Universitat Pompeu Fabra Barcelona

Case 3: Beanie Limited

Vicent Marí
Arnau González
Álvaro Romo

June 16th, 2022

Universitat Pompeu Fabra Barcelona

Dear Estefanía,

First of all, thank you for choosing us to help you with your requirements. We have thoroughly assessed the data you shared with us, and we are ready to present to you the results we obtained from it. We were really satisfied with the final outcome, and we hope you will be too. Below you can find an index that can help you with the structure of the document.

1. Exploratory Data Analysis

We are confident that the ‘drop-offs’ dataset you shared with us is very useful in a machine learning context. As you explained to us when we met, one of the issues you have is that the *engine-off* time estimates are usually off and it leads to operational inefficiencies.

Machine learning can be very useful to try to solve this problem. In this case, we would use something called *supervised learning.* What we do in these types of problems is train, or fit, a model to try to predict the engine-off time based on historical data. This is why the dataset you sent us is great: it provides us thousands of examples of different orders and their engine-off time. What the models do is try to find patterns in the different orders that can help get a better estimate on the engine-off time. For example, maybe certain times of the day, or the number of boxes a driver has to carry lead to a different engine-off time.

While the data you provided with us is clean and no data is missing, we had to do some preprocessing in order to represent some variables in a way that our models could deal with them. We used one-hot encoding for categorical variables (such as truck size, business category, etc). This is to deal only with numbers (model-friendly) and not words.

Once our team established that this dataset could be useful in a machine learning context, we performed an initial analysis on the properties of it (Exploratory Data Analysis), and this is what we found.

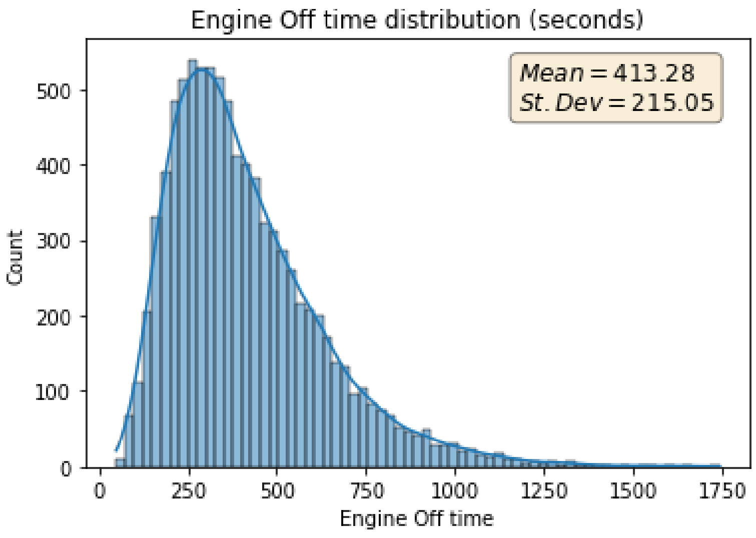


Figure 1: Histogram of *engine off time*

First, we analyzed how the engine time was distributed (figure 1). We noticed that the shape of the distribution was heavily skewed. It can also be seen in the distribution that even though there is a heavy concentration around the mean, the right tail also seems to be fat: meaning there are a lot of values greater than the mean. This fact can also be explained through the standard deviation. We see that while the mean is around 6.8 minutes, the standard variation is around 3.5 minutes.

This histogram indicated to us that there is a lot of variation in the data. So the next thing we did was try to find out which factors influenced this variation the most. Our first hypothesis was that the box count should influence the engine off time substantially: as more boxes may mean several trips for the driver in order to deliver the goods. Also, we were interested in analyzing the effect of a *fresh client* (a client with less than 30 days of doing business with Beanie) in the engine off time. We made this figure to try to assess these relationships.

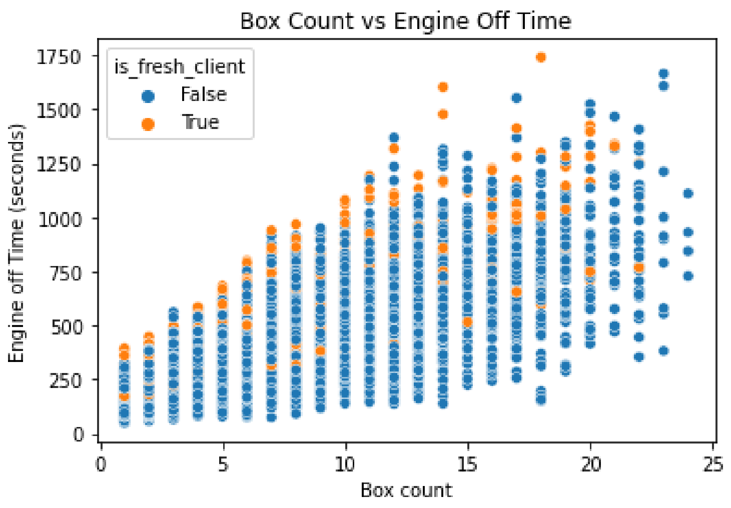


Figure 2: Distribution of *box count* vs *engine off time* depending on *is_fresh_client*

As we can see in figure 2, there seems to be a clear positive correlation between the box count, and the engine off time. We can't ensure that this is a causality relationship, but we think it is logical to assume that more boxes mean more time needed for the driver to deliver an order. From this figure as well, we can see how deliveries to fresh clients (in orange) tend to be longer than to non-fresh clients. These may be due to the fact that fresh clients are not as used to the delivering procedures from Beanie as older clients.

There were two other factors that we thought might influence the engine-off time: the time of the day, and the floor level of the delivery. This is what we found.

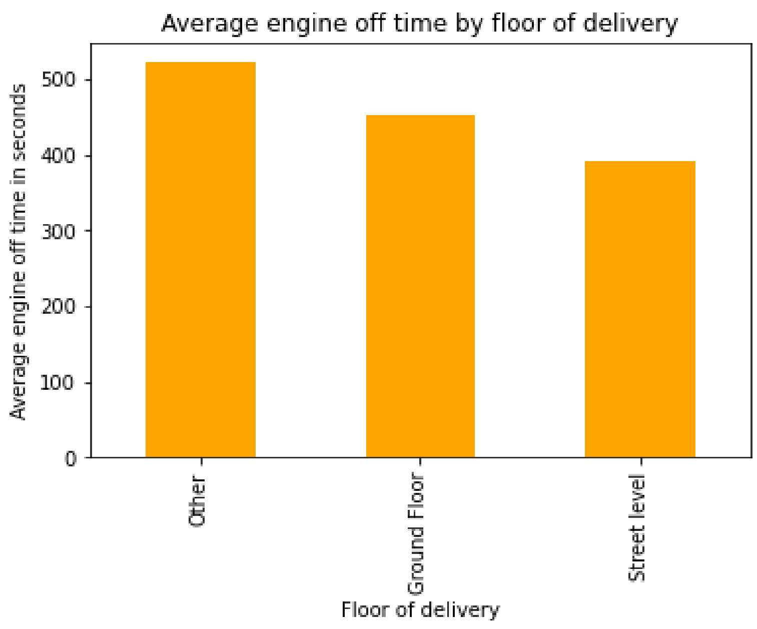


Figure 3: Average engine off time by floor of delivery

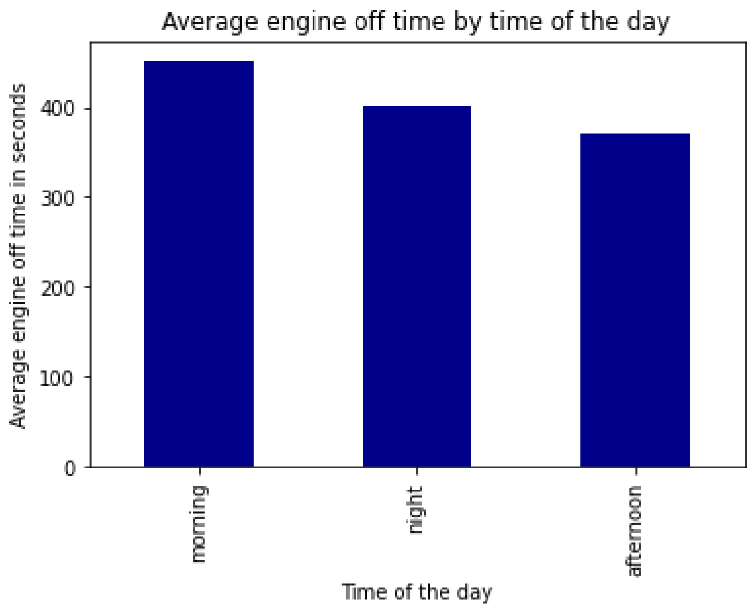


Figure 4: Average engine off time by time of the day

As we can see, both of these factors have an impact on the engine off time of a delivery. If a delivery is made on a floor that is higher than ground or street level, the engine off time will tend to be greater. Also, deliveries made in the morning or at night tend to have a greater engine off time than those made in the afternoon.

Finally, we assessed the impact that drivers themselves have on the engine-off time. We think that it is logical to assume that all drivers behave differently in terms of their delivery methods. That's why we think this is an important factor to take into account to get better estimates. The figure below shows the differences between them.

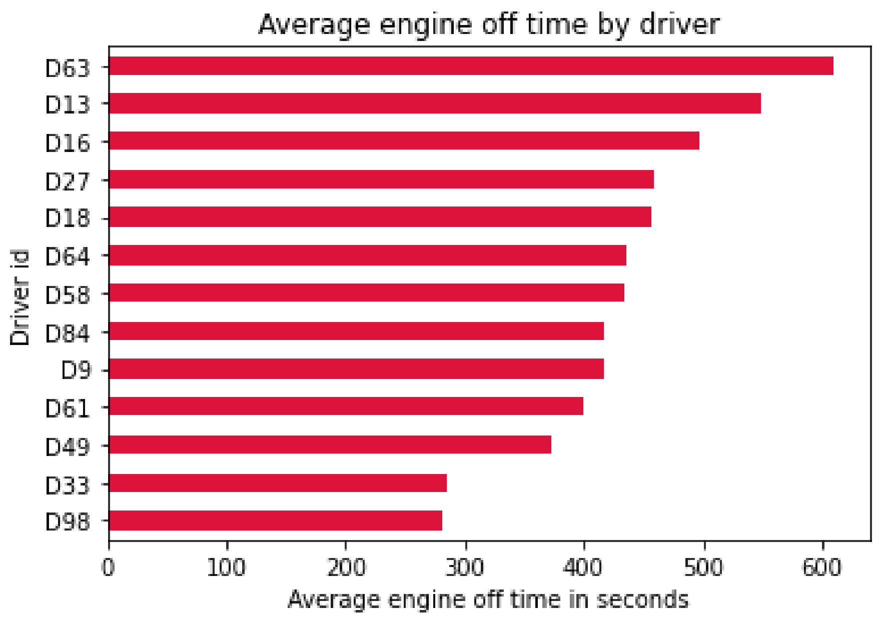


Figure 5: Average engine off time by driver

With these analyses performed on the dataset, we proceeded to train different models in order to get a better estimate on the engine-off time.

2. Model Performance Evaluation and Baseline Algorithm Comparison

Performance Metrics

Deciding how to evaluate a model's performance is no easy task in machine learning: different cases require different performance metrics. After careful consideration, we decided that the best way to evaluate the performance of our model is the *Mean Absolute Percentage Error.* In short, this metric yields, on average, the difference in relative terms of the true value against the predicted value. The word *Absolute* refers to the fact that this metric does not take into account if the error was made from overestimating or underestimating (if it is positive or negative). This doesn't mean that it is not important to take into account if a model over or underestimates. Another key factor here is that we use the errors as percentages. Our reasoning was that we should not give the same weight an error of 10 seconds on a delivery that took 1000 seconds, that an error of 10 seconds on a delivery that took 100 seconds.

Training and Testing Datasets

Calculating this number requires splitting the data into two subsets: the training and the testing dataset. In the training dataset we *show,* or present, the model with the real values of the variable it is trying to predict, in order for the model to *learn* from it.

In this case, we *show* the model the real engine-off times, and try to fit a model capable of predicting these values. In the testing dataset, we only use features in the dataset other than the engine-off time and try to use the model to predict the times only using that. This is how we will get the *Mean Absolute Percentage Error.* By using predictions the model made with the testing dataset, and comparing it with the true engine-off times from the testing dataset.

Cross Validation

We go one step further to get a more accurate estimate of the *MAPE (Mean Absolute Percentage Error).* We will use a technique called *Cross Validation.* This technique tries to deal with the fact that dividing the dataset into different training datasets yields different models, and thus, different predictions. What we do in Cross Validation is divide data into different training and testing datasets, and then fit and evaluate performance each time. By doing this, the MAPE we present is a mean of 10 different models with different training and testing sets.

Baseline Algorithm

We decided to use the mean of the engine-off time as the baseline predictor for a future delivery. We know this is a simple metric, but it helps us establish a baseline for the models we try. The MAPE obtained from this model is the number we will try to reduce as much as we can.

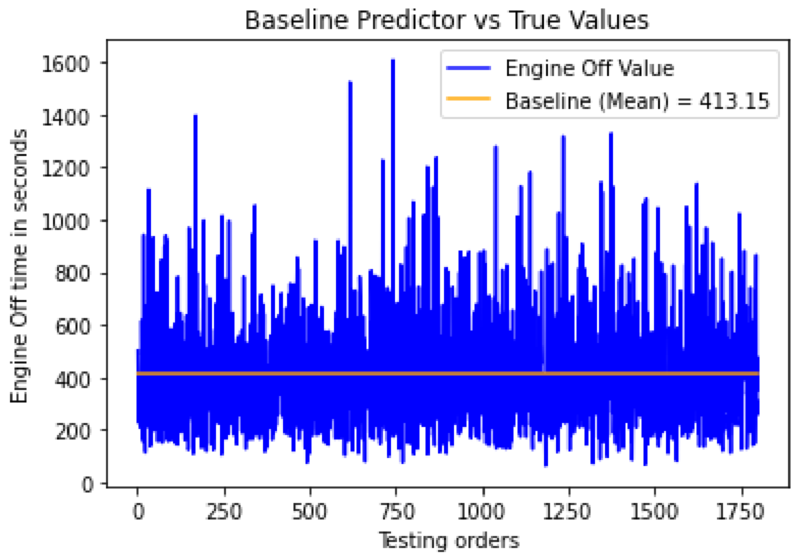


Figure 6: Performance of the baseline predictor against the actual values

This is the baseline model we will compare our other models with. We obtained a ***MAPE* of 50.8%.**

3. Model For Prediction

After trying out and fitting different models to predict engine-off time, we came up with the best model we could by comparing the different testing *MAPEs*.

Stacking Regressor and its variables

The model our team agreed upon for prediction was a Stacking Regressor model. This model combines different predictions from two or more different algorithms, and then decides using another prediction algorithm what is the best way to combine these different predictions. As complicated as it may sound, it is fairly easy to implement and we found out it had great performance with the testing data. Our particular implementation for the Stacking Regressor uses the predictions obtained from two different models. The first model is a Random Forest, and the second one is a Linear Regression. When training the model, we decided it would be good to try transforming the engine-off time variable into a logarithmic scale to avoid making calculations with big numbers. This turned out to be the best decision as it improved the training time, and it reduced the MAPE. As far as the variables we use with each model, we decided which ones to use and which ones to discard by trying out different models and evaluating their performance. In the end, the variables we ended up using for prediction were the following:

- Box Count

- Truck Size
- Business Category
- Floor
- Partnership Level
- Time of day (Morning, Afternoon or Night)
- Driver ID
- Fresh or Not Fresh client

Model Performance

In the following figure we can observe the performance of the stacking regressor (in purple) versus the baseline model and the actual values to predict.

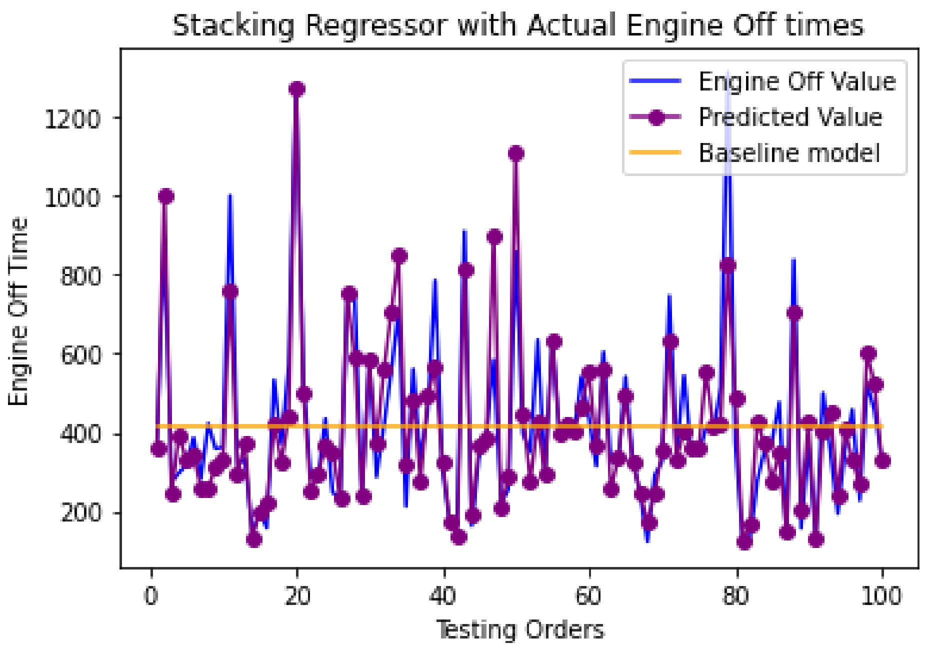


Figure 7: Comparison of the predicted values against the actual values and the baseline model

As it is evident from the plot, the performance that the model had was much better than what a simple average may tell us. Also, to get a good estimate on what the MAPE for this model is, we performed the Cross Validation technique we discussed earlier. By using 10 different folds (10 different training and testing datasets) we obtained a **MAPE of 17.9%**.

Comparison to Baseline Model

The baseline model, which was just the mean of the engine-off time, yielded a MAPE of 50.8% for the testing dataset, compared to a MAPE of 17.9% for the stacking regressor model. We think this improvement is due to the fact that our model takes into account each particular characteristic of a delivery. Our model considers each aspect (listed above) when making a prediction to try to come up with a more accurate number. The baseline model, on the other hand, is a number obtained by other engine-off times, but does not consider the specifics of each delivery: the box count, the driver, the floor, if it's a fresh client, etc.

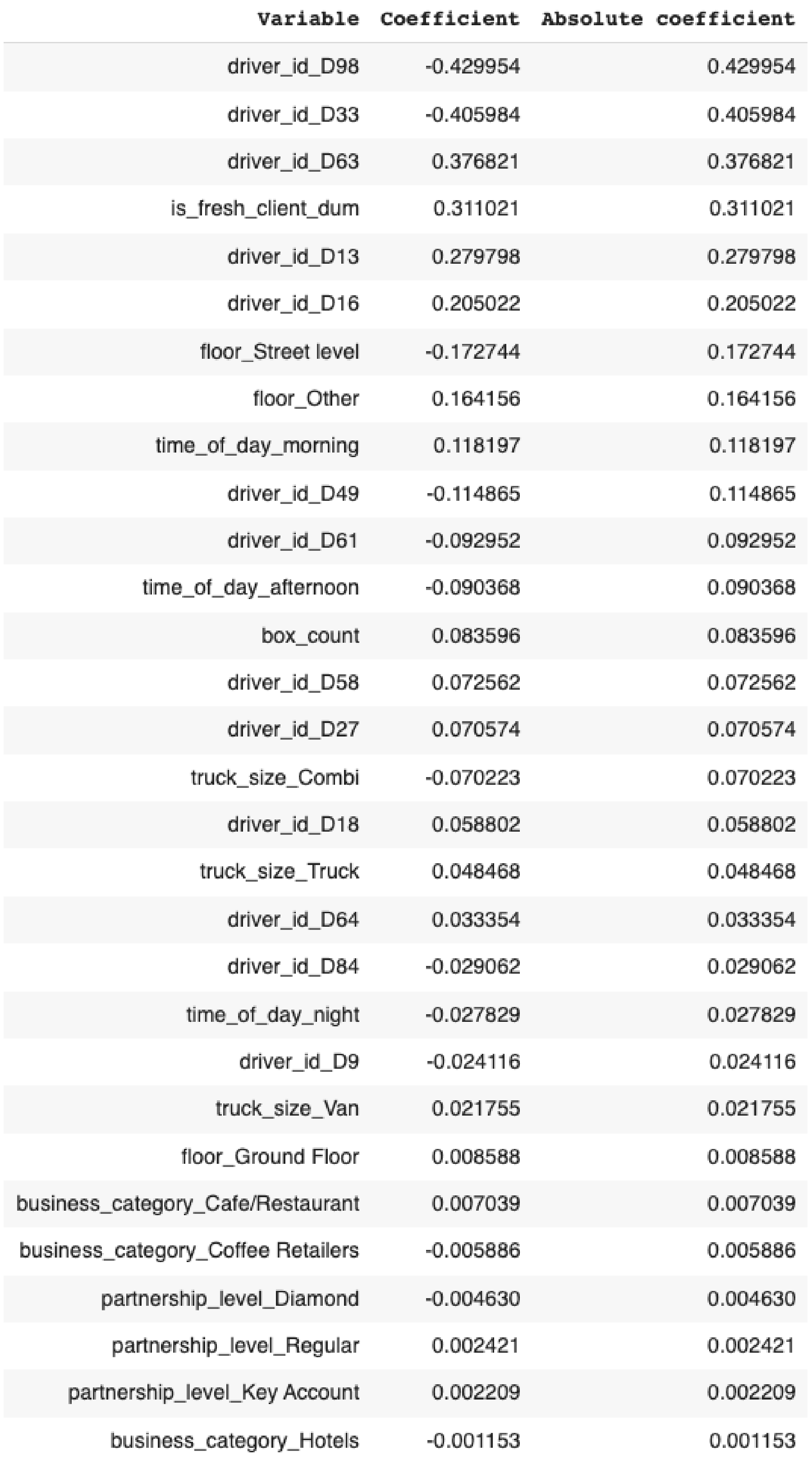
4. Machine Learning algorithm specifics

One of the drawbacks that ensemble methods have, such as Stacking (the model we use for prediction) is that they have a low degree of interpretability: it is unclear or difficult to see the effect that each variable has on the prediction. Fortunately, there are several models that do have a high degree of interpretability. In this case we can use a model called Linear Regression to try to predict the engine-off time as well. Even though the predictive power of this model is not as good as the stacking model, we can interpret certain aspects of the model to try to quantify the effect each one has on the prediction.

Linear Regression Model

Linear regression, even though it's a fairly simple model, is often the best solution for many problems. In this case, as we were focusing solely on accuracy and reducing the testing MAPE, we chose the Stacking Regressor model, but Linear Regression does a great job as well. With this model, we obtained a **MAPE of 18.6%** using the same Cross Validation technique as with the Stacking Regressor. However, Linear Regression, as we mentioned before, is very useful if we wish to analyze the specific impact of each variable.

When fitting a regression model, we obtain a coefficient associated with each variable we input the model. In a way, setting aside the mathematical aspect of the model, we can use each coefficient and sort them to quantify the effect the individual effect each variable has on the engine-off time. These were the results.



Variable	Coefficient	Absolute coefficient
driver_id_D98	-0.429954	0.429954
driver_id_D33	-0.405984	0.405984
driver_id_D63	0.376821	0.376821
is_fresh_client_dum	0.311021	0.311021
driver_id_D13	0.279798	0.279798
driver_id_D16	0.205022	0.205022
floor_Street level	-0.172744	0.172744
floor_Other	0.164156	0.164156
time_of_day_morning	0.118197	0.118197
driver_id_D49	-0.114865	0.114865
driver_id_D61	-0.092952	0.092952
time_of_day_afternoon	-0.090368	0.090368
box_count	0.083596	0.083596
driver_id_D58	0.072562	0.072562
driver_id_D27	0.070574	0.070574
truck_size_Combi	-0.070223	0.070223
driver_id_D18	0.058802	0.058802
truck_size_Truck	0.048468	0.048468
driver_id_D64	0.033354	0.033354
driver_id_D84	-0.029062	0.029062
time_of_day_night	-0.027829	0.027829
driver_id_D9	-0.024116	0.024116
truck_size_Van	0.021755	0.021755
floor_Ground Floor	0.008588	0.008588
business_category_Cafe/Restaurant	0.007039	0.007039
business_category_Coffee Retailers	-0.005886	0.005886
partnership_level_Diamond	-0.004630	0.004630
partnership_level_Regular	0.002421	0.002421
partnership_level_Key Account	0.002209	0.002209
business_category_Hotels	-0.001153	0.001153

Table 1: Linear regression estimations

This table has three columns: one for the variables, one for the coefficient values, and one for the absolute value of the coefficients. This last column is very important as it allows us to sort the magnitude of the coefficients from the biggest one to the smallest one, regardless if it is positive or negative.

When interpreting a coefficient, the sign of it tells us the effect it has on the variable we want to predict. If it is negative, an increase in that variable will result in a decrease on the engine-off time prediction. On the other hand, if a variable is positive, an increase in that variable will result in an increase on the engine-off time prediction.

As we can see from the table above, who is driving (the driver id) seems to be a great indicator of what will be the engine-off time. For example, driver 98 and 33 are the fastest, while driver 63 is the slowest. Also, if a client is fresh or not seems to have a great impact on the predicted variable. We can perform this analysis with every variable from the table, and we can get an idea of how important is each variable for the final engine-off time.

Next Steps

Every model relies on data. That's why if you want to improve a model, the first thing you have to do is improve your dataset. In this case, we think that the best thing to do would be to think thoroughly about what can be some other key factors that may influence the engine-off time of a delivery. Some examples that come to mind could be weather factors (like rain or fog) that affect traffic, the parking availability for each dropoff location, etc. Also, we noticed that the dataset you shared with us only contains information from January to May; our team thinks that the model could benefit from having data of deliveries from every month of the year, as seasonality could have an effect on traffic and order amounts. These are just some examples of variables we think could help the performance of the model.

5. Conclusion

We started off this task with one goal in mind: improving the estimates for the engine-off time. Initially, we established as a baseline model the mean for all engine-off times. We thought that this number would be around 3 minutes, but to our surprise, the number we got was closer to 7 minutes. As we kept on analyzing the data, we noticed how much variation was present in the engine-off time for each delivery; this led us to believe that there were a lot of factors influencing this that had to be taken into account. By trying and testing different models, we were able to come up with a model that had a *MAPE* of 17.8%. This was a great improvement from our baseline model, which had a *MAPE* of 50.8%. Additionally, we were able to identify how much each variable affects the engine-off time by using Linear

Regression coefficients. We think (and hope) this analysis will be very helpful for you and your team, and that these new estimates will improve operations for Beanie.