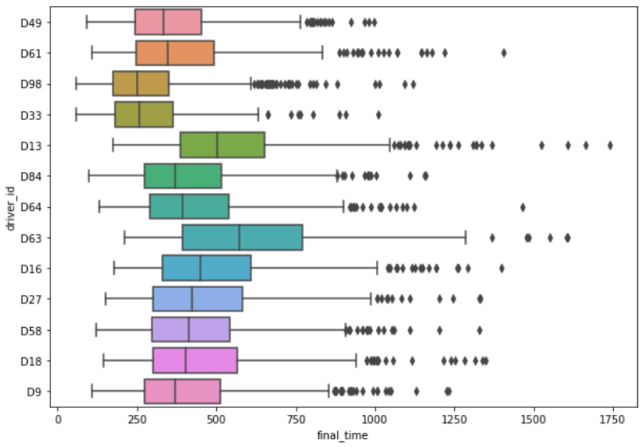
- client_name: the name of the client. No relationship was present with the dependent variable.
- truck_size: what type of truck was being used. Can be one of Combi, Van or Truck. Did not seem to have a meaningful relationship.



- truck_origin_warehouse: from which Beanie Limited warehouse did the route start. Did not have a meaningful relationship

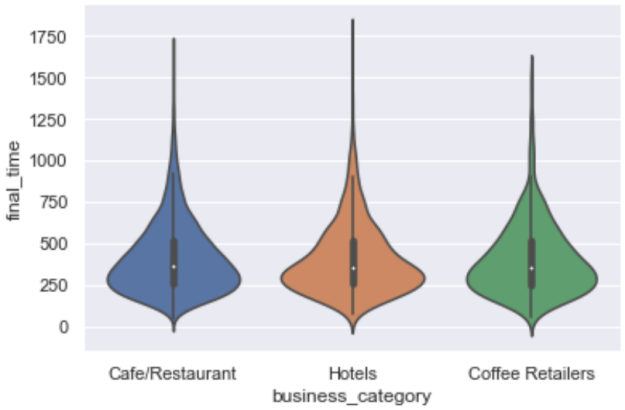


- driver_id: the ID of the driver that was driving the route. Some drivers were noticeably faster or slower, and some others were more homogeneous in terms of the delivery times. We also checked for relationships between the driver and the number of boxes and total weights they carried, but no relationship was apparent, so the differences in delivery times was likely to happen because of the person, not how the deliveries were assigned.

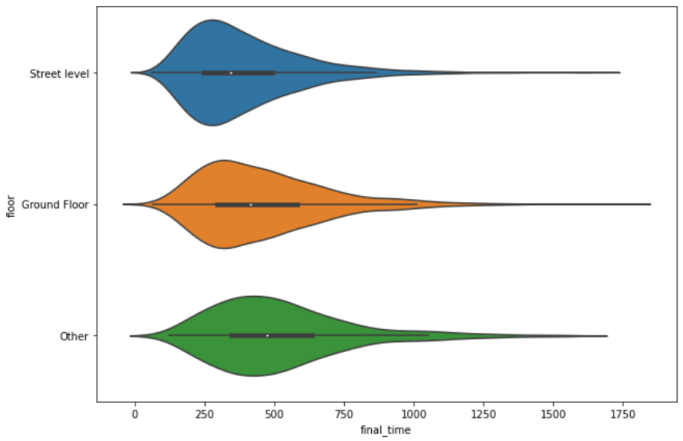


- is_fresh_client: whether the client was fresh at the date of the delivery. Fresh clients are clients that have been doing business with Beanie for less than 30 days. Some relationship with the dependent variable might happen at a first glance.

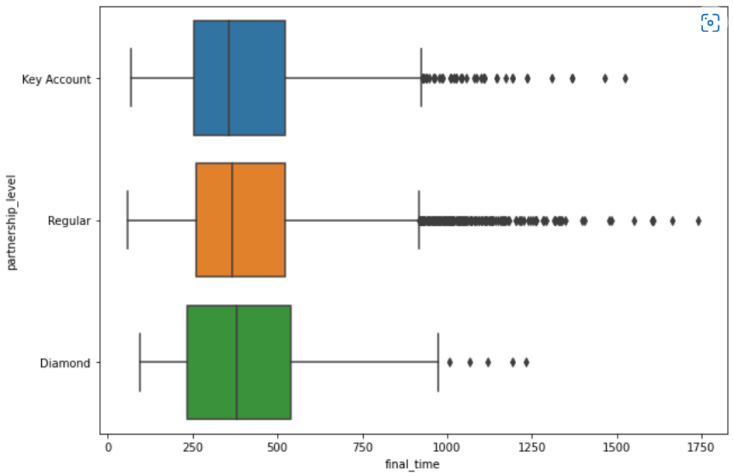
- postcode: the postcode of the client location. Did not have a meaningful relationship
- business_category: whether the client is a hotel, a cafe or restaurant or a coffee retailer. Did not have a meaningful relationship



- floor: the physical position of the client location. Some relationship may be observed, albeit not extremely significant.

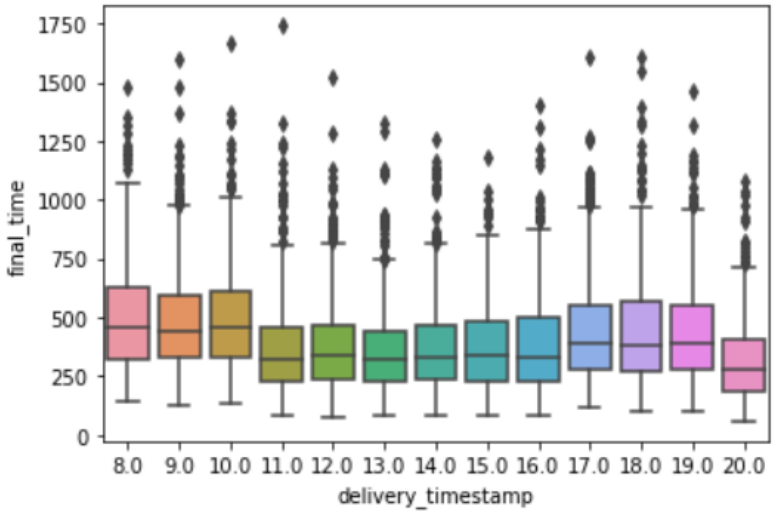


- partnership_level: indicates the partnership level with Beanie. Key Account are important clients for Beanie Limited. Diamond clients are the top priority clients for the company. Did not have a meaningful relationship in the interquartile range, but some relationships in extreme values might happen.

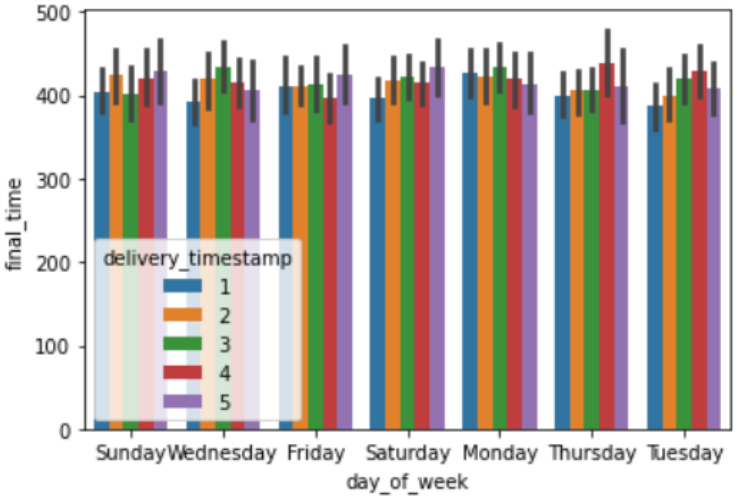


2) Numerical variables:

- delivery_hour: at hour was the delivery done (defined as the moment the engine-off time starts). Some sort of polinomic relationship seems to arise.



- Day of delivery: The day of the week and month the delivery was made. It does not appear to have a relevant relationship.



- total_weight: total weight of the goods delivery.
- box_count: how many distinct boxes were delivered to the client. The coffee beans bags are grouped into boxes for delivery.
- final_time: the engine-off time, measured in seconds, our independent variable which is going to be predicted

The following step is to calculate how the different variables interact between each other and with the final time specially. In order to do so we did the correlation between all of them so we could distinguish which of them have a negative of positive correlation which could help to explain the differences in engine-off time in different deliveries. Furthermore, in appendix 1 you can see a graphical relationship between all numeric variables to further explore insights from the data.

As we can see in the following heatmap there are only seven variables that have a strong relationship (measured in correlation) with the engine-off time (final_time):

Some feature engineering was performed to boost correlations, being able to use some features, improve the model effectiveness, and create new variables out of the existing data. Most notably, (i) the categorical variables were converted into binary variables (1 when it meets a criteria, 0 otherwise), (ii) the hour of the delivery was taken from the delivery time, and (iii) the dependent variable was transformed logarithmically, so as to decrease skewness, become more standardized, and improve performance.

Level 2

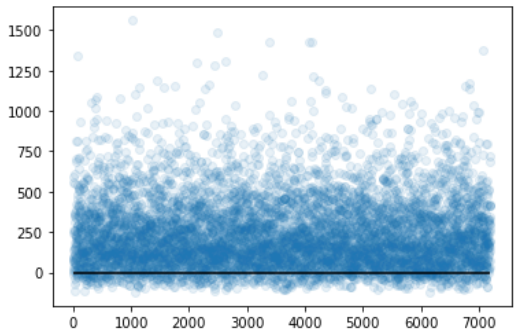
The performance in this study has been evaluated with reference to the MAE (Mean Absolute Error). The MAE is calculated as the sum of absolute errors divided by the sample size:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}\left|y_i - x_i\right|$$

MAE is a comparison of one technique of measurement versus an alternative technique of measurement and uses the same scale of the data that is being measured. And it's especially useful when the distribution of the dependent variable is heavily skewed, like our current case, as it is less affected by large values that would penalize too much in other cost functions, like a Mean Squared Errors approach.

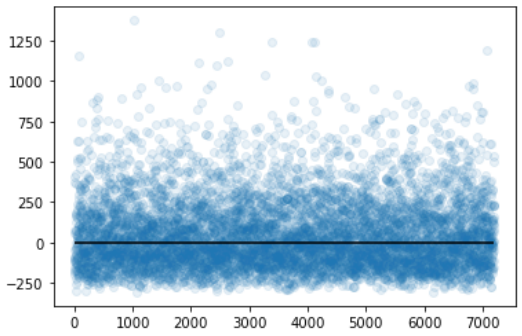
A baseline model is essentially a simple model that acts as a reference in a machine learning project. Its main function is to contextualize the results of trained models in order to have a benchmark of improvement. We have used two baseline algorithms and then tried to make more complex algorithms in order to get a better result. If the improvements we obtain from the more complex model and the explainability tradeoff is worth undergoing, then the Machine Learning model is beneficial for the company.

So, we have the first baseline, to which we will establish the name of Current Strategy (CS). This baseline is your current algorithm that assumes an engine off time of 3 minutes. This algorithm has a MAE of 240 seconds; this is around a 4 minutes error per each delivery.



In the above graph we can see a plot of the errors of the engine time if we use CS. We can see that there are a lot of errors above the estimated line, which leads to an increase of the MAE and means that the current policy is heavily underestimating delivery times.

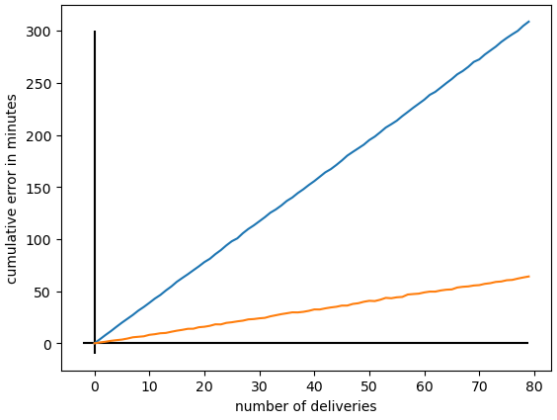
Another baseline that we have used is the median, from now on this estimation we will call Naive Strategy (NS), because it is the simplest strategy we can create, while being better than the baseline CS. The NS is the median of the final time of all the observations of the dataset –about 6 minutes–. We have used the median because it is a central metric and can be sufficiently representative for our data. The NS has a MAE of 161 seconds; this is around 2:41 minutes error per delivery.



In this graph, we can see the same plot as before but using the NS. This estimation is way better than the previous one because the predicted line is more in the middle of the errors.

We can see that the NS is more accurate than the CS, in fact, it decreases in 1:19 minutes the error that CS was making.

In the following graph, we can see a simulation where we ran 3,000 days for each iteration. This plot shows us the cumulative error for the CS (blue), NS (orange), in relation to the number of deliveries that have been made one day. You can see the effect having an underperforming model can have on the daily operations of the company. In only 20 daily deliveries, the accumulated errors are about 78 minutes for the CS, and 15 minutes for the NS. This means that we are consistently underestimating the time it takes for the delivery in both the CS and NS cases, mainly because there are certain points with very high delivery times, but the CS approach is much better in terms of cumulative errors.



In order to measure the performance in the Machine Learning models we have made some cross validation techniques. This technique is used to validate the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. The methods that we have applied are three-fold cross validation. This divides the input dataset into three groups of samples of equal sizes called folds. This took one group as the test data set and used the remaining two groups as the training dataset. Then fit the model on the training set and evaluate the performance of the model using the test set.
Also, we have made some metrics against the test set afterwards to check for external validity. External validity is about whether we think the results of a given study are likely to generalize to other contexts.

Level 3

Methodology:

A pipeline and custom transformers have both been used to develop the model and rapidly check for the best combinations of hyperparameters, feature selection, and feature engineering. The Pipeline is the common code that generates the dataset over which we want to train our model and predict any Machine Learning problem. Therefore, you can change the hyperparameters easily and you can see which variables influence the data the most and therefore get a better result.

Hyperparameters are adjustable parameters that allow you to control the training process of a model. So, Hyperparameter optimization is the process of finding the hyperparameter settings that produce the best performance. It is an important step to optimize the model on the last steps of the process.

The hyperparameters in the pipeline that have been modified are listed below:

i) Logarithmic transformation in the dependent variable. The engine-off time has been transformed into a logarithmic variable to reduce the range of the variable and therefore reduce its absolute error.

ii) Lead time rounded up to seconds. The slicing formula has been used to round this variable. It realistically doesn't make sense to work with the final time in units of measure smaller than that, as we don't need as much precision and it only ads more error to our model.

iii) The delivery time was analyzed to see if drivers were more active at some time of the day depending on the time of day at which the deliveries were made.

iv) The amount and selection of features to see which variables could improve the model.

v) Several hyperparameters of the model themselves, simulated over to check for the optimum in-sample settings).

Once these hyperparameters were analyzed, a train/test dataset was constructed and the model was built.

We chose to run three models: (i) A decision tree regressor as the simplest, most easily explainable model and to aid with feature selection, (ii) a Random Forest regressor as a stronger option of the decision tree regressor, which loses a bit in terms of explainability but considerably improves performance, and (iii) a support vector machine regression model to check how another model would behave.

A Random Forest model consists of a set of individual decision trees, each trained on a slightly different sample of the training data (generated by bootstrapping). The prediction of a new observation is obtained by aggregating the predictions of all the individual trees that make up the model. Many predictive methods generate global models in which a single equation applies to the entire sample space. When the use case involves multiple predictors, which interact with each other in a complex and non-linear way, it is very difficult to find a single global model that is able to reflect the relationship between the variables.

The model that behaved the best was the Random Forest, as it had the best performance, and it retains a fairly good level of explainability.

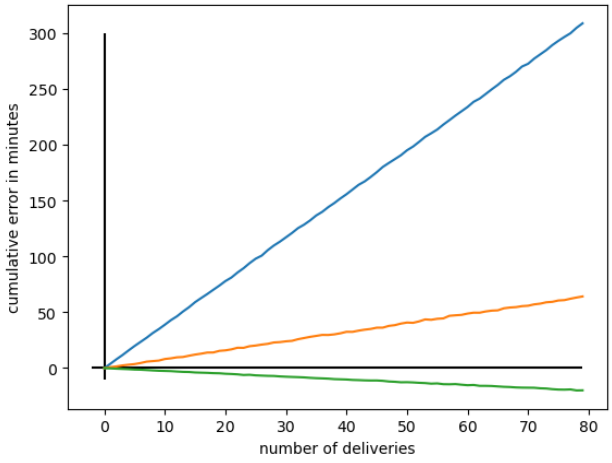
Performance

In order to evaluate the performance of the three models chosen, we ran an error analysis on the predict vs the observed values, according to the Mean Absolute Error cost function specified above on both the train and test samples. Doing so will help us not only to check for external validity of our models, but also verify in-sample overfitting issues.

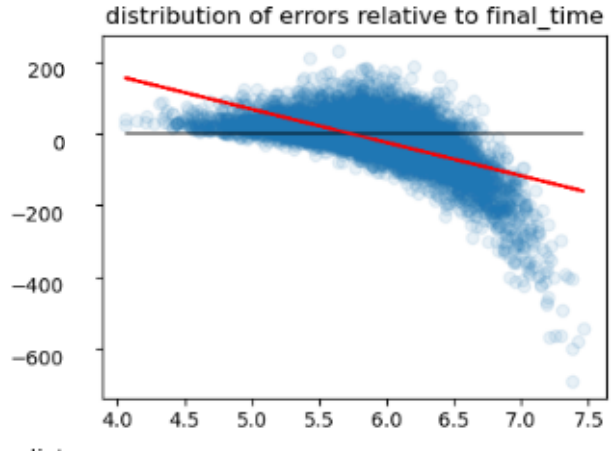
After running the check, we observed that the ranking of the models were consistent between the train and test datasets. In both cases the random forest outperformed the decision tree and the support vector regression models by 10.9% and 31,7%, respectively, in terms of cost function, and outperformed the CS and NS 318% and 181% respectively. The differences between the train and test scores on the decision tree signals the model suffers from overfitting issues.

	Cost Function		**Standard deviation**
	Train	**Test**	**Test**
Random Forest	50.5155	57.4901	63.62
Decision Tree	45.3352	63.7711	70.53
Support Vector Regression	78.3499	75.7319	70.91
Current strategy	240.4923		208.86
Naive Strategy	161.7951		151.21

To analyze the performance of the model, the residuals have been tested to see how they behave and whether they are correlated. Full graphical results of these analyses can be found on the appendix section.



The most relevant residual analysis is show In the following graph, where we can see that the distribution of the errors follows a path, that is, present polynomic relationship. The X-axis shows the observed delivery times in logarithmic scale, and the y-axis shows the error.



The reason why our model is considerably better than both baseline algorithms is that there is a great amount of correlation and other non-linear relationships on the data, meaning that they can be used to explain why a driver takes shorter or longer in a given delivery. Being able to explain the model reduces the error in the amount that we can explain the relationships between variables, and we approximate the factors that influence delivery times. There will be always some amount of error, of course, as there is randomness associated with every delivery and action, but when we find relationships between variables, we can isolate these randomness, and reduce uncertainty.

Level 4

As shown in the table below, there are relatively few variables that are truly important in modeling what the delivery times will be. Most notably, (i) the number of boxes to be delivered, as the driver will have to handle more items; (ii) the total weight of all the packages, because heftier things are harder to move; (iii) the time of the day of the delivery, because at some points during the day, the drivers might be more tired or the clients might have more customers coming in; and (iv) some drivers are faster than others, for instance it could be explain through physical factors and the weight of the boxes.

The following table shows the proportion of times a feature was used in the random forest. Meaning that it is a weighted average of the number of trees that have the feature, to the total number of trees in the random forest. In a nutshell, the logic behind is that the more that individual trees decide to use a feature, the more important it must be to predict the final outcome.

box_count	0.3716	**Street level**	0.0092
total_weight	0.2372	**Van**	0.0090
delivery_hour	0.1005	**Truck**	0.0070
D63	0.0775	**Combi**	0.0061
D64	0.0347	**D61**	0.0046
Other	0.0244	**D33**	0.0037
D98	0.0220	**D27**	0.0029
True	0.0183	**D84**	0.0027
D18	0.0182	**D58**	0.0025
False	0.0173	**D16**	0.0020
Ground Floor	0.0132	**D9**	0.0020
D49	0.0106	**D13**	0.0019

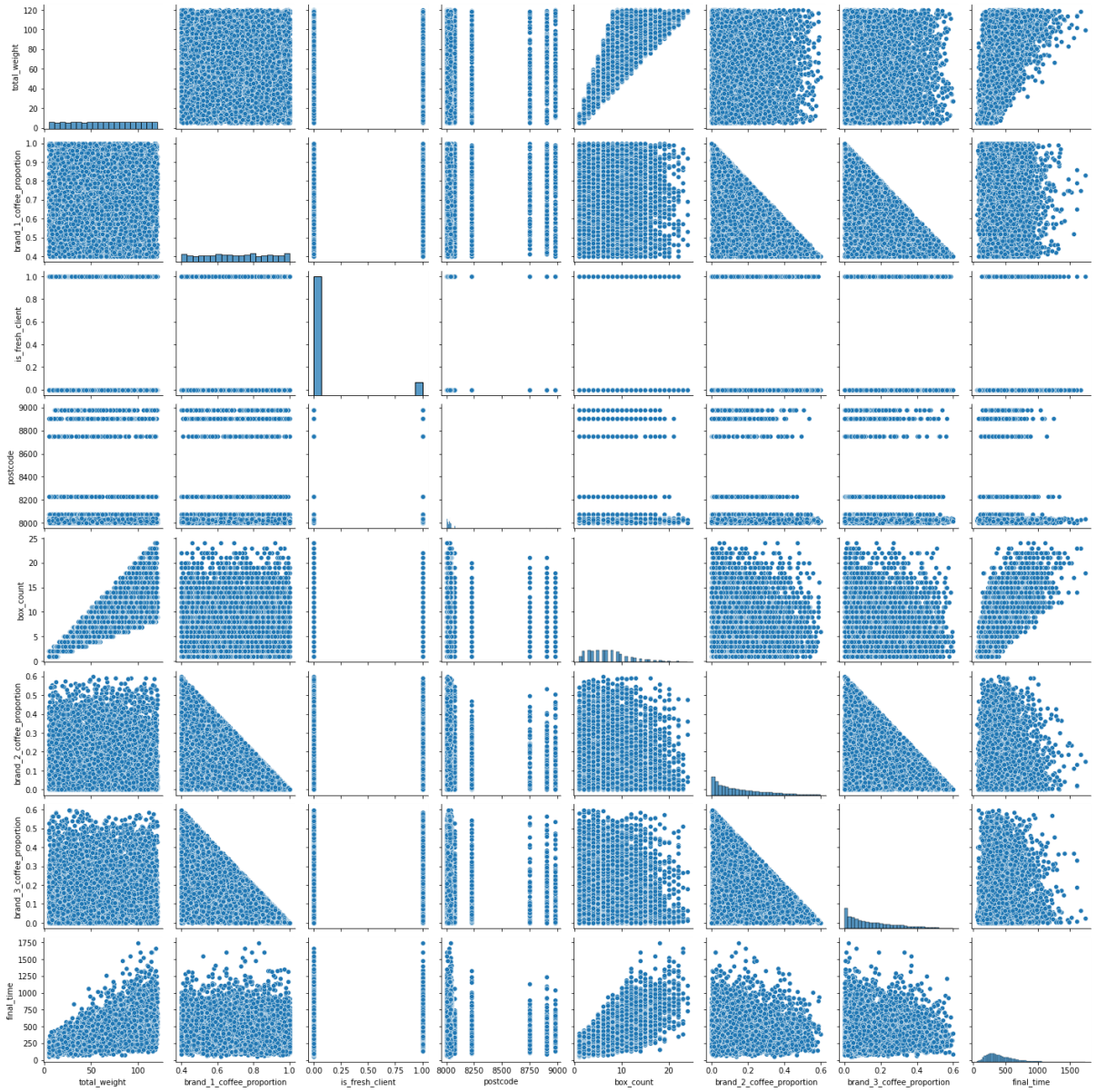
Next steps

The analysis of the errors of our model has yielded fairly interesting conclusions that can be used moving forward. Primarily, that our model, although fairly good –especially compounded throughout a day where the deliveries can be in the forties or even fifties and the errors cancel themselves fairly well, making us observe on average about a 10 minute underestimation– is not sufficiently powerful to explain relationships when the delivery times tend towards high values. What we see in the errors analysis is some heteroskedasticity in many variables, and even some non-linear relationships between the errors and some features. For this reason, you might want to explore the possibility of building a more advanced Machine Learning model, or even a Neural Network, which could potentially further improve the predicting capabilities, but at the cost of explainability. Whether the

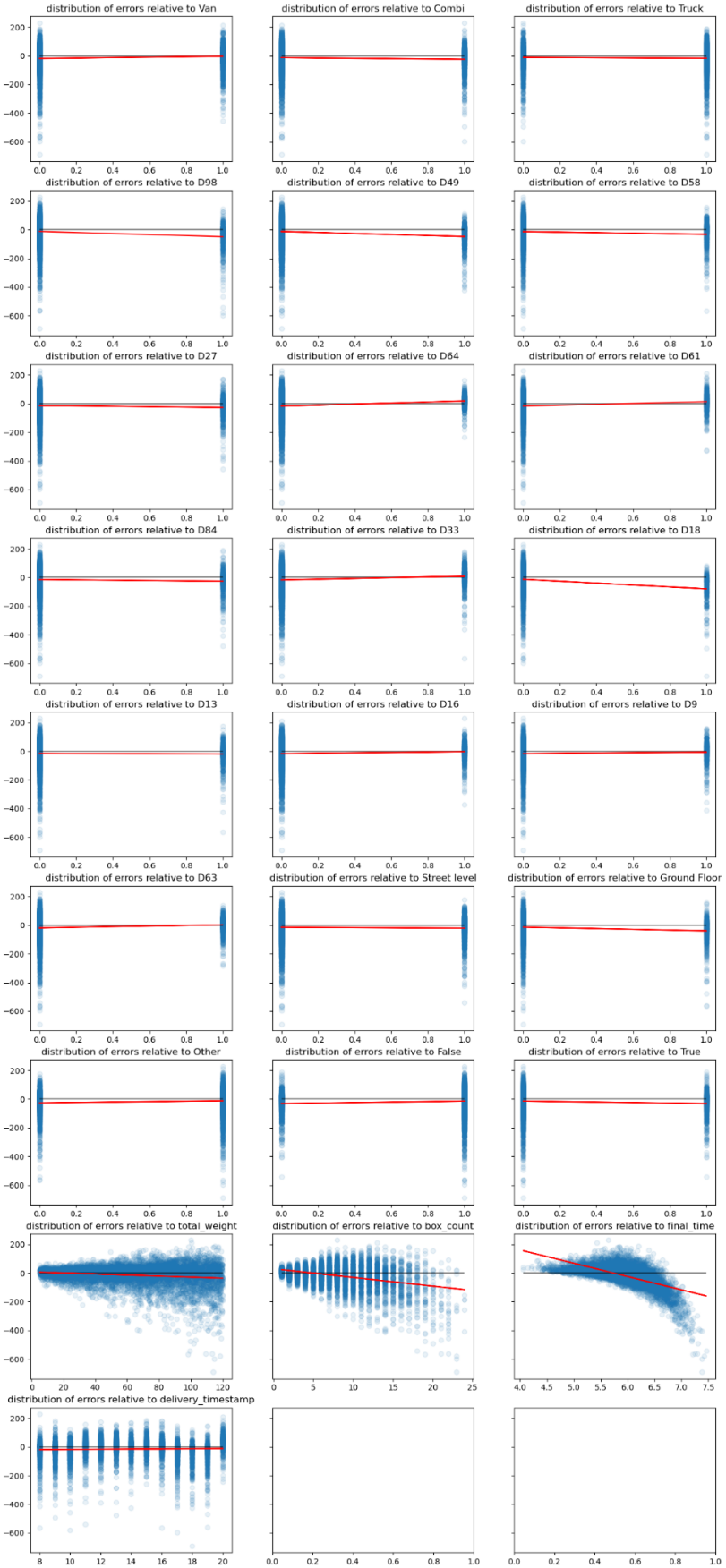
trade-off is worth undergoing or not depends on the specific business necessities. So in our opinion, and considering the results of the simulation carried out and the fact that you have shown an interest in understanding what's behind the predictions the algorithm makes, we would hardly advise trying to obtain this extra performance at the cost of losing explainability.

But some things can be done to improve the model developed. Especially, obtaining more data to make better estimates, other features that might be interesting to add to the model, and more importantly, changing some operations in order to make delivery times lower according to the model estimates. Actions such as making the boxes more homogeneous in terms of weight or size, assigning heavier deliveries to stronger drivers, and so on. This way, the dispersion of the data would be lower, and the estimates would increase to some extent.

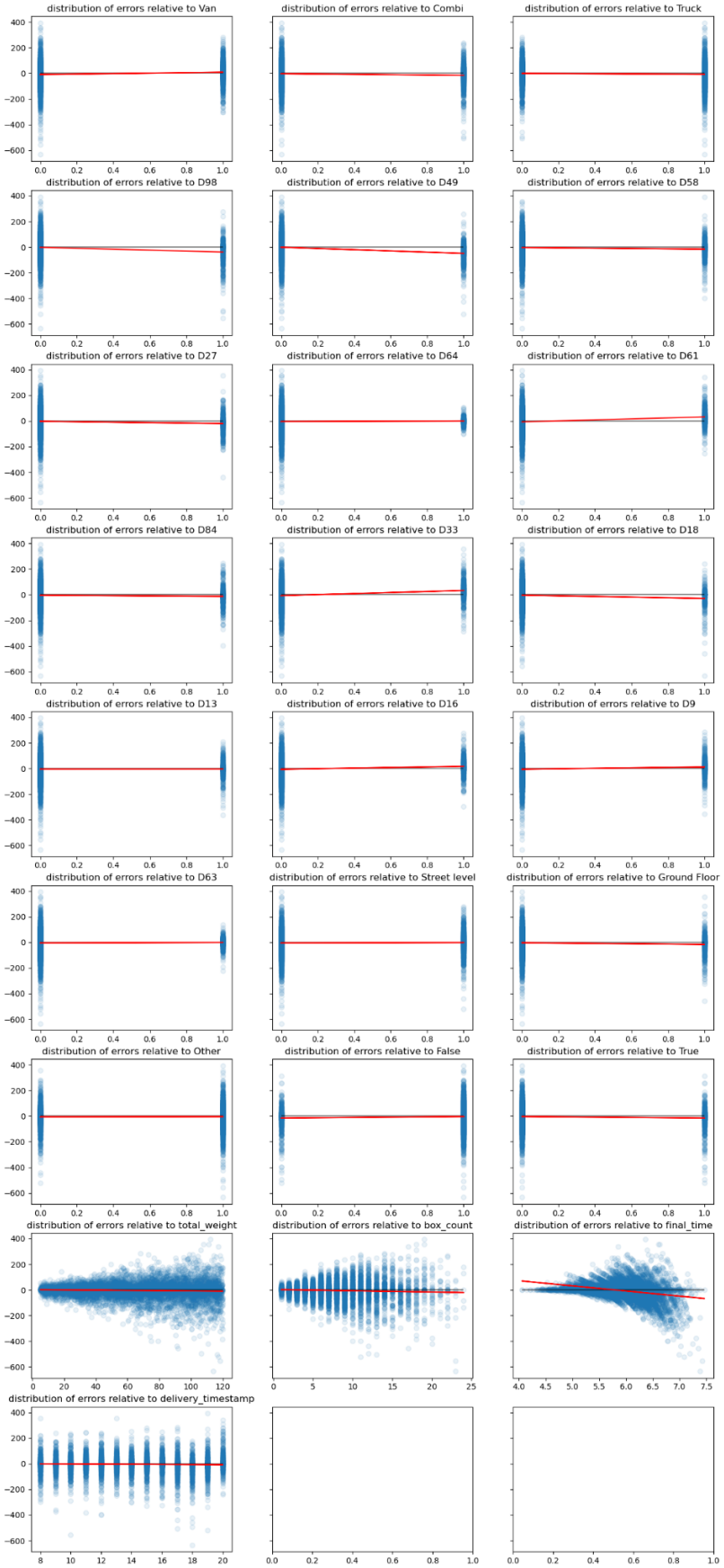
Appendix 1 – Scatterplot matrix of numerical variables



Appendix 2 – Random Forest Regressor



Appendix 3 – Decision tree regressor



Appendix 4 – Support Vector Regressor