Anomaly detection in average fuel consumption with XAI techniques for dynamic generation of explanations

Alberto Barbado[1,2]
alberto.barbadogonzalez@telefonica.com
[1] Telefónica, 28050 Madrid, Spain
[2] Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, 28660 Boadilla del Monte, Spain

Abstract

In this paper we show a complete process for unsupervised anomaly detection for the average fuel consumption of fleet vehicles that is able to explain what variables are affecting the consumption in terms of feature relevance. For doing that, we combine the anomaly detection with a surrogate model that is able to provide that feature relevance. For this part, we evaluate both whitebox models from the literature, as well as novel variations over them, and blackbox models combined with local posthoc feature relevance techniques.

The evaluation is done using real IoT data belonging to Telefónica, and is measured both in terms of model performance, as well as using Explainable AI metrics that compare the explanations generated in terms representativeness, fidelity, stability and contrastiveness.

The explanations generate counterfactual recommendations that show what could have been done to reduce the average fuel consumption of a vehicle and turn it into an inlier. The procedure is combined with domain knowledge expressed in business rules, and is able to adequate the type of explanations depending on the target user profile.

Keywords XAI. Fuel Consumption. Anomaly Detection. Explainable Boosting Machines. Feature Relevance. IoT.

1. Introduction

Combining Advanced Analytics techniques together with IoT (Internet of Things) data offers many possibilities to find and extract relevant insights for business decisions. At Telefónica, for instance, we see how the union of Machine Learning (ML) with IoT data helps to create new use cases for Fleet Management Industry. An example of it is the usage of ML for anomaly detection of the average fuel consumption of vehicles. For a fleet manager, it is very useful to be able to find which vehicles are having an abnormal fuel consumption, since it is crucial for optimizing costs.

However, detecting which vehicles have an anomalous average fuel consumption alone is not enough. Only providing that information leads to more questions than answers. Why are the vehicles consuming that extra amount of fuel? How could it be reduced?. These questions are not answered by a binary output that indicates which consumptions are anomalous and which ones are not.

This is one of the reasons why Explainable AI (XAI) is so critical: it enhances that initial information with different types of explanations, helping to answer those questions that may arise. In fact, XAI is one of the core elements of Responsible Artificial Intelligence (RAI) [1], which is a fundamental part of Telefónica's AI principles [2].

Nonetheless, XAI is still an emerging field with many uncharted or relatively new territories. For instance, how do we now that the explanations generated are good? How do we compare XAI quantitatively in order to find the one that provides better explanations? These questions address the importance of metrics for XAI for measuring the understandability of explanations.

Together with those questions, another issue is the following one: Do the explanations adapt to the user profile? Are they adjusted in such a way that the target audience finds them clear and useful enough?

Also, even though explanations themselves are useful, there is always a caveat present: What happens when explanations contradict apriori knowledge of a field? How do we ensure that apriori knowledge and explanations are aligned?. Though, regarding the first question, it may be possible that explanations differ from domain knowledge either because it is wrong or because it may complement it, in many cases the important question is the second one: ensuring alignment between apriori knowledge and explanations.

Finally, even considering good understandable explanations that are aligned with domain knowledge and that are expressed in an understandable way for their audience, there are still questions unanswered. For example, what shall we do about it? The prescriptive dimension also arises, remarking the importance of not only providing insights, but also suggesting possible actions to further help the decision maker.

Taking all these questions in consideration, in these paper we will propose a complete process to address the business need of not only detecting anomalies within the average fuel consumption of a fleet of vehicles, but also explaining what causes them. This process will include how to adjust the explanations to be understandable by its audience, how to include business rules in order to ensure that they are aligned with domain knowledge, and how to provide counterfactual recommendations on what may be done to reduce the average fuel consumption of outliers in order to turn them to inliers.

We will analyse how to generate these explanations for unsupervised anomaly detection using surrogate models. These models help to find the feature relevance relationship between input features and a target one related to the output of the unsupervised anomaly detection. Among these surrogate models, we will use both blackbox models (XGBoost and LightGBM) together with posthoc local explanations XAI techniques for feature relevance (SHAP and LIME), and also whitebox models both classical (ElasticNet) and novel ones (Explainable Boosting Machines, EBM). Additionally, we will propose and analyse a modification over the standard EBM that takes into account, both for explanations and for predictions, differences that may exist within different subgroups of vehicles.

Following this, the main contributions of our work are:

- Proposing a complete process for unsupervised anomaly detection in the average fuel consumption of the vehicles of a fleet that includes: choosing relevant features, detecting outliers in an unsupervised manner, generating explanations of what affects the average fuel consumption of outliers, aligning those explanations to business rules, generating recommendations with them proposing what may be done to turn outliers to inliers, and adjusting them to be easily understandable for their target audience, considering two different user profiles that could benefit from them.
- Comparing blackbox models with local posthoc XAI techniques against EBM using Telefónica's real-world IoT industry data in order to see if a whitebox model (EBM) could match the blackbox one plus posthoc solution within our use case. This comparison will be done at two complementary levels.
    - First, we will compare if the predictive power of EBM as a surrogate model could match the one obtained with other reliable boosting models (XGBoost and LightGBM).
    - Second, we will compare if the local explanations provided directly by EBM are similar to those obtained by combining the blackbox models with local posthoc XAI techniques for feature relevance (LIME and SHAP). For that, we will propose a set of metrics useful for quantifying and comparing explanations from different perspectives: representativeness, fidelity, stability and contrastiveness.
- Finally, along with EBM, we will analyse the aforementioned metrics using a variation of the standard EBM that takes into account a set of categorical features in order to adjust the predictions and features importance.

The rest of the paper is organized as follows. First, we describe some related work in the area of anomaly detection for fuel consumption, together with previous works regarding XAI for feature relevance explanations. This Chapter will also mention other works regarding the combination of explanations with domain knowledge, as well as some of the research conducted regarding metrics for measuring explainability. Then, we describe the different steps of our process proposal, including the metrics for both comparing the different XAI solutions as well as measuring the understandability of the explanations in themselves. We will also include at this point what business rules are we considering for expressing the domain knowledge and how we combine them with the explanations. Following this, we present an empirical evaluation and comparison using real-world IoT data. We then conclude, showing also potential future research lines of work.

2. Related Work

2.1. Anomaly detection for fuel consumption

The detection of anomalous fuel consumption vehicles from a fleet is present at different research works within the literature. In [3], the authors show how to detect fuel anomalies using unsupervised algorithms (Self-Organizing Maps, SOM). The authors aim to find fuel fraud situations within fleet vehicle data at Bolivia (using a data set of 1000 vehicles with 190627 data points). These situations are normally linked to high fuel purchases within a short period of time. They effectively show how to find clusters within the space of the SOM in order to identify fuel anomalies and detect fraudulent scenarios by evaluating their proposal over a test set. As the authors mention, there are many features that can be used in order to contextualize the fuel consumption (p.e. the normal monthly consumption of the vehicle, the behaviour of other vehicles of the same subgroup...). Their proposal leads only to an output that identifies anomalies, but it could be greatly enhanced with XAI techniques that provides additional insights on what contextual features are relevant for that high fuel consumption.

Fuel fraud is not the only case of possible fuel anomalies within a fleet. As researched at [4], driving behaviour may also lead to an increased fuel consumption. Within driving behaviour variables they mention several features, such as RPM speed, acceleration (both forward, and negative from braking), over speed or gear position.

Even though the previous literature includes researches related to the detection of anomalous fuel consumption (both from fraud scenarios and from contextual variables), to the best of our knowledge there are no previous works regarding the explanation of those anomalies using XAI techniques.

2.2. Surrogate method for feature relevance

From among the different outputs than a post-hoc XAI technique can provide, one of them is in terms of feature relevance [5, 1]. This output quantifies the individual contribution of each training feature to the target variable. Regarding specifically the case of local explanations, LIME (Local interpretable model-agnostic explanations) [6] is a widely known solution. It approximates the decision frontier of the underlying model through an algorithmic transparent model (p.e. a Linear Regression) trained over artificially generated neighbours data points, in order to indicate the relative contribution of each feature to the prediction. LIME, then, fits an independent model for each data point that are going to be explained. Due to this, a particular feature value could have different feature relevance values depending on the data point considered. Also, since the models are independent, each one of them will have its own intercept value.

Another well-known XAI technique for feature relevance outputs and local explanations is SHAP (SHAPley Additive exPlanations) [7]. SHAP is based on the concept of SHAPely values [8], rooted in the field of game theory. SHAPely values consider each feature (or set of features) as ”players” within a game, where the ”gain” is the difference between the predicted target feature value and the average value of the target feature. Thus, it distributes that ”gain” among the ”players” depending on their contribution. Following the idea of SHAPely values, SHAP brings it forward, and represents the explanations through an additive feature attribution method (a Linear Model).

There are different algorithms alternatives for SHAP. The first proposal is known as Kernel SHAP. The literature, however, saw some disadvantages with this proposal, mainly that its computation time was too high, and that it ignores possible feature dependence (p.e. correlations), like most permutation-based methods [5]. These limitations are solved with other SHAP algorithms, like Tree-SHAP [9]. Tree-SHAP was conceived as an alternative for tree-based models, and in facts shortens the computation time, as well as takes into account existing dependences by modelling the conditional expected prediction. However, it has been noted that Tree-SHAP may yield unintuitive feature attributions [5].

Regardless of the SHAP algorithm considered from those two options, the output can be expressed through a pairwise plot, where each of the data points feature values is represented, along with their corresponding feature relevance. Thus, as it happens with LIME, a particular feature value may correspond to different feature relevances, depending on the remaining feature values at that data point. However, contrary to LIME, SHAP has common intercept for every individual data point explained.

SHAP, as a feature relevance XAI technique, has been seen as a way to generate counterfactual explanations for binary classification algorithms. This is the case of [10], where the authors propose a method that searches for neighbours of the data point to be explained (from the same or from a different class) and compare their feature relevance in order to see what is helping to maintain the current prediction and what contributes more to change it.

Though feature relevance posthoc XAI techniques are suitable for building explanations later on (like the aforementioned example), [1] proposes a guideline for ensuring interpretable AI models where an algorithmic transparent model should be tried before changing to a blackbox+XAI combination. The literature is advancing on the research of whitebox models that have performances on pair with complex blackbox ones, in order to contribute to the usage of models that do not need posthoc XAI techniques to understand how they took a decision. This is the case of Explainable Boosting Machines (EBM) [11]. EBM are based on Generalized Additive Model algorithms (GAM) [12]. GAM models use an additive function, similar in structure to that of a Linear Regression, where each feature is modeled through a function that provides a feature relevance value that quantifies the individual contribution of a particular feature to the predicted value. By being modeled with a function that does not have to be linear, GAM provide the option to infer non-linear relationships, thus potentially increasing the model generalization [5]. Equation 1 shows the basic structure of that equation.

$$y = \beta_0 + \sum_{n=1} f_i(x_i) \qquad (1)$$

GAM proposal is improved by GA^2M algorithm [13], the algorithm behind EBM (the difference between them is that EBM is a faster implementation of it). EBM have several improvements over the original GAM. First, the feature functions of EBM can be learned through bagging and boosting techniques. During boosting, only one feature is trained at each step (round-robin) using a very low learning rate in order to make the feature order used irrelevant. This round-robin procedure also lessens the effects of colinearity. Finally, if there are pairwise interactions between features, EBM can detect them and include them as additional terms, as shown at Equation 2 [11].

$$y = \beta_0 + \sum_{n=1} f_i(x_i) + \sum_{n=1} f_{ij}(x_i, x_j) \qquad (2)$$

2.3. Domain knowledge combined with XAI

Within the review of [1], one of the open research challenges is combining domain knowledge with the explanations generated in order to enhance the user's understandability. The review mentions that this challenge is specially addressed through the combination of deep learning blackbox models (connectionism) together with symbolic approaches, that are algorithmic transparent and generally directly interpretable, and with domain knowledge expressed through ontologies. This is the case of [14], where the authors propose a variant of TREPAN algorithm that uses domain ontologies in the XAI phase. TREPAN uses surrogate decision trees to explain any blackbox model (model agnostic). However, as the authors highlight, many times those trees are not understandable by a final user. That is why they propose a variation on the algorithm that gathers information from a domain ontology, and uses it to prioritise using features for the splits that are more general within the ontology. The prioritisation is done by penalizing more the Information Gain value from considering a feature for the split if that feature is too specific. They have assessed their proposal with real users, and they found that indeed using domain knowledge enhances user understandability.

We see that the domain knowledge can indeed be applied to adjust the explanations generated, and it can be done at different moments during a ML model life cycle. It can be done at the ML model itself (for instance, finding hyperparameters that enhance more the model understandability). it can also be done, like in the aforementioned paper, during the training of a posthoc XAI method. Finally, it can be also applied after the XAI method generates the explanations, in order to adjust them to the domain knowledge.

2.4. Metrics for XAI

The review of [1] dedicates part of the section regarding XAI future research needs to the necessity of metrics to assess the understandability of the explanations generated. The authors propose the following definition of explainability: ”Explainability is defined as the ability a model has to make its functioning clearer to an audience”. However, there is a lack of metrics to measure how well the explanations generated matches that definition of explainability. So, there is an open research opportunity in that area. However, regardless of the lack of metrics, they highlight some aspects that should be considered within those future metrics. Among those aspects, they mention the ”goodness”, ”usefulness” and ”satisfaction” of the explanations, along with the improvement of the mental model of the user thanks to them. Also, it should be measured how the explanations impact the ”trust” and ”reliance” by the user in the model.

This connects with what several authors have mentioned what properties should be measured by XAI metrics. A relevant research is [15]. There, the authors analyse the scientific literature and propose a taxonomy of properties for different explainability scenarios that depend on the use case and the audience of the explanations. The scenarios are explanations at a general level, individual explanations, and explanations that are human-friendly. Metric properties regarding general and individual explanations aim to measure the explanation's understandability regardless of the user. Human-friendly ones take the user into account for assessing the explanations. Thus, the properties that a metric to evaluate individual explanations should have include aspects like ”representativeness” (instances covered by the explanations), ”fidelity” (how well the explanations approximate the underlying model), or ”stability” (how similar are explanations for similar instances). In contrast, metric properties to check if explanations are human-friendly include aspects like ”contrastiveness” (if explanations are in the form of ”if you have done X instead or Y, the output would have changed from A to B”) or ”selectivity” (if explanations do not include all the causes, but only the most relevant ones).

Another example of taxonomies for explanation metrics is [16]. The authors also propose a split between metrics with and without considering the user. First, they refer to ”explanation goodness” for metrics that assess explanation understandability regarding the ML model. Within these type of metrics they include properties like ”precision”. However, even when a set of explanations have good metric values for ”explanation goodness”, they may not help the users. This is why there is a second group of metric properties for ”explanation satisfaction”, that include aspects such as ”understandability”, ”completeness”, ”usefulness” or ”feeling of satisfaction”. The authors propose a set of questionaries to evaluate all these aspects within explanations.

Though it is true that the use of questionaries is an approach to evaluate the aforementioned metric properties, one of the challenges is turning them into quantitative metrics for automatically assessing the explanations generated by XAI over a ML model. This is something that the literature is already addressing. The work of [17] shows how to use quantitative metrics for measuring some of the properties mentioned before. They first consider three families of metrics, ”explicitness”, ”faithfulness” and ”stability”. Then, they propose different algorithms to infer them, evaluating the results over different data sets.

The research at [18] also shows how to implement different metrics for quantitative measurement of the understandability of explanations. They use four families of metrics, ”comprehensibility”, ”representativeness”, ”stability” and ”diversity”. These metrics are calculated over local explanations used for explaining a blackbox model for anomaly detection, where the explanations considered are only focused in the outliers (in order to explain how to turn them to inliers). However, some of the metrics are ”explanation specific”, since they cannot be used for every type of explanations. The authors generate explanations using rule extraction techniques, and metrics like ”diversity” measure the degree of overlapping between the hypercubes generated. Hence, they only work for a particular type of explanations: local explanations trough rules. Other metrics, such as the ones for ”stability”, that measure how many similar data points are classified within the same category, are ”explanation agnostic” since they can be easily applied for other type of explanations, such as the case of feature relevance.

Finally, [14] also include explanations for measuring the model understandability for decision trees, in terms of number of interior nodes and number of leaves, and for measuring user understandability, through using online surveys and registering metrics different metrics. These last metrics include the response time that takes for a user to understand the decision tree, as well as if the users are able to predict the decision tree prediction for an individual data point, and their perceived understanding of the model through a rating given by them.

3. Method

In this Chapter we will introduce the flowchart followed by our proposal for the dynamic generation of explanations applied to anomaly detection of average fuel consumption. We will first describe the overall process, and then we will focus on each of the main modules.

3.1. Process overview

The overall process is detailed below in Figures 1 and 2, and in Figure 14 included within the Annex.

Figure 1 offers a summarized view of the training phase of the process, using high-level descriptions for the modules. Thus, at this phase, the process will start by selecting the relevant data for training from all the raw data available, and preprocess it in order to have the FAR data frame with the structure detailed in Section 3.2. This data frame will be both used for training the ML model and for detecting the vehicle-dates combinations (data points) that have an anomalous average fuel consumption in that day.

We want to quantify the feature value impact for the average fuel consumption of vehicles, and explain how they affect anomalous data points (Section 3.3). The purpose is to offer both explanations of which features are affecting and how much they affect, as well as recommending changes for features that are actionable in order to change the average fuel consumption to a value that is not anomalous (counterfactual explanations). Due to this, we train a regression ML model that infers the relationship between input features and output (average fuel consumption).

The next step is to identify those data points that are anomalous, and provide a visual explanation using a value limit to distinguish inliers from outliers (Section 3.4). Everything, from the model, to the historical preprocessed data used for training, and the limits that classify inliers versus outliers, is stored for its usage for the explanation phase.

Figure describes the overall modules involved for generating explanations dynamically. Selecting any period of data, the process extracts its corresponding FAR after the preprocessing module in case that period was not included within the training dataset described previously. Then, it identifies the outliers using the previous calculated limits. After it, it generates explanations for the outliers data points using the ML model and applying a posthoc XAI method in case the ML model is not a whitebox one (Section 3.5). These explanations are filtered in order to comply to specific business rules. With the final explanations, the process generates recommendations (Section 3.6) of data points that will have a change in their target variable if specific input features are changed. We only keep the recommendations that will lead to an average fuel consumption considered as inlier. Thus, the process provide the following explanations for outliers.

- **Visual explanations**: Limit value for the average fuel consumption that classify a data point as inlier or outlier.
- **Feature relevance**: For each outlier data point, it indicates which features affect the target and contribute to its increase. It also indicates the relative importance among them.
- **Recommendations**: It shows a counterfactual explanation that indicate feature changes that will lead an outlier data point to change and become an inlier.

Then, the process provides different metrics for the explanations and recommendations, as described in Section 3.7. Finally, at Section 3.8, the process adjust the recommendations for two user's profiles considered: technical specialists and fleet managers.

3.2. Data preprocessing

Obtain daily features

The first step within the preprocessing module is obtaining the daily aggregated information for each of the vehicles within the fleet. The IoT devices within the vehicles provide real-time information of the vehicle's status. However, for our proposal, we are interested in a daily vision of the vehicle. Thus, we aggregate the raw information into a set of features, described at the Annex in both Section 6.1 as well as in 2. The features chosen correspond to a business a

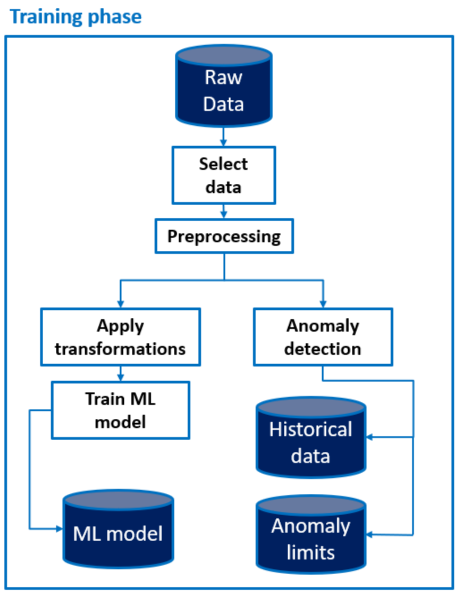

Figure 1: Overview of the training phase of the process.

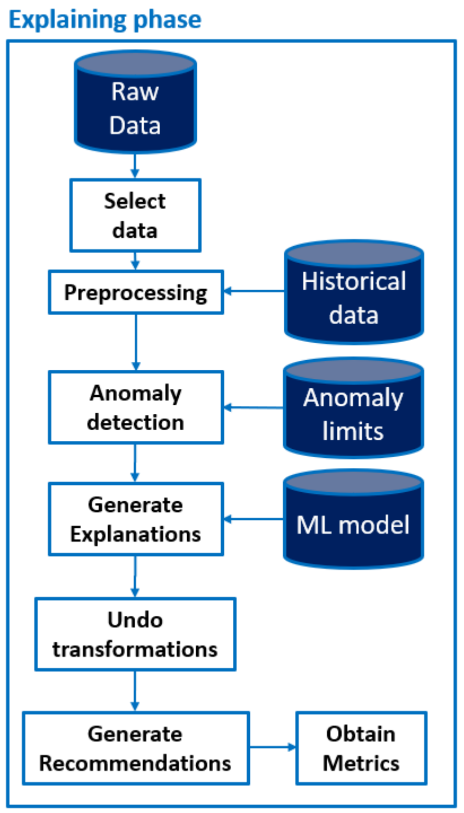

Figure 2: Overview of the explanation phase of the process

priori knowledge, since they may affect a vehicle's average fuel consumption. Some of the features also appear within the literature as potential causes of increased fuel usage [4].

At this point in the module, the initial data aggregation will include all the features described at 2 except for the ones under the "Categorical" category, since these features will be obtained at another step.

The features are divided into 4 groups: Index, Categorical, Explainable and Target.

Index features refer to features used to identify each row (namely a vehicle's unique id, vehicle_id, and the date, date_tx).

Categorical features refer to non numerical features used to distinguish group of vehicles (p.e. "vehicle_group" indicates vehicles with the same make-model). As mentioned before, they will be covered later, since they are not obtained yet at this point.

Regarding the explainable features, they are further divided into three groups. First, there are features related to the vehicle status itself. For instance, the pressure of the tyres. If the pressure is too low, the fuel consumption needed to cover the same amount of distance will increase, thus increasing the average fuel consumption of the vehicle. These features are identified in 2 as *Vehicle Parameters*. The next group of features are the *Driving Behaviour* ones. They correspond features related to the vehicle's driver behaviour itself that may affect the average fuel consumption. An example of these features is the idle time spent. More idle time may increase average fuel consumption. The last group of features considered are the *Environment Variables*. For instance, the exterior temperature may affect a vehicle's thermodynamic cycle harming its efficiency. Within these type of features there are also binary features (p.e. "lights_left_on").

The final feature is the target column, the average fuel consumption itself. This is calculated directly as:

$$avg\_fuel\_consumption = \frac{trip\_fuel\_used}{trip\_kms} \times 100 \quad (3)$$

This yields a data frame where each row corresponds to the daily aggregated values of the selected features for a specific vehicle. Thus, we want to analyse the potential relationship between those features with the average fuel consumption of that vehicle in that day.

It is worth mentioning that all the features at this point are going to be positive (value above or equal to 0). In most of the features this comes naturally (p.e. harsh brake events). However, in some other it is not true, mainly for the temperature. This is the reason why we use Kelvin scale at this point instead of Celsius when the range of temperatures provided include values below 0ºC.

Discard Y null registers

In some cases, the IoT devices may not provide either the fuel spent during that day (trip_fuel_used), the distance driven (trip_kms), or both. Then, for those cases we do not have the value of the average fuel consumption (target column, also known as Y column, referring as X the remaining features used as input). Those registers are not considered and are discarded.

Eliminate non relevant data

Within the initial processing of the data, non-representative vehicle-days are also eliminated, in which the distance driven is too low to be significant. A minimum threshold is defined that eliminates all vehicles that have a distance traveled less than that threshold. In addition, given that the information provided by IoT devices sometimes include erroneous data, in order to avoid including noise in the system, vehicles whose average fuel consumption is excessively high or low are eliminated within this step, taking as reference business values. Also, highly correlated numerical features are removed at this point.

Identify vehicle groups

The following steps aim to complete the previous features obtained from the IoT devices with relevant categorical features. The categorical features include two different variables. First, a feature named *vehicle_group*. This features classifies vehicles corresponding to their make-model. Using a vehicle's VIN (Vehicle Identification Number) we can identify their make and model, and group them accordingly. The VIn decoding procedure yields vehicle groups that do also have the same fuel type (diesel or gasoline; our data sets do not include electric or hybrid vehicles).

Identify route type

The second feature is *route_type*. It is used to identify the main type of route of a vehicle in a specific day. We assign a route type for each vehicle-day according to the following rules:

- IF $per\_time\_city \leq low\_th\_time$ AND $trip\_kms \geq th\_kms$ THEN $route\_type = hwy$
- ELSE IF $per\_time\_city \geq low\_th\_time$ AND $trip\_kms \leq th\_kms$ THEN $route\_type = city$
- ELSE $route\_type = combined$

Thus, we consider a "city" route if the vehicle spent a minimum amount of time driving within city and if the distance driven does not exceed a specific threshold. On the contrary, to consider the route of a vehicle-day as "highway" (hwy), the time spent driving within city should be lower than a threshold, and the distance driven should be above another threshold. Any other scenario is considered as "combined". This feature is important since the reference average fuel consumption of a vehicle is different depending on the route type since it impacts on other feature values (such as the average speed or the driving time).

Fill missing values

In addition to sometimes not having the data related to the target variable, in some cases the IoT devices do not send information about some of the input features. In order to avoid losing excessive registers and maintain a statistically significant set of data, these values are imputed with inferred values from the rest of the fleet. Separating the data set according to its vehicle_group, each missing value is assigned with the median value of that feature in order to be able to maintain the record but that the value of that variable for that vehicle-day is not significant to the model.

This module provides a final data frame ready to be used in the following modules. We will further address this data frame as FAR (Fleet Analytics Record).

The median values considered are from the historical dataset using during the training phase. Since the periods of time that are used for the explanation may be different from the historical dataset, the median values used to fill null values for explanations are the ones obtained during the training phase.

3.3. Unsupervised anomaly detection

Using the previous FAR data frame, the next module is responsible for detecting the vehicle-dates where there is an anomalous average fuel consumption. Since there is no prior knowledge on when the average fuel consumption is anomalous, the module needs to detect it in an unsupervised manner. Also, the module needs to provide a threshold value to distinguish outliers from inliers, since we want to include that information as a visual explanation.

To comply with both requirements, we apply an univariate unsupervised anomaly detection approach using a Box-Plot that classifies data points as outliers if they are above or below the following thresholds:

$$\begin{aligned} lim\_sup &= Q3 + 1.5 \times IQR \\ lim\_inf &= Q1 - 1.5 \times IQR \end{aligned} \tag{4}$$

However, we will only consider as outliers those vehicles above the superior threshold, thus, not considering as outliers those below it.

The Box-Plot will be applied over the different combinations of the categorical variables (make-model with vehicle_group and route type with route_type), obtaining then a different limit depending on the combination considered.

With that, the output of this module is the original FAR with both the limit that classifies data points as inliers or outliers, and a binary column indicating whether that data point is actually an outlier. The limits are obtained during the training phase. For the explanation phase, the limits inferred previously for each vehicle_group and route_type are used to identify outliers within the dataset that wants to be explained.

3.4. ML model

The following module is the training of a ML supervised model that finds relationships between the explainable and categorical features from the FAR dataset and the target variable.

Since part of this paper is benchmarking different proposals for XAI for local explanations based on feature relevance, this module can use different ML supervised algorithms. Regarding whitebox models, our proposal includes three options. First, we want to evaluate the usage of EBM [11] since they offer both the possibility to infer relationships between the input features and the average fuel consumption while providing feature relevance values that show the contribution of each feature to the final prediction for every data point. At this point, in order to offer a baseline benchmark, we will also include a Linear Regression model with the usage of the ElasticNet [19] algorithm. Also, we will include a variation over EBM that will be addressed as "EBM variation". It will be described at the end of this Section. For the black box models, we will include the tree based methods that we are going to use later for benchmark against EBM. They include XGBoost [20] and LightGBM [21].

Our final solution will use the proposal that yield best results (according to the metrics defined at 3.7.). There are two additional aspects to consider in this module, shown at the detailed flowchart in Figure 14 within the "Adjust features" module. First, some of the algorithms aforementioned need to have all the features within similar value scales. Thus, we apply a standarization over the input features for two scenarios: when using ElasticNet and when the posthoc XAI method is LIME. Finally, the evolution of some the feature values according to the evolution of the average fuel consumption should be monotonic (either positive or negative), like is indicated within the column "Type" in Table 2. For some of the ML algorithms, like LightGBM or XGBoost, we can specify as an input parameter if we want any monotonic constraint (either positive, negative or none). However, we cannot do this directly with EBM or ElasticNet. In the case of ElasticNet, the monotonic constraint consists in enforcing the coefficients to be positive, but that does not work for monotonically decreasing features. Thus, for these last two algorithms, we simply change the sign of the features in order to make them negative and reverse the dependence with the target variable.

EBM variation

The EBM variation that we propose that takes into account possible differences that may exists within different subgroups of vehicles, in order to adjusts feature relevance and predictions. Regarding our use case, the feature relevance may be different depending on the vehicle group. For instance, the impact on the average fuel consumption for each additional harsh brake may change depending on the vehicle's model and make considered. Thus, there should be different feature relevance-values pairs depending on that vehicle group category. Using only one EBM provides unique pairs of value-importance regardless of the vehicle group, meaning that the final impact in the target variable will be the same for a specific feature value.

The intuition behind our proposal is similar to other works in the literature [22]. We will add an additional layer of models to predict the error of a previous one. As represented in Figure 3 for one subgroup of vehicles, first, we will train an EBM model over all data during the training phase. Then, we will predict the error for each of the vehicle's subgroups, and train additional EBM in order to be able to predict that error and both improve the predictions of the first one as well as adjusting the results to the specificity of each of the subgroups. This last consideration is based on the fact that while the first model provides unique feature relevance-values pairs, because the second one is predicting the error of the first one in order to add it to its prediction, we can also use the feature relevance values of the second one to add them to the first one. This may be done since the feature relevance values of the second model show the feature contribution to the error. With that, there will be different feature relevance-value pairs, as well as predictions, for each of the vehicle subgroups considered.

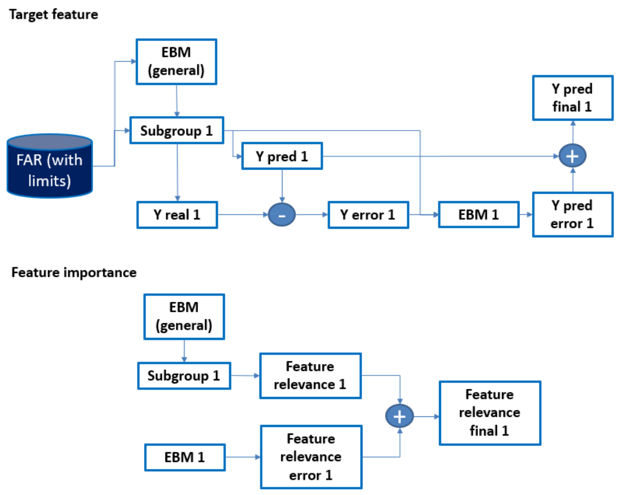

Figure 3: Proposal of the "EBM variation" over only one subgroup.

The detailed description of EBM variation appears at Algorithms 1 and 2. Algorithm 1 describes the training process. The function trainEBMvar receives the input feature matrix X together with the real target variable y, and a list with the columns used to consider the subsets, l_s. In this case, l_s includes only the variable vehicle_group. After that, it initializes an empty dictionary dct_m where the error predicting models are going to be stored. Then, it obtains the potential combination of l_{comb} (in this case, there are no combinations since there is only one feature). Following this, it trains an EBM model using X and y. Iterating through all of the combinations, it filters the input matrix X for the subset for that iteration, X_i, getting also the indexes associated to those registers, idx_i. If there are not enough data points (less than a threshold th_ebm_var), it skips that iteration. In other case, it obtains the error for that subset using the original model emb, y_err_i. Using that error and the matrix filtered for that iteration, it trains a new model ebm_i that tries to predict the error for that subset. This model is stored within the dictionary dct_m.

After the training, the next step is using those models for prediction and explanations. Algorithm 2 describes the function expEBMvar used for that purpose. It receives a data frame to explain (X), together with the general model (ebm), and the dictionary with the models used for error prediction (dct_m). It also receives the list of features for the subsets of data. The function initialize a data frame to store the feature relevance values (df_imp) and a list with the target feature predictions (y_pred). After obtaining the different combinations for iterating (l_{comb}), it firsts predicts the target feature for that subset X_i using the general model ebm. Then, if that combination was used for training error predicting models, it obtains the error predictions of the subset, together with their feature relevance values, and adds them to the ones from the original model. If that combination does not belong to any error predicting model, then the function uses only the predictions and feature relevance values from the general model (ebm).

Algorithm 1 EBM Variation training

```
1: procedure TRAINEBMVAR($X, y, l_s$)
2:     $dct\_m \leftarrow null$
3:     $l_{comb} \leftarrow combinations(X, l_s)$
4:     $ebm \leftarrow trainEBM(X, y)$
5:     for $comb \in l_{comb}$ do
6:         $X_i \leftarrow X[X[l_s] = comb]$
7:         $idx_i \leftarrow X_i[index]$
8:         if $len(X_i) < th\_ebm\_var$ then
9:             continue
10:        end if
11:        $y\_pred_i \leftarrow ebm.predict(X_i)$
12:        $y\_real_i \leftarrow y[idx_i]$
13:        $y\_err_i \leftarrow y\_real_i - y\_pred_i$
14:        $ebm_i \leftarrow trainEBM(X_i, y\_err_i)$
15:        $dct\_m[comb] \leftarrow ebm_i$
16:    end for
17:    return $ebm, dct\_m[comb]$
18: end procedure
```

Algorithm 2 EBM Variation explanations

```
1: procedure EXPEBMVAR($X, ebm, dct\_m, l_s$)
2:     $df\_imp \leftarrow null$
3:     $y\_pred \leftarrow null$
4:     $l_{comb} \leftarrow combinations(X, l_s)$
5:     for $comb \in l_{comb}$ do
6:         $X_i \leftarrow X[X[l_s] = comb]$
7:         $y\_pred_i \leftarrow ebm.predict(X_i)$
8:         if $comb$ in $dct\_m$ then
9:             $ebm_i \leftarrow dct\_m[comb]$
10:            $y\_err_i \leftarrow ebm_i.predict(X_i)$
11:            $y\_pred_i \leftarrow y\_pred_i + y\_err_i$
12:            $df\_imp\_i \leftarrow ebm.feat\_imp(X_i)$
13:            $df\_imp\_err\_i \leftarrow ebm_i.feat\_imp(X_i)$
14:            $df\_imp\_i \leftarrow df\_imp\_i + df\_imp\_err\_i$
15:        end if
16:        $y\_pred \leftarrow y\_pred.append(y\_pred_i)$
17:        $df\_imp \leftarrow df\_imp.append(df\_imp\_i)$
18:    end for
19:    return $y\_pred, df\_imp\_i$
20: end procedure
```

3.5. Generate explanations

The generation of explanations belongs to the explaining phase. First, we apply a preprocessing just like the one during the training phase, with the minor difference that the median values used for filling null features correspond to the historical feature value (feature value during the training phase). After that, using the average fuel consumption limits obtained during the training phase, we classify each data point as outlier or inlier. As a result, at this point the process will use for explanations an input data set analogous to the one used for training. This data set, however, will be filtered for only outliers, since the aim of the process is to explain anomalous average fuel consumption cases.

The generation of explanations will also use the regression model trained in the previous phase, so the data set aforementioned will have it's features adjusted ("Adjust features" module), scaling its values if needed according to the historical values used during training, and changing some feature signs depending on the ML regression algorithm used.

Within the "Generate explanations" module itself, the first step checks whether the ML model used for regression is a whitebox model or a blackbox one. In case the model is a whitebox one, the feature relevance for each data point can be extracted directly. On the contrary, if the model is a blackbox one, a posthoc XAI method for feature relevance is applied over it in order to extract that feature relevance for each data point included in the data set use for explanations. The posthoc XAI methods considered are tree-based SHAP algorithm [9], and LIME [6].

This provides a raw data frame with explanations that could be used directly to explain every instance (Table 2). However, it needs to be combined with the business rules in order to select not all the explanations, but the ones that comply with them. Since every XAI method considered in this paper establish an additive relationship between input and target features through feature relevance value, from among all available features the process considers only for each data point those that comply with the rules. Generally speaking, for a particular data point *n*, the raw explanations provide the following equations:

$$y\_pred(n) = \varepsilon + \sum_{i=1}^{k} \alpha_i(x_i(n)) \times x_i(n) \qquad (5)$$

Thus, Equation 5 show the relationship for a data point *n* between the predicted value of the target variable y_pred with respect to k input features, x_i, through their coefficient α_i. This coefficient α_i changes depending on the corresponding $x_i(n)$ value for that data point in case of EBM method or "EBM variation" (for whitebox) and the posthoc XAI techniques of tree-based SHAP and LIME (for blackbox). Regarding the baseline linear model (ElasticNet), the coefficient is constant for every data point. Finally, a constant intercept value ε is added to the feature terms.

There are two business rules filter steps that area applied over the raw explanations. First, there is a "Monotonicity filter" to ensure that feature relevance are monotonic. Then, there is a "Business Rules filter" step that apply the remaining business rules. This split of rules is useful since the monotonicity filter will always be needed within the process, but the remaining business rules may change according to customer needs or customer profile.

The "Monotonicity filter" step analyses each pair of feature value and feature relevance for every vehicle group and route type combination and discards the pairs that are not monotonic. An example of it can be seen in Figures 4, 5, 6. Starting from the evolution of the relevance-value pair of a particular feature, in this step the process finds the feature values intervals where the feature relevance is not monotonic, and discards those combinations. Thus, the raw explanations for each vehicle-day, where all the features are included, are filtered so that the feature values that correspond to feature relevance ones that are not monotonic are now discarded. Figure 4 shows the original feature relevance-value pairs for a combination of route type and vehicle group for the feature count_harsh_brakes. As the Figure shows, the evolution is not monotonic.

Figure 5 shows precisely those intervals where the importance decreases while the value increases. Thus, this step removes those importance-value pairs from all the raw explanations for every data point belonging to that vehicle group and route type combination. After removing those intervals, the evolution is indeed monotonic, as Figure shown in 6.

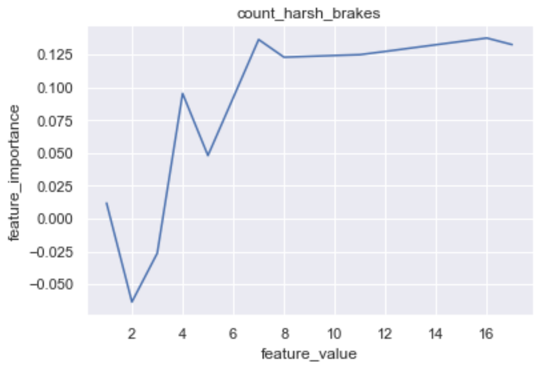

Figure 4: Example of evolution of the feature value and the feature relevance for feature count_harsh_brakes where the evolution is not monotonic.

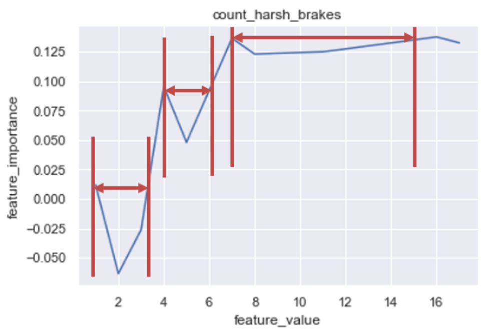

Figure 5: Example of evolution of the feature value and the feature relevance for feature count_harsh_brakes showing the intervals that make the evolution not monotonic.

Formally, the step analyses the evolution of the relevance-value pair of every feature for every combination of categorical features as indicated in Algorithm 3. The function "ensureMonotonic" receives four variables: X_i with the FAR data frame that wants to be explained, X_{exp} with the raw explanations generated previously, l_e with a list of the numerical features (the ones for analysing the monotonicity),

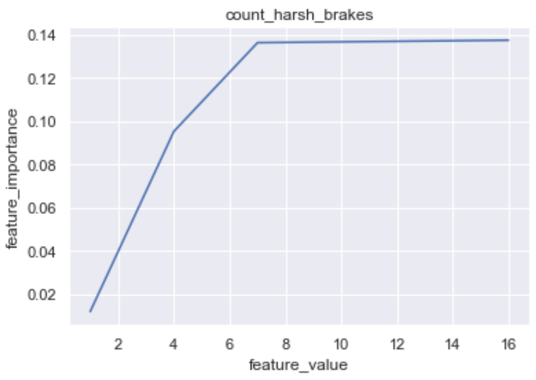

Figure 6: Example of evolution of the feature value and the feature relevance for feature count_harsh_brakes where the evolution is monotonic after removing the appropriate pairs.

and l_c with a list of the categorical columns. Using both X_i and l_c, the function first obtains the possible combination of categorical features and stores that information within l_{comb}. Thus, l_{comb} and l_e are the parameters that are going to be considered during each iteration: a unique combination of categorical feature values ($comb$) and one explainable feature (f). $comb$ and f are used for filtering the explanations of every vehicle-date of the period in order to have a unique data frame of the importance-value pairs inside that iteration (X_{check}). This data frame is sorted in an ascending order using the feature value. After that, the function gets the difference of the feature relevance between one feature value and the following one. If the evolution is monotonic, the difference should be 0 or higher (0 because we only check for monotonic evolution, not strictly monotonic). The function discards the rows that are not monotonic, and keeps checking the difference of feature relevance between one row and the following one until no rows are discarded (which means that the data frame is already monotonic).

Since the monotonicity filter analyses the combined evolution of both feature relevance and feature value, it works either for EBM (where there is only one value-importance pair per feature at the pairwise function [11]), "EBM variation" (where there is potentially one value-importance pair per feature and vehicle group), or LIME and SHAP (where there may be more than one importance value per unique feature value [5]). Indeed, as shown in Figure 7, there may be more than one importance-value pair per feature value. However, since Algorithm 3 checks a pair and the immediate following one, it will, for instance, check $(x0, y0)$ against $(x0, y1)$ with $y1 > y0$, and will remove the latter if the importance is lower. After removing the non monotonic pairs, Figure 7 turns to Figure 8.

A final comment is that, in some cases, there may be only one feature relevance-value pair whether because there was only one instance to begin with, or whether there is only one remaining instance after applying the filter. In those cases, the instance is kept.

After applying the monotonicity filter, the "Generate explanations" module applies the remaining business rules at the "Business Rules filter" step. Though, as mentioned before, the business rules may vary, for the purpose of this paper and it's analyses, the following business rules are considered:

Algorithm 3 Monotonicity

```
1: procedure ENSUREMONOTONIC($X_i, X_{exp}, l_e, l_c$)
2:     $X_{exp\_new} \leftarrow null$
3:     $l_{comb} \leftarrow combinations(X_i, l_c)$
4:     for $comb \in l_{comb}$ do
5:         for $f \in l_n$ do
6:             $X_{check} \leftarrow filter(X_{exp}, comb, f)$
7:             $X_{check} \leftarrow dropDuplicates(X_{check})$
8:             $X_{check} \leftarrow sort(X_{check})$
9:             $n_{diff} \leftarrow -1$
10:            while $n_{diff} \neq 0$ do
11:                $n_i \leftarrow len(X_{check})$
12:                $X_{check}['diff'] \leftarrow getDiff(X_{check})$
13:                $X_{check} \leftarrow X_{check}['diff'] \geq 0$
14:                $n_e \leftarrow len(X_{check})$
15:                $n_{diff} \leftarrow n_i - n_e$
16:            end while
17:            $X_{exp\_new} \leftarrow append(X_{exp\_new}, X_{check})$
18:        end for
19:    end for
20:    return $X_{exp\_new}$
21: end procedure
```

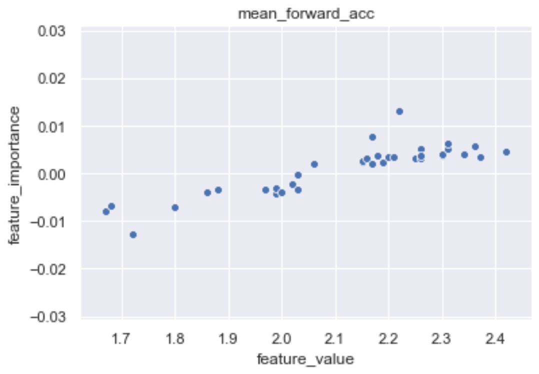

Figure 7: Example of evolution of the feature value and the feature relevance using SHAP for feature mean_forward_acc showing not only that the evolution is not monotonic, but that sometimes there are more than one feature relevance value per feature value

- BR1: Do not use the feature relevance of "general" features within the explanations.
- BR2: Feature value should be higher than the median value of inliers for that combination of "general" features for features with monotonic positive constraint, or lower for features with monotonic negative constraint.

The reason behind the usage of BR1 is that combinations of categorical columns are only going to be used to divide the set of explanations into the specific combination for

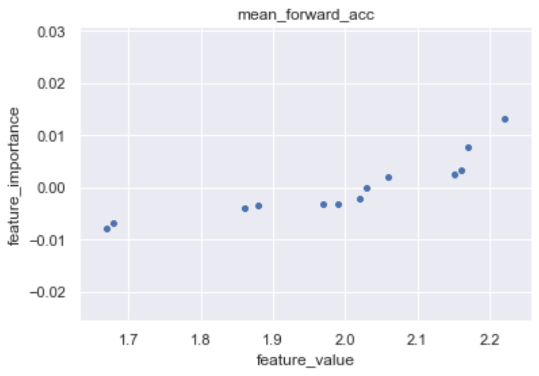

Figure 8: Example of evolution of the feature value and the feature relevance using SHAP for feature mean_forward_acc where the evolution is monotonic after removing the appropriate pairs.

each one of those categories. For instance, the user will see which vehicles belonging to vehicle_group 3 and for a route type of "city" have an anomalous average fuel consumption and which features are affecting it and how much. However, there is no need of indicating the feature relevance itself of the categorical features, so those registers are discarded from the explanations obtained in the previous step.

BR2 further filters the explanations to consider only the cases where outliers have a feature value above the median feature value of a reference data set of vehicle-date inliers for the same categorical combination. This applies to monotonic positive constraint features. For monotonic negative constraint features the logic is similar but the feature value should be lower than the median one. Regarding the reference data set, is either the historical one used in the training phase for features with "No" at "Period only?" column in 2, or the one used in the explaining phase itself for features with "Yes". It is like this because for features such as the exterior temperature or the total odometer value there is no sense in comparing against their historical value, since they are features that are either continually increasing or are seasonal.

With that, the output for this module includes the feature relevance (after filtering with monotonicity and business rules) for each of the anomalous vehicles and dates included within the date range considered for the explanation phase. A final step is applied, where any data transformation applied before (such as feature scaling or reverse signs) is undone.

3.6. Generate recommendations

As mentioned at the Introduction, whitebox models that include feature relevance are useful for counterfactual explanations. Since there is a unique intercept and unique feature relevance-value pairs, they can provide counterfactual explanations where one of the feature values alone may be changed and with that, recalculate the predicted target value in order to see how it will change. With both SHAP and LIME, we would need to obtain again the whole explanations for the new data point with the modified feature value, and that may lead to a feature relevance change for the remaining features. Thus, the approach will be different. This is why the "Generate Recom." (generate recommendations) step will only be offered for whitebox models (EBM, "EBM variation" and ElasticNet).

The intuition behind it is the following one. "Generate Recom." will change the feature values of the outliers used within the explaining phase for the corresponding median feature value of the inliers belonging to the same vehicle group and route type. This will be applied for one feature at a time and for every feature labeled as "actionable". Then, by substracting the relative change in the predicted value from the real average fuel consumption, it will indicate which vehicles-dates would have a fuel consumption below the outlier limit for that vehicle group and route type.

The details are described in Algorithm 4; getRecom function receives the historical median values of the inliers (obtained during the training phase; X_{med}), the data points of the explaining phase with their feature relevance (X_{exp}), and two lists, one with the explainable features that are actionable (l_a) and one with the categorical ones (l_c). Using these inputs, getRecom function initializes two empty lists (l_up_ind and l_up_all) and gets the feature relevance for the median inliers feature values ("coeff") with $checkPairwise(X_{med}, l_c)$ function. After obtaining the feature relevances, the function analyses every data point (x) within the explanations and obtains it's predicted target feature (y_pred) using the feature relevance and the intercept. It also stores the real value (y_real) of the target feature. Then, it checks every feature (f) within the explanations and gets its corresponding feature relevance from the median inliers reference (β_{fn}). It laters sums again al the feature relevance and intercept for data point x, without the feature relevance for feature "f", and instead sums β_{fn}. This leads to a new predicted value (y_new) where all the other feature values are kept the same, but there is a change for the specific feature considered. The difference between y_pred and y_new is Δ, and this difference is used to compute the change in the real average fuel consumption (l_up_ind). After iterating for all the available combinations, getRecom uses groupVal function to obtain the estimated value in case all the actionable features change at the same time to their median inlier value. This is simply done by aggregating all the individual changes in the prediction for each feature, and subtract the aggregated difference from the real average fuel consumption.

Thus, Algorithm 4 provides a list with the new estimated average fuel consumption value for every individual feature change for every vehicle-date pair (l_up_ind). Comparing this values against the outlier limit for that vehicle group and route type, the step indicates which individual feature changes will lead from outlier to inlier, and what would be the corresponding average fuel consumption. It provides as well a similar result but considering that every actionable feature changes at the same time (l_up_group).

Algorithm 4 Generate Recommendations

```
1:  procedure GETRECOM($X_{med}, X_{exp}, l_a, l_c$)
2:      $l\_up\_ind \leftarrow null$
3:      $l\_up\_all \leftarrow null$
4:      $coeff \leftarrow checkPairwise(X_{med}, l_c)$
5:      for $x \in X_{exp}$ do
6:          $y\_pred \leftarrow \varepsilon + \sum_{i=1}^{k} F_i(x_i)$
7:          $y\_real \leftarrow x[target]$
8:          $comb \leftarrow x[l_c]$
9:          for $f \in l_a$ do
10:             $\beta_{fn} \leftarrow coeff[f]$
11:             $y\_new \leftarrow \varepsilon + \sum_{i=1}^{k \neq f} F_i(x_i)$
12:             $y\_new \leftarrow y\_new + \beta_{fn}$
13:             $\Delta \leftarrow y\_pred - y\_new$
14:             $y\_updated \leftarrow y\_real - \Delta$
15:             $l\_up\_ind \leftarrow l\_up\_ind.append(y\_updated)$
16:         end for
17:     end for
18:     $l\_up\_group \leftarrow groupVal(l\_up\_ind, l_a, X_{exp}, l_c)$
19:     return $l\_up\_ind, l\_up\_group$
20: end procedure
```

3.7. Metrics

As shown in Figure 14, there are two steps where the process computes metrics for evaluation. The first group of metrics, obtained during the training phase, aim to benchmark the predictive power of the different models considered. These metrics will analyse the average fuel consumption predicted by all those models against its real value. We will further refer to them as *model metrics*.

The second group of metrics are the ones obtained during the explaining phase, and they aim to measure different aspects regarding the understandability of the explanations generated, as well as comparing the explanations generated between every model and XAI technique. We will further refer to them as *XAI metrics*.

With that, we are analysing not only if the predictive power of the EBM is good enough (model metrics), but we are also measuring the explanations themselves in order to compare them to against posthoc XAI techniques (XAI metrics). Thanks to that, we offer a whole comparison of the usage of EBM with real-world IoT data for outlier explanation against using blackbox models with posthoc XAI techniques. Again, the same is applicable for the "EBM variation" proposal. A final comment is that even tough it may be difficult (or not reliable) to compare individual explanations with certain metrics due to Rashomon's Effect [5], the ones that we propose analyse the explanations from a general perspective. Thus, even if the explanations differ at a very low level, the general view should be similar.

Model metrics
These metrics include the following:

- explained variance (EV)
- maximum error (ME)
- root mean squared error (RMSE)
- median absolute error (MAE)

The metrics above are the one used for comparing the models among themselves. Together with that, we analyse over the test set if the models are good enough. For doing that, we will consider two metrics:

- Adjusted R2 (adj-R2)
- mean absolute percentage error (MAPE)

All the model metrics are evaluated over a test set that includes both outliers and inliers, since the purpose is to measure how close are the target feature predictions to the real value. There are other potential metrics that can be considered, especially classification metrics that measure if after applying the anomaly limits over the predicted values, the inlier/outlier predicted class matches the one of the real target feature. However, since we are not using the ML surrogate model to actually predict the outlier/inlier class, we do not find it necessary to measure.

XAI metrics
Using the taxonomy of metrics present in [15] for individual explanations, we consider the following properties for comparing the explanations generated by the different methods studied in this paper.

Representativeness metrics include two subgroups: general metrics and monotonicity metrics. *General metrics* measure global aspects regarding the explanations generated. They include the following ones:

- **n_datapoints**: Number of unique combinations of vehicles-dates (data points) within the explained data frame.
- **per_datapoints_explained**: Percentage of explained data points from the total n_datapoints.
- **n_variables_used**: Number of features used for the explanations.
- **mean_variables_used_per_day**: Mean daily features used for explanations.
- **mean_variables_used_per_group**: Mean features used for explanation per group.

Monotonicity metrics measure the impact of the monotonicity filter over feature the importance-value pairs. Since for EBM (or "EBM variation") we are not explicitly applying any constraint for feature monotonicity, these metrics will offer a comparison against XGBoost and LightGBM where the constraint was applied in order to see if there are any significant changes. These metrics also allow to see how SHAP and LIME respect the constraints existing in the model since at an ideal scenario there will no discarded pairs.

- **per_monotonic_datapoints**: Percentage of data points that are kept after applying the monotonicity filter. The data points considered are the pairwise relationships (feature relevance-value) separated by vehicle group and route type. The percentage is obtained by seeing how many data points remain for each of the subgroups with respect to the total data points of all of those subgroups.

$$\hat{L}(x_i) = \underset{x_j \in B_\epsilon(x_i)}{\operatorname{argmax}} \frac{\|f_{\text{expl}}(x_i) - f_{\text{expl}}(x_j)\|_2}{\|h(x_i) - h(x_j)\|_2}$$

Figure 9: Stability metric as appear in [17]

Fidelity metrics focus on comparing the output value for the target variable after applying the business rules against its previous value, in order to see which model is less penalized by applying them. It will include two kind of metrics, depending on whether the output comparison is against the real value of the target variable, or if it is against the predicted value of the surrogate ML model.

The first subgroup of metrics within *fidelity* are identified as fidelity-target metrics. *Fidelity-target* measure the predictive power for each data point considering only the feature relevance of the remaining explanations after applying all the filters. Of course, these metrics do not represent by themselves any insights for the real surrogate ML model (since they may use training data and since they do not account for all the raw feature relevance). They are useful only to see if there are significant changes between the surrogate model-XAI combinations. Also, these metrics only make sense for surrogate model-XAI combinations that do not have significant changes in their ”General metrics” (if one combination yields significantly less explanation than other, it is not possible to compare this subgroup of metrics). Finally, these metrics are not useful to compare ”EBM variation” since it tries to reduce the training error, and as mentioned before, the data sets used for explanations may contain training data. The metrics themselves are some of the ones used for ”Model metrics”, but calculated for each data frame used for the explaining phase.

- mean average precision error (MAPE)
- maximum error (ME)
- root mean squared error (RMSE)

The last subgroup of *fidelity* metrics are identified as *fidelity-model*. They include a metric called ”faithfulness_error” which calculates the Absolute Error (AE) between the predictions before and after applying the business rules for whitebox models, and after applying the XAI method together with the business rules for blackbox ones. This metric will be further identified as ”faithfulness_error”

Stability metrics includes two metrics, identified as *stability_error* and *xai_stability_error*. Both of them are computed using the stability metric proposal of [17], as indicated in Figure 9. For *stability_error* we analyse the stability of the predictions before applying the business rules (raw predictions of the whitebox models and predictions from the blackbox ones). For *xai_stability_error* we analyse the predictions using the feature relevance values from the resulting explanations (after applying the business rules). Both of these metrics are calculated over the test set for each of the data points.

Finally, following also [15], we include metrics related to the usefulness of the explanations generated, measuring **contrastiveness**. For doing it, we include the metric *per_rec_below*, that measure the percentage of outlier data points that receive a countefactual recommendation that changes the values of the final features into the median value of the same feature values for the inliers of that group, and doing that changes it the average fuel consumption into an inlier. The features included are the ones identified as ”actionable” at Table 2. Also, the recommendations are the last step of the process, so the features used have already passed by the business rules filters. Thus, we are measuring the understandability of explanations from the user's perspective: considering only features that the user can alter directly, how many instances can significantly decrease their average fuel consumption?

3.8. Recommendations according to user profiles

As mentioned in the previous Chapter, [1] highlight that explanations should be tailored for the specific profile of the user that will receive them taking into account both their expectations and their domain knowledge.

Within the use case proposed in this paper, we identify two user's profiles, as indicated in Figure 10, where the users are highlighted over the image from [1].

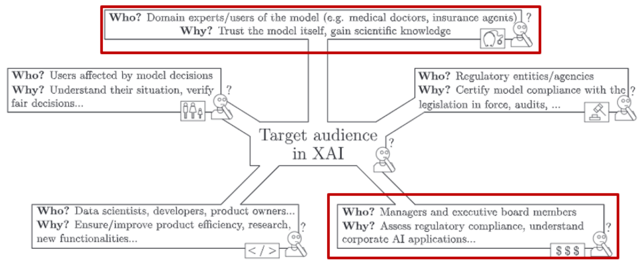
Who? Domain experts/users of the model (e.g. medical doctors, insurance agents)
Why? Trust the model itself, gain scientific knowledge
Who? Users affected by model decisions
Why? Understand their situation, verify fair decisions...
Who? Regulatory entities/agencies
Why? Certify model compliance with the legislation in force, audits, ...
Target audience in XAI
Who? Data scientists, developers, product owners...
Why? Ensure/improve product efficiency, research, new functionalities...
Who? Managers and executive board members
Why? Assess regulatory compliance, understand corporate AI applications...
</ >
$ $ $

Figure 10: Relevant user profiles for this use case, following the proposal of [1]

User Profile 1: Technical Specialists The first group of users are technical specialists, responsible for the status of the vehicles. Their main interest regarding the explanations is detecting what vehicles are consuming excessively, and what is causing it, considering for that not every feature, but only the ones that are actionable (as seen in Table 2).

To accomplish that, the explanations generated at Section 3.5. may be enough. However, explaining every single date for each combination of vehicles and route types in terms of the numeric feature relevance is overwhelming, not being useful for them. This is why we provide the explanations for these users at two different levels. First, a summary of the main recommendations for a specific period of time (p.e. a month). Second, we provide the individual daily detail only if the want to dive deeper into a particular vehicle and route type.

First level - Summary of recommendations
As already mentioned, the first level include a summary of the individual recommendations yielded by the system. Algorithm 5 described the way to accomplish it. First, it receives the same input, $X_{med}, X_{exp}, l_a, l_c$, as Algo-

rithm 4. The difference is that, before obtaining the recommendations, it applies a filter that choose only some vehicles and route types, from among all the combinations, according to some business parameters. These parameters are $min\_days\_anomalies$, min_day_km, and $min\_dev\_total\_avg\_fuel$. With $min\_days\_anomalies$, the filter chooses only the vehicle-route type combinations that have at least that specified number of outliers. Then, with min_day_km, it chooses only the dates that have a trip distance over that minimum threshold. Finally, with $min\_dev\_total\_avg\_fuel$, the filter chooses only dates with individual recommendations that have a decrease in the target variable after applying the recommendations over that threshold.

After applying the aforementioned filters with $filterPoints()$ function, the algorithm applies another function, $summaryPoints()$. This function aggregates the remaining individual data points of the outliers into their median values. So, it will yield a data frame with unique points for each combinations of vehicles-route type. These points will represent a prototype for each of those combinations, representing the most common anomalous scenario. These data points are stored in X_{summ}. In order to always have feature values already present within the explanation period, if the vehicle values are pairs (not odds) we keep the lowest middle value in order to offer later the most conservative recommendation. Then, the algorithm uses X_{summ} for obtaining the recommendations with $getRecom$ function. In this case, we are only interested in the output l_up_ind, that indicates the new average fuel consumption after applying each individual feature change in order to have the median inlier value.

This individual contributions are aggregated with $aggContribution()$ function, providing l_agg with the total average fuel consumption reduction if all the features had the value of the median of the inliers of that same vehicle group and route type. With that, the user will see the general recommendations (how much average fuel could be decreased by applying all the feature value changes), as well as seeing the individual impact of each feature to the average fuel consumption (seeing how much average fuel consumption could be reduced by applying only one feature change).

Algorithm 5 Summary of Recommendations

```
1: procedure GETSUMMARYRECOM($X_{med}, X_{exp}, l_a, l_c$)
2:     $X_{exp} \leftarrow filterPoints(X_{exp})$
3:     $X_{summ} \leftarrow summaryPoints(X_{exp})$
4:     $l\_up\_ind, \_ \leftarrow getRecom(X_{med}, X_{summ}, l_a, l_c)$
5:     $l\_agg \leftarrow aggContribution(X_{summ}, l\_up\_ind)$
6:     return $l\_up\_ind, l\_agg$
7: end procedure
```

Second level - Daily detail

As mentioned before, we first provide the summary of recommendations and then, if the user wants to see recommendations for individual days, they can access to these second explainability level. At this level, the explanations provided to the user contain two elements. The first one includes the recommendations from Section 3.5. For the second element, we provide the daily feature relevance of every variable (not only the actionable ones). However, directly providing the feature relevance value is also not useful since it is cumbersome to directly analyse it in order to see how the relatively influence the average fuel consumption. Because of this, we complement the quantitative feature relevance explanations with qualitative ones. Considering additional business rules, the module classifies the degree of influence for each feature at a specific vehicle-date depending on how much they contribute to average fuel consumption. In order to do that, the module uses the following business rules:

- IF $var < th\_degree\_1$ THEN $var\_cat =$ no influence
- ELSE IF $var < th\_degree\_2$ THEN $var\_cat =$ low influence
- ELSE IF $var < th\_degree\_3$ THEN $var\_cat =$ medium influence
- ELSE $var\_cat =$ high influence

With var corresponding to:

$$var = \left| \frac{feature\_importance}{y\_pred} \right| \tag{6}$$

Thus, dividing each feature relevance value with the predicted average fuel consumption, it gets the relative contribution of each of those features. Thanks to that, the user can see which features where the ones that contributed the most.

User Profile 2: Fleet Manager The final user profile considered is the "fleet manager". The main interest for this user profile is having a global comparative view at a vehicle group level, not seeing information about individual vehicles or particular dates. At this level of information, in order to have useful explanations, the individual ones must be aggregated into explanations at vehicle group level, like is done with l_up_group from Algorithm 4. However, offering explanations in terms of anomalies and average fuel consumption is not what is expected. The useful explanations should be expressed in terms of extra litres of fuel consumed, because that can be immediately turned into an economic cost. So, after having the individual recommendations from Algorithm 4, the individual explanations are aggregated in order to first have the total average fuel consumption reduction per day, and they are later expressed in terms of total fuel, calculating it according to the new feature values (because it will depend on the new value of "trip_kms"). Then, with all that, the final explanations provide the vehicle group view (with l_up_group) and how much fuel could have been saved without anomalies (both at a global level and per each vehicle group).

4. Evaluation

We use our algorithm over different IoT data sets from Telefonica's, to evaluate the following hypotheses:

- It is possible to obtain similar model metrics using EBM compared to other boosting models (XGBoost and LightGBM). EBM metrics will also be significantly better than other whitebox model used as baseline: ElasticNet.

- Model metrics obtained by evaluating EBM over a test set will be good enough, showing that we can use this model for the use case described in this paper.
- It is possible to obtain local explanations using EBM that are similar to the ones obtained with blackbox models combined with local posthoc XAI techniques based on feature relevance (LIME and tree-based SHAP). This will be evaluated with the XAI metrics described in the previous Chapter: "general", "monotonicity", "fidelity-target" and "degree of influence". This will also show how EBM, regardless of not including monotonicity constraints, will yield similar results than blackbox models with those posthoc XAI techniques where the models do indeed include the constraints. Also, XAI metrics will be significantly different than those obtained by the baseline model.
- XAI metrics obtained with EBM are good enough to use it for explanation generation for the use case described in this paper.
- Our proposal "EBM variation" obtain metrics similar to standard EBM, so it can be used to have different feature relevance-value pars per vehicle group without losing neither explainability nor predictive power.

4.1. Data sets

As mentioned before, the IoT are going to be processed at the preprocessing phase of 14 in order to obtain a data set with the features described in 2, where their values represent the daily aggregation of the real-time information of a vehicle at a specific date. For the analyses carried out in this paper, we will consider 3 data sets, belonging to different fleets. This data sets are samples for some of their vehicles, and the aggregated information includes information collected during a whole year. Each of those data sets have a size that refers to the unique combinations of vehicle-dates. Their sizes are the following ones:

- **Data set 1 (D1)**: 115860 data points. 12 vehicle groups.
- **Data set 2 (D2)**: 28665 data points. 273 vehicle groups.
- **Data set 3 (D3)**: 823 data points. 30 vehicle groups.

We will use those data sets for the training phase (using a part of them for training and another one for testing, as described in the previous Chapter), and use a subset of them for the explaining phase. Particularly, we will use subsets of complete months to generate those explanations. Due to their size, since D3 is very small, we will use the whole data set for explanations, instead of only one month.

Regarding the business variables described in the previous Chapter, the values chosen for the evaluations conducted are the following ones:

- $th\_kms = 30$
- $low\_th\_time = 0.55$
- $high\_th\_time = 0.55$
- $th\_degree\_1 = 0.1$
- $th\_degree\_2 = 0.21$
- $th\_degree\_3 = 0.4$
- $th\_ebm\_var = 100$
- $min\_day\_km = 6.7$
- $min\_days\_anomalies = 3$
- $min\_dev\_total\_avg\_fuel = 1$

Also, we will consider a 90/10 train/test split for testing.

4.2. Model configuration

The hyperparameters used for every model match the default ones provided by the software libraries used (only modifying the parameters related to the monotonic constraints) since we did not find any significant improvements after using a grid search over the training data. Regarding "EBM variation", both the general EBM and the EBM for error prediction within the different subgroups use the same hyperparameter configuration than the ones described above.

4.3. Model evaluation

Metrics over K-Fold Cross-Validation

First, we address the comparison between the different models using the "model metrics" described in the previous Chapter, in order to see if there are significant differences between the predictive power of the ML models analysed in this paper. For doing it, we perform a k-fold Cross-Validation (CV) over the train data set using 30 splits. For every one of those splits, we train a model on a subset of the training data and evaluate it over the validation data selected by k-fold CV. This is done for each of those 30 splits, and for each of the three different data sets used.

This yields a vector of 30 components for each data set-metric-ML model that will be used for comparing against the other combinations of ML models belonging to the same data set-metric. The comparison is carried out by using Wilcoxon signed-rank test [23] in order to see if the metrics of two of the ML models have the same distribution. Wilcoxon signed-rank test is chosen for this hypothesis testing since it's a non-parametric test that can be applied over paired or potentially related data. This last consideration is important since the metrics obtained after the k-fold CV may be related to some degree, because the same data sets are used for different models, and the metrics from a k-fold of a particular data set-metric-ML model may be using similar training data compared to another k-fold.

Thus, we check the p-value resulting from the hypothesis test in order to see if H0 is rejected (H0 = distributions are equal), using 0.05 as the threshold value for rejecting H0.

The results of the hypothesis tests for each of the data sets are included in Table 4. That table contain the pair of models compared ("model_1" and "model_2"), along with the metric considered and the median value for the 30 k-fold splits used at every data set (for example, D_3_m_2 is the median value for model_2 with the metric considered at data set 3). It also includes the pvalue from Wilcoxon signed-rank test at each data set (P1 is the pvalue at D1, and so on).

First, we analyse the comparisons regarding the baseline model, ElasticNet (labeled as "linear_model"). Out of all the metrics and data sets, in 93% of the cases there are significant differences between this model and the other ones, while this model has a worst median value (higher error metrics, lower r2 and explained variance). This highlights how the predictive power of ElasticNet for our use case is almost always significantly worse than using any of the other models considered.

The next analysis that we consider is regarding XGBoost results versus LightGBM. The expected result is that their metrics should be similar, as it is reported in different benchmarks within the literature [24], [25]. Out of the 18 combinations of metrics-data sets, 13 of them (72%) have significantly different metrics distributions according to the hypothesis test. Regarding D1 and D2, in all the metrics the results from XGBoost outperform those from LightGBM (lower error metrics, higher r2 and explained variance) considering those cases with p-values$<$0.05. However, for the cases with p-values < 0.05 in D3, LightGBM offer better results. The gap between the metrics, however, is clearly smaller than the one comparing ElasticNet (p.e. the median value r2 for D3 is 0.65 for XGBoost, 0.675 for LightGBM, while being 0.28 for ElasticNet).

Regarding the comparisons between LightGBM and EBM, we see that 11 out of the 18 data sets-metrics combinations (61%) have significantly different metric distributions. In all those cases, EBM are worse than those from LightGBM (higher error metrics, lower r2 and explained variance), though with a much smaller difference than that compared to ElasticNet (p.e. for instance, the median r2 value for D1 is 0.67 for EBM and 0.69 for LightGBM).

Something similar takes place when comparing XGBoost versus EBM. There are no significant differences regarding D3, but the differences regarding D1 and D2 are bigger since XGBoost obtained better metrics than LightGBM for those data sets. The percentage of data sets-metrics that have significantly different distributions comparing EBM to XGBoost is also 11 out of 18 (61%).

These analyses show how EBM matches XGBoost for model performance over D3. However, there are significant differences between those two models in all the metrics of data sets D2 and D1, even though the difference between them is much lower than the one compared to ElasticNet (EBM significantly outperforms ElasticNet in 17 out of 18 data set-metric combinations). Also, it matches LightGBM metrics regarding the ”median_absolute_error” in all data sets, as well as the ”max_error” in D3 and D3, and the r2 score and explained variance at D3.

The next step is comparing the results from ”EBM_variation”. Comparing against the base EBM, ”EBM_variation” outperforms it in 7 out of the 18 data set-model combinations. The cases where it outperforms EBM all belong to D1 and D2, the data sets with more registers. This happens due to the fact that D3 have many vehicle_groups where the number of registers do not meet the threshold th_ebm_var, hence the model used is the base EBM and that lead to the exactly the same metrics. So, the proper comparison is regarding D1 and D2 only. Thus, it outperforms the base model in 7 out of the 12 data set-model combinations. This includes all the metrics except for ”max_error” in both data sets, and ”mean_squared_error” in D2.

Comparing ”EBM_variation” to LightGBM, we see how the 11 different combinations from ”EBM” change significantly. In these comparison, there are only 3 (16.7%) metric distributions (”median_absolute_error” for D1 and D2, and ”mean_absolute_error” for D1), where ”EBM_variation” actually outperforms LightGBM (lower error metric values).

Regarding XGBoost, there are only 2 significantly different metric distributions, belonging to the ”median_absolute_error” at both D1 and D2. In those cases, ”EBM_variation” also outperforms XGBoost.

With all these analyses, we first see regarding EBM, that even though its metrics are significantly lower than those form XGBoost and LightGBM, it only takes place for some combinations of data sets-metrics. And even then, the differences are significantly lower than those against the baseline model ElasticNet. Second, we see how using ”EBM_variation” significantly improves the results, offering a model that generally matches in performance both XGBoost and LightGBM, even outperforming them for some data sets and metrics combinations.

To visually illustrate these comparisons, we include with Figures 11, 12 and 13 the model metrics results for explained variance, max error and mean squared error respectively. We only show these three metrics since r2 is similar to explained variance, and the metric distributions of median and mean absolute errors are similar to the ones obtained with mean squared error.

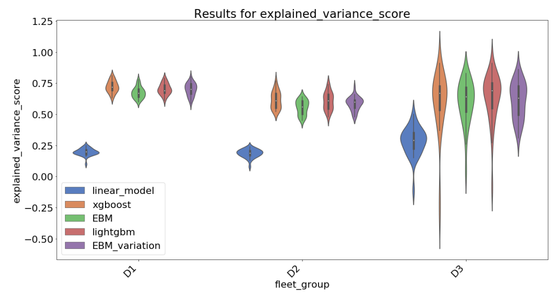

Figure 11: Model metric results for explained variance. X-axis include the metric value, and Y-axis the three different data sets used.

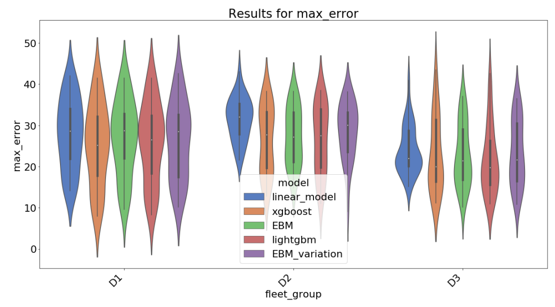

Figure 12: Model metric results for max error. X-axis include the metric value, and Y-axis the three different data sets used.

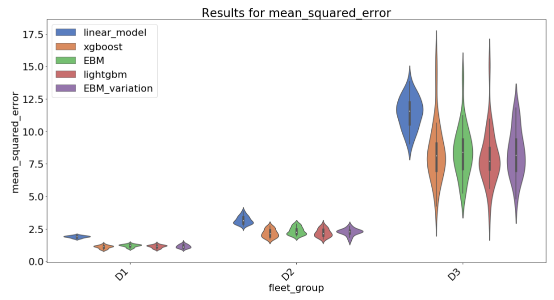

Figure 13: Model metric results for mean squared error. X-axis include the metric value, and Y-axis the three different data sets used.

Metrics over test set

After the model comparison checked above, we analyse if the metrics regarding the test set used for each of the three data sets are good enough. Here, as mentioned in the previous Chapter, we use adj-R2 and MAPE, since they both yield a result in terms of percentage that is easily comprehended.

Regarding adj-R2, even if it's clear that it indicates the proportion of the variance in the target feature that can be predicted using the input features, it is not trivial to define value thresholds to indicate if the model is good or not. It heavily depends on both the context and the units of the target feature [26, 27]. However, there are some guidelines that may be considered. As a reference, we use the proposal of [28], that mentions the following levels:

- 0.67: Substantial
- 0.33: Moderate
- 0.19: Weak

MAPE, though it's a metric commonly used for forecasting models, it is also useful for regression one [29]. Again, though it is also not direct to define thresholds for MAPE, we use as reference the ones detailed in [30], even though they are originally proposed for forecasting models.

- < 10: Highly accurate forecasting
- $10 - 20$: Good forecasting
- $20 - 50$: Reasonable forecasting
- > 50: Inaccurate forecasting

The metrics over the test set are included in Table 1. We only use D1 and D2 for this evaluation since those data sets are the only ones that have a size for the test set meaningful enough (D3 is too small).

Regarding the ElasticNet model ("linear_model"), its adjusted r2 belongs to the "Weak" level for both data sets. Its MAPE metric is in the frontier between "Good forecasting" and "Reasonable forecasting", with D2 belonging to that first level mentioned and D1 to the second one. Thus, the linear model have reasonable predictions, but it lacks generalization power since its adjusted r2 is very low.

fleet_group	model	adjst_r2	mape
D1	EBM	0.6785	0.1033
D1	EBM_variation	0.7024	0.0981
D1	lightgbm	0.6558	0.1129
D1	linear_model	0.1021	0.2038
D1	xgboost	0.7357	0.0930
D2	EBM	0.5215	0.1280
D2	EBM_variation	0.5851	0.1201
D2	lightgbm	0.6221	0.0987
D2	linear_model	0.1025	0.1955
D2	xgboost	0.5685	0.1119

Table 1: Model metrics results with respect to test set

XGBoost has a "Substantial" adjusted r2 level for D1 and a "Moderate" one for D2. Its MAPE is in the frontier of "Highly accurate" and "Good", being the first one for D1 and the second one for D2. LigthGBM has an adjusted r2 within the "Moderate" level for both data sets, but with values close to the "Substantial" level threshold (in both cases adjusted r2 is above 0.6). Its MAPE is also oscillating between "Highly accurate" for D2 and "Good" for D1. We see how LightGBM seems to fit better to D2 than XGBoost, and XGBoost seems to fit better for D1.

Next, we analyse the metrics regarding EBM. For D1, EBM has a "Substantial" adjusted r2 (better than LightGBM) and "Good" MAPE (close to being "Highly accurate" and better than LightGBM). For D2, it has a "Moderate" adjusted r2 (closer to the value of XGBoost), and "Good" MAPE (worse than that of XGBoost and LightGBM). Thus, EBM is obtaining model metrics over the test set belonging to the same levels of either XGBoost or LightGBM, with some differences depending on the data set considered.

Finally, we analyse the metrics for "EBM_variation". For both metrics and in both data sets, "EBM_variation" is improving the results obtained with EBM (higher adjusted r2, lower MAPE), even surpassing XGBoost in its adjusted r2 for D2.

4.4. XAI evaluation

In this Section, we present the results for the XAI metrics described in the previous Chapter. For these evaluations, we use D1 and D2 as data sets only, without using D3 since its size is too small for obtaining meaningful XAI metrics. We have trained a different model over each of those data sets, using as training data size the same size as from the previous analyses. Then, we use these models for computing the XAI metrics. These metrics will be obtained using two types of data. First, using as input different months from the historical data (thus, they may contain either training data, test data or both), having then 12 data points for each model-xai technique-data set and metric. Second, using as input data the test data not used for training the models (which corresponds to the 10% of the data points from the input data set).

The XAI metrics are computed considering only the outlier data points, since we are interesed in measuring the understandability of explanations for those data points (because these are the explanations that are going to be received by the different users). So the 10% of test data has the following sizes, depending on the data set:

- $D1$: 244 data points
- $D2$: 98 data points

Thus, the XAI metrics will either use D1 and D2 with 12 data points each, or D1 with 244 and D2 with 98 data points. This again is enough for using Wilcoxon signed-rank test [23] to see if there are significant differences between the model-xai technique combinations for each of those metrics using D1 or D2. The comparison will only be focused in EBM or "EBM_variation" against the remaining ML models-xai techniques combinations, beginning with the analysis of EBM and the seeing if there are any improvements by using "EBM_variation".

Table 3 contains the summary of the metrics used for the different analyses, together with the type of data used for the evaluations (the 12 monthly periods, the test set, or both).

4.4.1. Representativeness metrics It includes two subgroups: *general metrics* and *monotonicity metrics*.

General metrics
The metric results belonging to *general metrics* appear at Table 5. Again, columns with "D" indicate the median value for that combination (for instance, D_1_m_2 is the median value for model_2 with the metric considered at data set 1). P indicates the p-value for that data set.

Comparing EBM to ElasticNet (linear_model), there is a significant improvement in every metric considered for both data sets, since ElasticNet is normally not able to obtain explanations after applying the rules. Due to this, the remaining analyses will not include ElasticNet.

Compared to XGBoost as ML model and Tree-SHAP as XAI technique, the metrics results do not show any significant variations at D2, except for "mean_variables_used_per_group", where EBM has better results. It is at D1 where we can see significant differences in the metrics. EBM is able to significantly explain more data points, while also having significantly better metrics for all of the remaining metrics, except for "n_variables_used", where "SHAP_tree_xgboost" has better results. It shows how "SHAP_tree_xgboost" is able to retain more features after applying the filtering, but EBM is using more features on average for explaining every day within each month period.

Considering EBM against "SHAP_tree_lightgbm", the metrics show significant differences for all the metrics and all data sets, except for "n_variables_used" at D1 and "per_datapoints_explained" at D2, where EBM has similar results to LightGBM with Tree-SHAP. At the metrics where there is significant difference, EBM surpasses "SHAP_tree_lightgbm".

Regarding ML models using LIME, the metrics show even bigger differences, with better results for EBM than when considering tree-SHAP. The only exception is "n_variables_used", where EBM is significantly below LIME with either of the ML models.

Finally, considering the comparison of EBM versus "EBM_variation" we see that besides the "per_datapoints_explained" at both data sets and "n_variables_used" at D2, where they have similar results, the metrics have significant differences. It is worth mentioning that both models have a perfect score in "per_datapoints_explained", meaning that they are able to explain the 100% of data points. For the remaining metrics with significant differences, we see that "EBM_variation" improves EBM at D1 in the "n_variables_used", closing the gap for that metric to the rest of the models (and, in fact, not having significant differences with the models using LIME). However, is below EBM for "mean_variables_used" and "mean_variables_per_group" for D1, and "mean_variables_per_group" only for D2.

Monotonicity metrics
In this set of metrics, we only consider "per_monotonic_datapoints", which analyse the percentage of the remaining data points after applying the monotonicity filter. The results appear at Table 7. Since the median values do not clearly shows which group is above the other in some cases, we also include the mean values.

First, we see that at neither D1 nor D2, EBM has significant differences compared to "EBM_variation". In fact, at D2 they have an almost perfect score, so they do not discard many data points after applying the monotonicity filter. EBM also does not show any significant differences compared to ML models with Tree-SHAP in any of the data sets. This also happens with "EBM_variation". The only significant differences happens with models with LIME. There are significant improvements in the metric with either EBM or "EBM_variation". Finally, the comparison against ElasticNet is interesting since its score is always perfect because the model is strictly monotonic. At D1, EBM has significant differences to ElasticNet. At D2, there are also significant differences, but very close to the 0.05 threshold.

4.4.2. Fidelity metrics Fidelity metrics include *fidelity-target* and *fidelity-model*.

Fidelity-target metrics
As mentioned in the previous Chapter, *fidelity-target metrics* apply several metrics over a predicted value that was computed considering only the feature relevance values remaining after applying all the business rules and monotonicity filters. Results appear at Table 8. Also, as mentioned previously, in this comparison we only consider EBM and not "EBM_variation" since the predictions are obtained using different periods of data that could have already been used for training the model, and regarding "EBM_variation" they may have even been used for optimising the error. With this analysis we only want to compare the proposals. The metrics themselves and alone are not informative.

We see that EBM have significantly better MAPE value at D1 compared to any of the blackbox configurations. At D2, EBM is significantly better than models with LIME, while being significantly worse than models with Tree-SHAP. Regarding "max_error", at D1 there are no significant differences, expect for LIME with LightGBM, where EBM is significantly worse. At D2, EBM has similar results for that metric compared to models with LIME, but has worse results compared to models with Tree-SHAP. Finally, regarding "mean_squared_error", EBM is significantly better at D1 than any oth the other combinations, except for XGBoost with Tree-SHAP, where the results are similar. At D2, EBM is also significantly better than the remaining combinations, except for Tree-SHAP with LightGBM, where EBM is significantly worse.

Fidelity-model metrics
Following with fidelity metrics, Table 9 show the results of the "fidelity-model" metric. As mentioned in the previous Chapter, the lower the value, the better, since it indicates less difference between model prediction before and after applying the rules. These metrics uses the test set.

First, we see that EBM do not have significant differences when compared to "EBM_variation" at neither D1 nor D2. For the remaining comparisons, with the exception of EBM and "EBM_variation" with Tree-SHAP and LightGBM at D1, we see significant differences. Generally speaking, EBM and "EBM_variation" have worse metrics (more metric value) than the combinations with Tree-SHAP, while outperforming the ones with LIME.

4.4.3. Stability metrics Finally, we include the results of the stability metrics at Table 10. That table contains the two metrics described in the previous Chapter: *stability_error* and *xai_stability_error*. For both of them, the lower the value, the better, since it indicates less difference between similar data point predictions (before applying the rules for *stability_error*, and after for *xai_stability_error*). These metrics use the test set.

Stability_error metrics
For "stability_error", we see that EBM has similar results to Tree-SHAP with XGBoost at D1, while having significantly better results at D2. Regarding LightGBM, it significantly outperforms EBM at both data sets. Considering ML models with LIME, EBM significantly has better metrics at both data sets, except for LIME with LightGBM at D2, which improves the results of EBM. EBM compared to "EBM_variation" have similar results at both data sets.

XAI_stability_error metrics
For "xai_stability_error", we see that EBM has similar results to Tree-SHAP with XGBoost at D1, while having significantly better results at D2. Regarding LightGBM, it significantly outperforms EBM at D1 while having similar results at D2. Considering ML models with LIME, EBM significantly has better metrics at both data sets, except for LIME with XGBoost at DE, which improves the results of EBM. EBM compared to "EBM_variation" have similar results at both data sets.

4.4.4. Contrastiveness metrics Contrastiveness metrics include "per_rec_below", that calculates the percentage of data points (original ones) that receive recommendations over the actionable features from Table 2 that turns the anomalous average fuel consumption into an inlier if the feature values for that vehicle-route type change for the median inlier values of the vehicles of the same group over the same route type.

This analysis is only performed over EBM and "EBM_variation", since they are the only models considered for the recommendation Algorithm 4, as mentioned at Section 3.6. ElasticNet is not considered as indicated earlier.

T

Table 6 shows the results over both the test set and the monthly periods. We see that there are only significant differences at D2 using the monthly periods. For the remaining comparisons the results are similar.

4.5. Software Used

The main libraries used for the work done in this paper are the following:

- XGBoost [31]
- LightGBM [32]
- ElasticNet [33]
- EBM, LIME [34]
- Tree-SHAP [35]

4.6. Limitations of our approach

One of the limitations in our proposal is that we check monotonicity within the period of data that is going to be explained. This, however, has two downsides. First, it would not be suitable for explaining only one data point. Second, even if the periods of time used for the evaluation are big (one whole month of data), the results may differ if the monotonicity is analysed by combining both the period of data to explain and the whole historical data used for training.

Also, the domain knowledge used needs to be expressed through business rules, but this may not be suitable for all use cases. This may be improved by using a more flexible framework to gather that apriori knowledge (p.e. using ontologies).

Together with that, we only work with the individual feature relevance of each variable for building the recommendations, not considering possible pairwise terms if they exist.

Finally, our approach deals with explaining average fuel consumptions that are outliers due to several factors. This does not account for all possible features that may affect fuel usage, it uses only a subset of them. Also, we are not dealing with every possible cause of anomalous fuel usage. There are other causes, like fuel fraud, that are not considered within the scope of our proposal, mainly because they do not take place within the data sets used.

5. Conclusion

We have proposed a complete process for unsupervised anomaly detection in the average fuel consumption of the vehicles of a fleet, where the anomalies are explained using XAI and based on the feature relevance of several variables that may impact fuel usage. The explanations take into account domain knowledge expressed through business rules, and expressed through counterfactual recommendations that are adjusted depending on two different user profiles that will use them. The process is evaluated using real IoT industry data belonging to Telefónica.

Using that real data, we have also evaluated different possibilities for building a surrogate model model that infer the relationships between the input data and the predicted average fuel consumption, in order to be able to explain later how it can be reduced to be below the anomaly limit inferred unsupervised. For those surrogate models, we have considered both blackbox models together with posthoc XAI techniques for feature relevance, and whitebox models like EBM, that directly have algorithm transparency in terms of feature relevance. We include in this evaluation a novel variation over EBM.

In order to compare the different surrogate model alternatives, we have performed evaluations in terms of performance metrics (how well the model predicts the target feature), and XAI metrics, that compare the explanations generated in terms representativeness, fidelity, stability and contrastiveness.

The evaluations showed that both EBM, and our variation of EBM, either outperform the blackbox models counterparts regarding those performance metrics, or are below but very close to them. They also provide satisfactory results analysing their metrics in absolute terms. For XAI metrics the conclusions are similar. Using EBM or our EBM variation yields either similar or even better results than using a blackbox model together with a posthoc XAI technique for local feature relevance.

5.1. Future Work

We see two main research lines where our current research can be continued. The first one is regarding the unsupervised algorithm for anomaly detection. Within our proposal, we have used a boxplot applied over the average fuel consumption of the vehicles of a same group since it directly provides a limit that helps seeing the threshold value that sets apart anomalous fuel consumption and non anomalous one. It also provides a visual limit that provides an additional insight for the users since they can see the average fuel split between inliers and outliers. However, there are other unsupervised algorithms that can be used if they are able to provide that threshold limit.

The second line is regarding the XAI metric usage. The literature propose other aspects that can be measured in terms of human-friendly explanations, and is important to both include those aspects, as well as assessing with different real users that the metrics do indeed measure that aspects.

Acknowledgements

This research was done following the registered patent [36] for LUCA Fleet at Telefónica. We thank Pedro Antonio Alonso Baigorri, Federico Pérez Rosado, Raquel Crespo Crisenti and Daniel García Fernández for their collaboration.

References

[1] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115.

[2] Richard Benjamins, Alberto Barbado, and Daniel Sierra. *Responsible AI by Design*. 2019. arXiv: 1909.12838 [cs.CY]. URL: https://arxiv.org/abs/1909.12838.

[3] Vanessa Gironda Aquize, Eduardo Emery, and Fernando Buarque de Lima Neto. "Self-organizing maps for anomaly detection in fuel consumption. Case study: Illegal fuel storage in Bolivia". In: *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE. 2017, pp. 1–6.

[4] Mingming Zhang et al. "SafeDrive: online driving anomaly detection from large-scale vehicle data". In: *IEEE Transactions on Industrial Informatics* 13.4 (2017), pp. 2087–2096.

[5] Christoph Molnar. *Interpretable machine learning*. Lulu.com, 2019. URL: https://christophm.github.io/interpretable-ml-book/.

[6] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "" Why should I trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[7] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems*. 2017, pp. 4765–4774.

[8] Lloyd S Shapley. "A value for n-person games". In: *Contributions to the Theory of Games* 2.28 (1953), pp. 307–317.

[9] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. "Consistent individualized feature attribution for tree ensembles". In: *arXiv preprint arXiv:1802.03888* (2018).

[10] Shubham Rathi. "Generating counterfactual and contrastive explanations using SHAP". In: *arXiv preprint arXiv:1906.09293* (2019).

[11] Harsha Nori et al. "InterpretML: A Unified Framework for Machine Learning Interpretability". In: *arXiv preprint arXiv:1909.09223* (2019).

[12] Trevor Hastie and Robert Tibshirani. "Generalized additive models: some applications". In: *Journal of the American Statistical Association* 82.398 (1987), pp. 371–386.

[13] Yin Lou et al. "Accurate intelligible models with pairwise interactions". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2013, pp. 623–631.

[14] Roberto Confalonieri et al. "TREPAN Reloaded: A Knowledge-driven Approach to Explaining Blackbox Models". In: ().

[15] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. “Machine Learning Interpretability: A Survey on Methods and Metrics”. In: *Electronics* 8.8 (2019), p. 832.

[16] Robert R Hoffman et al. “Metrics for explainable AI: Challenges and prospects”. In: *arXiv preprint arXiv:1812.04608* (2018).

[17] David Alvarez Melis and Tommi Jaakkola. “Towards robust interpretability with self-explaining neural networks”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 7775–7784.

[18] Alberto Barbado, Óscar Corcho, and Richard Benjamins. “Rule Extraction in Unsupervised Anomaly Detection for Model Explainability: Application to OneClass SVM”. In: *arXiv preprint arXiv:1911.09315* (2019).

[19] Hui Zou and Trevor Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the royal statistical society: series B (statistical methodology)* 67.2 (2005), pp. 301–320.

[20] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[21] Guolin Ke et al. “Lightgbm: A highly efficient gradient boosting decision tree”. In: *Advances in neural information processing systems*. 2017, pp. 3146–3154.

[22] Salwa Waeto, Khanchit Chuarkham, and Arthit Intarasit. “Forecasting time series movement direction with hybrid methodology”. In: *Journal of Probability and Statistics* 2017 (2017).

[23] Frank Wilcoxon. “Individual comparisons by ranking methods”. In: *Breakthroughs in statistics*. Springer, 1992, pp. 196–202.

[24] Martin Nemeth, Dmitrii Borkin, and German Michalconok. “The Comparison of Machine-Learning Methods XGBoost and LightGBM to Predict Energy Development”. In: *Proceedings of the Computational Methods in Systems and Software*. Springer. 2019, pp. 208–215.

[25] Andreea Anghel et al. “Benchmarking and Optimization of Gradient Boosting Decision Tree Algorithms”. In: *arXiv preprint arXiv:1809.04559* (2018).

[26] Joseph F Hair Jr et al. *A primer on partial least squares structural equation modeling (PLS-SEM)*. Sage publications, 2016, pp. 209–210.

[27] Joseph F Hair, Christian M Ringle, and Marko Sarstedt. “Partial least squares structural equation modeling: Rigorous applications, better results and higher acceptance”. In: *Long range planning* 46.1-2 (2013), pp. 1–12.

[28] Wynne W Chin and Peter R Newsted. “Structural equation modeling analysis with small samples using partial least squares”. In: *Statistical strategies for small sample research* 1.1 (1999), pp. 307–341.

[29] Arnaud De Myttenaere et al. “Mean absolute percentage error for regression models”. In: *Neurocomputing* 192 (2016), pp. 38–48.

[30] Colin David Lewis. *Industrial and business forecasting methods: A practical guide to exponential smoothing and curve fitting*. Butterworth-Heinemann, 1982.

[31] *XGBoost: eXtreme Gradient Boosting*. Version 1.2.0. 2019. URL: https://github.com/dmlc/xgboost.

[32] Microsoft. *LightGBM*. Version 3.0.0. 2020. URL: https://github.com/microsoft/LightGBM.

[33] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[34] Microsoft. *InterpretML*. Version 0.2.1. 2020. URL: https://github.com/interpretml/interpret.

[35] Scott M. Lundberg et al. “From local explanations to global understanding with explainable AI for trees”. In: *Nature Machine Intelligence* 2.1 (2020), pp. 2522–5839.

[36] Alberto Barbado et al. *Metodo y Programas de Ordenador para Gestion de Flotas de Vehiculos*. 2020.

[37] Shane T Mueller et al. “Explanation in Human-AI Systems: A Literature Meta-Review, Synopsis of Key Ideas and Publications, and Bibliography for Explainable AI”. In: *arXiv preprint arXiv:1902.01876* (2019).

[38] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation forest”. In: *2008 Eighth IEEE International Conference on Data Mining*. IEEE. 2008, pp. 413–422.

[39] Saket Sathe and Charu Aggarwal. “LODES: Local density meets spectral outlier detection”. In: *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM. 2016, pp. 171–179.

[40] Markus M Breunig et al. “LOF: identifying density-based local outliers”. In: *ACM sigmod record*. Vol. 29. 2. ACM. 2000, pp. 93–104.

[41] Vijay Arya et al. “One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques”. In: *arXiv preprint arXiv:1909.03012* (2019).

[42] Rich Caruana et al. “Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2015, pp. 1721–1730.

[43] Harmanpreet Kaur et al. “Interpreting Interpretability: Understanding Data Scientists’ Use of Interpretability Tools for Machine Learning”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–14.

[44] Chun-Hao Chang et al. “How Interpretable and Trustworthy are GAMs?” In: *arXiv preprint arXiv:2006.06466* (2020).

[45] Jörg Henseler, Christian M Ringle, and Rudolf R Sinkovics. “The use of partial least squares path modeling in international marketing”. In: *New challenges to international marketing*. Emerald Group Publishing Limited, 2009.

6. Annex

6.1. Features involved

- **count_harsh_brakes**: Total harsh brake events.
- **count_harsh_turns**: Total harsh turn events.
- **count_jackrabbit**: Total jackrabbit events.
- **count_neutral**: Total events of gear position in neutral.
- **count_reverse**: Total events of gear position in reverse.
- **engine_oil_variation**: Difference between maximum and minimum of the remaining life of the engine's oil (percentage).
- **fuel_exhaust_fluid_variation**: Difference between maximum and minimum of DEF (Diesel Exhaust Fluid).
- **fuel_filter_life_variation**: Difference between maximum and minimum of engine's fuel filter.
- **hours_speed_control**: Hours driving with speed control set on.
- **max_engine_cool_temp**: Maximum temperature reached by the coolant.
- **max_engine_oil_temp**: Maximum temperature reached by the engine's oil.
- **mean_braking_acc**: Mean value for braking acceleration.
- **mean_forward_acc**: Mean value for front acceleration.
- **mean_exterior_temp**: Mean value of the exterior temperature.
- **mean_speed_city**: Mean value of the speed within city.
- **mean_speed_hwy**: Mean value of the speed within highways.
- **mean_tire_pressure_fl**: Mean value of the wheel's pressure (front-left).
- **mean_tire_pressure_rl**: Mean value of the wheel's pressure (real-left).
- **mean_tire_pressure_fr**: Mean value of the wheel's pressure (front-right).
- **mean_tire_pressure_rr**: Mean value of the wheel's pressure (rear-right).
- **per_fuel_idle**: Percentage of total fuel consumption spent for idling.
- **per_time_city**: Percentage of time spent driving within city.
- **rpm_high**: Events with engine's speed (RPM) equal or above 1900.
- **rpm_red**: Events with engine's speed (RPM) above 3500 and vehicle speed below 40 Km/h.
- **rpm_orange**: Events with engine's speed (RPM) above 3500 and vehicle speed between 40 and 80 Km/h (included).
- **rpm_yellow**: Events with engine's speed (RPM) above 3500 and vehicle speed above 80 Km/h.
- **speed_over_120**: Time with driving speed above 120 Km/h.
- **total_odometer**: Maximum value of the odometer.
- **trip_kms**: Distance driven.
- **ignition_events**: Events of engine's ignition.
- **with_passenger**: Whether there are at least one additional passenger inside the vehicle in that day (1) or not (0).
- **lights_left_on**: Whether the lights of the vehicle where left on at least once that day (1) or not (0).
- **vehicle_id**: Unique id vehicle's number.
- **vehicle_group**: Vehicle group for that vehicle_id.
- **date_tx**: Date for each register.
- **route_type**: Primary route type for every vehicle-day (0:City, 1:Combined, 2:Highway)
- **avg_fuel_consumption**: Target column. Vehicle's average fuel consumption per 100 Km in that day.

6.2. Acronyms

- **ML**: Machine Learning
- **IoT**: Internet Of Things
- **AI**: Artificial Intelligence
- **RAI**: Responsible Artificial Intelligence
- **XAI**: Explainable Artificial Intelligence
- **FAR**: Fleet Analytical Record
- **SOTA**: State Of The Art
- **VIN**: Vehicle Identification Number
- **GBM**: Gradient Boosting Machines
- **EBM**: Explainable Boosting Machines
- **GAM**: Generalized Additive Model
- **CV**: Cross-Validation
- **EV**: Explained Variance
- **ME**: Mean Absolute Percentage Error
- **RMSE**: Mean Absolute Percentage Error
- **MAE**: Mean Absolute Percentage Error
- **MAPE**: Mean Absolute Percentage Error
- **adj-R2**: Adjusted R2

6.3. Figures and Tables

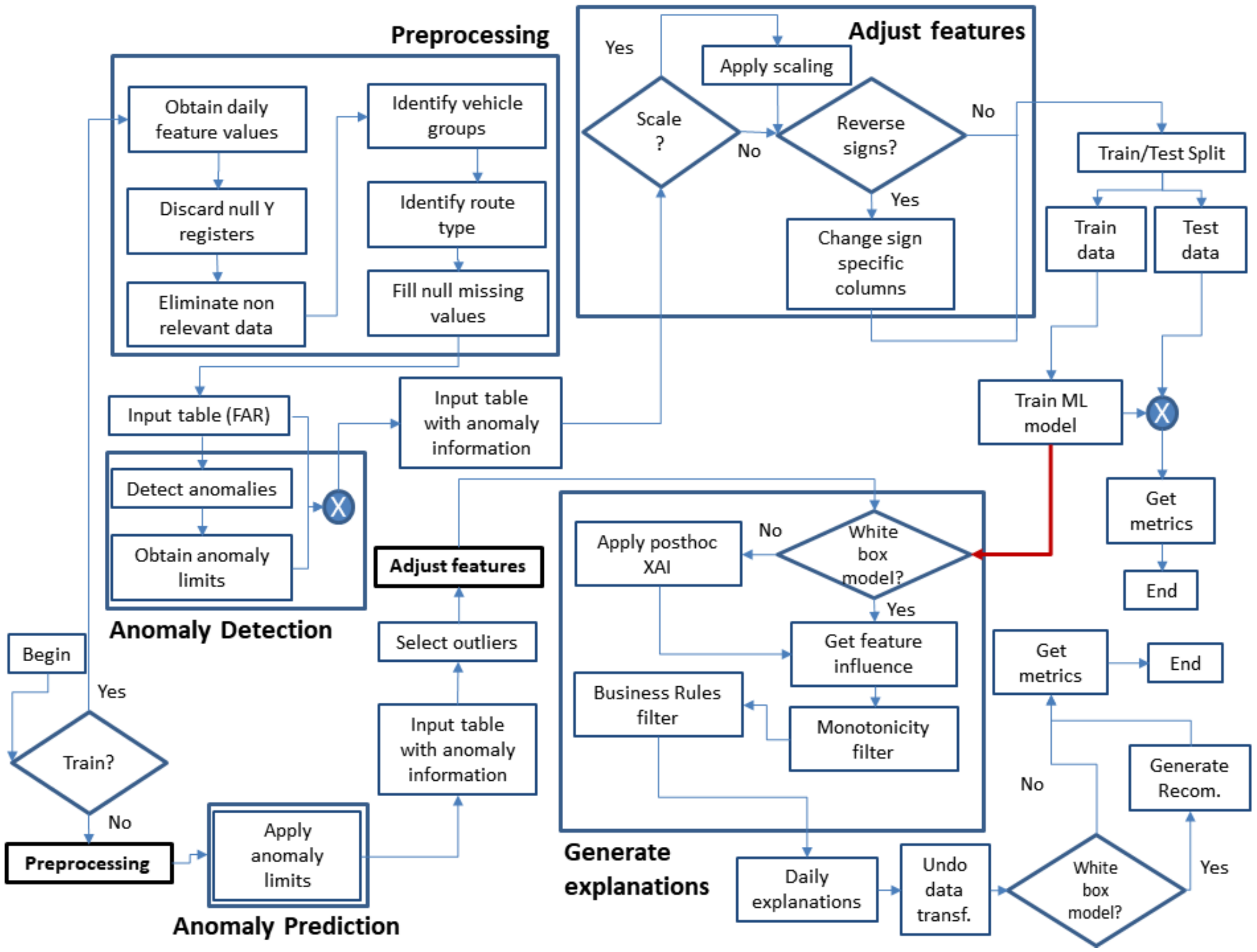

Figure 14: Flowchart followed by the process.

Variable Name	Units	Type	Category	Period only?	Actionable?
count_harsh_brakes	Events	+	Driving Behaviour	No	Yes
count_harsh_turns	Events	+	Driving Behaviour	No	Yes
count_jackrabbit	Events	+	Driving Behaviour	No	Yes
count_neutral	Events	+	Driving Behaviour	No	Yes
count_reverse	Events	+	Driving Behaviour	No	Yes
engine_oil_variation	%	+	Car Parameters	No	Yes
fuel_exhaust_fluid_variation	%	+	Car Parameters	No	Yes
fuel_filter_life_variation	%	+	Car Parameters	No	Yes
hours_speed_control	Hours	+	Driving Behaviour	No	Yes
idle_time	Hours	+	Driving Behaviour	No	Yes
max_engine_cool_temp	ºC	+	Car Parameters	No	Yes
max_engine_oil_temp	ºC	+	Car Parameters	No	Yes
mean_braking_acc	m/s^2	+	Driving Behaviour	No	Yes
mean_exterior_temp	%	-	Environment Variables	Yes	Yes
mean_forward_acc	m/s^2	+	Driving Behaviour	No	Yes
mean_speed_city	m/s	+	Driving Behaviour	No	Yes
mean_speed_hwy	m/s	+	Driving Behaviour	No	Yes
mean_tire_pressure_fl	Pa	-	Car Parameters	No	Yes
mean_tire_pressure_fr	Pa	-	Car Parameters	No	Yes
mean_tire_pressure_rl	Pa	-	Car Parameters	No	Yes
mean_tire_pressure_rr	Pa	-	Car Parameters	No	Yes
per_fuel_idle	%	+	Driving Behaviour	No	Yes
per_time_city	%	+	Driving Behaviour	No	Yes
rpm_high	Events	+	Driving Behaviour	No	Yes
speed_over_120	Hours	+	Driving Behaviour	No	Yes
total_odometer	Km	+	Car Parameters	Yes	No
trip_kms	Km	-	General	No	Yes
with_passenger	Binary	+	Driving Behaviour	No	Yes
rpm_red	Events	+	Driving Behaviour	No	Yes
rpm_orange	Events	+	Driving Behaviour	No	Yes
rpm_yellow	Events	+	Driving Behaviour	No	Yes
vehicle_id			General		
vehicle_group			General		
road_type			General		
date_tx			General		
trip_fuel_used	Litres/100Km		General		

Table 2: Input table used for anomaly detection. It contains the predictor columns with their units and monotonicity sign (type) as well as with the feature category (p.e. Driving Behaviour). If empty, there is no monotonicity constraint. It also contains the categorical features (Categorical), the indexes (date_tx and vehicle_id) and the target column (avg_fuel_consumption).

Taxonomy	Subtype	Metric name	Data used
Representativeness	General metrics	n_datapoints	Monthly period
Representativeness	General metrics	per_datapoints_explained	Monthly period
Representativeness	General metrics	n_variables_used	Monthly period
Representativeness	General metrics	mean_variables_used_per_day	Monthly period
Representativeness	General metrics	mean_variables_used_per_group	Monthly period
Representativeness	Monotonicity metrics	per_monotonic_datapoints	Monthly period
Fidelity	Fidelity-target	MAPE	Monthly period
Fidelity	Fidelity-target	ME	Monthly period
Fidelity	Fidelity-target	RMSE	Monthly period
Fidelity	Fidelity-model	faithfulness_error	Test set
Stability	Model stability	stability_error	Test set
Stability	XAI stability after rules	xai_stability_error	Test set
Contrastiveness		per_rec_below	Both

Table 3: Resume of the XAI metrics analysed, linking them to their taxonomy.

model_1	model_2	metric	D1_m1	D1_m2	p1	D2_m1	D2_m2	p2	D3_m1	D3_m2	p3
EBM	xgboost	explained_variance_score	0.67	0.72	0.0	0.56	0.62	0.0	0.63	0.66	0.98
EBM	xgboost	max_error	27.8	24.84	0.0	27.22	27.0	0.21	21.12	19.96	0.99
EBM	xgboost	mean_absolute_error	0.65	0.62	0.0	1.21	1.12	0.0	6.26	6.04	0.27
EBM	xgboost	mean_squared_error	1.25	1.14	0.0	2.28	2.16	0.0	8.42	8.2	0.99
EBM	xgboost	median_absolute_error	0.42	0.41	0.0	0.67	0.63	0.0	4.66	4.5	0.09
EBM	xgboost	r2_score	0.67	0.72	0.0	0.56	0.62	0.0	0.59	0.65	0.83
EBM	lightgbm	explained_variance_score	0.67	0.69	0.0	0.56	0.6	0.0	0.63	0.68	0.05
EBM	lightgbm	max_error	27.8	26.49	0.0	27.22	26.72	0.3	21.12	19.44	0.48
EBM	lightgbm	mean_absolute_error	0.65	0.64	0.0	1.21	1.16	0.0	6.26	5.68	0.0
EBM	lightgbm	mean_squared_error	1.25	1.2	0.0	2.28	2.18	0.0	8.42	7.64	0.04
EBM	lightgbm	median_absolute_error	0.42	0.42	0.24	0.67	0.66	0.76	4.66	4.23	0.07
EBM	lightgbm	r2_score	0.67	0.69	0.0	0.56	0.6	0.0	0.59	0.68	0.08
EBM	linear_model	explained_variance_score	0.67	0.2	0.0	0.56	0.19	0.0	0.63	0.29	0.0
EBM	linear_model	max_error	27.8	28.05	0.0	27.22	32.08	0.0	21.12	21.82	0.25
EBM	linear_model	mean_absolute_error	0.65	1.26	0.0	1.21	1.75	0.0	6.26	9.72	0.0
EBM	linear_model	mean_squared_error	1.25	1.9	0.0	2.28	3.14	0.0	8.42	11.47	0.0
EBM	linear_model	median_absolute_error	0.42	0.91	0.0	0.67	1.04	0.0	4.66	9.25	0.0
EBM	linear_model	r2_score	0.67	0.2	0.0	0.56	0.19	0.0	0.59	0.28	0.0
EBM	EBM_variation	explained_variance_score	0.67	0.7	0.02	0.56	0.6	0.05	0.63	0.62	0.77
EBM	EBM_variation	max_error	27.8	27.48	0.75	27.22	29.7	0.39	21.12	21.57	0.63
EBM	EBM_variation	mean_absolute_error	0.65	0.62	0.0	1.21	1.13	0.0	6.26	5.76	0.44
EBM	EBM_variation	mean_squared_error	1.25	1.14	0.07	2.28	2.29	0.55	8.42	8.04	0.85
EBM	EBM_variation	median_absolute_error	0.42	0.4	0.0	0.67	0.6	0.0	4.66	4.46	0.54
EBM	EBM_variation	r2_score	0.67	0.7	0.03	0.56	0.6	0.05	0.59	0.62	0.83
xgboost	lightgbm	explained_variance_score	0.72	0.69	0.0	0.62	0.6	0.01	0.66	0.68	0.06
xgboost	lightgbm	max_error	24.84	26.49	0.0	27.0	26.72	0.69	19.96	19.44	0.53
xgboost	lightgbm	mean_absolute_error	0.62	0.64	0.0	1.12	1.16	0.0	6.04	5.68	0.29
xgboost	lightgbm	mean_squared_error	1.14	1.2	0.0	2.16	2.18	0.01	8.2	7.64	0.05
xgboost	lightgbm	median_absolute_error	0.41	0.42	0.0	0.63	0.66	0.0	4.5	4.23	0.64
xgboost	lightgbm	r2_score	0.72	0.69	0.0	0.62	0.6	0.01	0.65	0.68	0.04
xgboost	linear_model	explained_variance_score	0.72	0.2	0.0	0.62	0.19	0.0	0.66	0.29	0.0
xgboost	linear_model	max_error	24.84	28.05	0.0	27.0	32.08	0.0	19.96	21.82	0.3
xgboost	linear_model	mean_absolute_error	0.62	1.26	0.0	1.12	1.75	0.0	6.04	9.72	0.0
xgboost	linear_model	mean_squared_error	1.14	1.9	0.0	2.16	3.14	0.0	8.2	11.47	0.0
xgboost	linear_model	median_absolute_error	0.41	0.91	0.0	0.63	1.04	0.0	4.5	9.25	0.0
xgboost	linear_model	r2_score	0.72	0.2	0.0	0.62	0.19	0.0	0.65	0.28	0.0
xgboost	EBM_variation	explained_variance_score	0.72	0.7	0.11	0.62	0.6	0.1	0.66	0.62	0.9
xgboost	EBM_variation	max_error	24.84	27.48	0.64	27.0	29.7	0.3	19.96	21.57	0.85
xgboost	EBM_variation	mean_absolute_error	0.62	0.62	0.42	1.12	1.13	0.34	6.04	5.76	0.89
xgboost	EBM_variation	mean_squared_error	1.14	1.14	0.17	2.16	2.29	0.12	8.2	8.04	0.85
xgboost	EBM_variation	median_absolute_error	0.41	0.4	0.02	0.63	0.6	0.0	4.5	4.46	0.6
xgboost	EBM_variation	r2_score	0.72	0.7	0.1	0.62	0.6	0.1	0.65	0.62	0.96
lightgbm	linear_model	explained_variance_score	0.69	0.2	0.0	0.6	0.19	0.0	0.68	0.29	0.0
lightgbm	linear_model	max_error	26.49	28.05	0.0	26.72	32.08	0.0	19.44	21.82	0.01
lightgbm	linear_model	mean_absolute_error	0.64	1.26	0.0	1.16	1.75	0.0	5.68	9.72	0.0
lightgbm	linear_model	mean_squared_error	1.2	1.9	0.0	2.18	3.14	0.0	7.64	11.47	0.0
lightgbm	linear_model	median_absolute_error	0.42	0.91	0.0	0.66	1.04	0.0	4.23	9.25	0.0
lightgbm	linear_model	r2_score	0.69	0.2	0.0	0.6	0.19	0.0	0.68	0.28	0.0
lightgbm	EBM_variation	explained_variance_score	0.69	0.7	0.91	0.6	0.6	0.28	0.68	0.62	0.62
lightgbm	EBM_variation	max_error	26.49	27.48	0.91	26.72	29.7	0.38	19.44	21.57	0.43
lightgbm	EBM_variation	mean_absolute_error	0.64	0.62	0.0	1.16	1.13	0.53	5.68	5.76	0.6
lightgbm	EBM_variation	mean_squared_error	1.2	1.14	0.97	2.18	2.29	0.21	7.64	8.04	0.37
lightgbm	EBM_variation	median_absolute_error	0.42	0.4	0.0	0.66	0.6	0.0	4.23	4.46	0.73
lightgbm	EBM_variation	r2_score	0.69	0.7	0.84	0.6	0.6	0.29	0.68	0.62	0.55
linear_model	EBM_variation	explained_variance_score	0.2	0.7	0.0	0.19	0.6	0.0	0.29	0.62	0.0
linear_model	EBM_variation	max_error	28.05	27.48	0.43	32.08	29.7	0.08	21.82	21.57	0.61
linear_model	EBM_variation	mean_absolute_error	1.26	0.62	0.0	1.75	1.13	0.0	9.72	5.76	0.0
linear_model	EBM_variation	mean_squared_error	1.9	1.14	0.0	3.14	2.29	0.0	11.47	8.04	0.0
linear_model	EBM_variation	median_absolute_error	0.91	0.4	0.0	1.04	0.6	0.0	9.25	4.46	0.0
linear_model	EBM_variation	r2_score	0.2	0.7	0.0	0.19	0.6	0.0	0.28	0.62	0.0

Table 4: Model metrics results for model comparison. Columns with "D" indicate the median value for that combination (for instance, D3_m2 is the median value for model_2 with the metric considered at data set 3). P indicates the p-value for that data set.

model_1	model_2	metric	D1_m1	D1_m2	p1	D2_m1	D2_m2	p2
EBM	shap_tree_xgboost	per_datapoints_explained	1.0	0.98	0.011	1.0	1.0	0.317
EBM	shap_tree_xgboost	n_variables_used	27.0	31.0	0.001	19.0	19.0	1.0
EBM	shap_tree_xgboost	mean_variables_used_per_day	3.25	2.54	0.001	2.16	2.08	0.24
EBM	shap_tree_xgboost	mean_variables_used_per_group	7.74	5.62	0.001	1.38	1.31	0.001
EBM	shap_tree_lightgbm	per_datapoints_explained	1.0	0.96	0.001	1.0	1.0	0.157
EBM	shap_tree_lightgbm	n_variables_used	27.0	27.0	0.571	19.0	19.0	0.014
EBM	shap_tree_lightgbm	mean_variables_used_per_day	3.25	2.43	0.001	2.16	2.0	0.012
EBM	shap_tree_lightgbm	mean_variables_used_per_group	7.74	5.16	0.001	1.38	1.27	0.001
EBM	linear_model	per_datapoints_explained	1.0	0.0	0.001	1.0	0.03	0.001
EBM	linear_model	n_variables_used	27.0	0.0	0.001	19.0	1.0	0.001
EBM	linear_model	mean_variables_used_per_day	3.25	0.0	0.001	2.16	1.0	0.001
EBM	linear_model	mean_variables_used_per_group	7.74	0.0	0.001	1.38	1.0	0.01
EBM	lime_xgboost	per_datapoints_explained	1.0	0.75	0.001	1.0	0.52	0.001
EBM	lime_xgboost	n_variables_used	27.0	30.0	0.005	19.0	6.0	0.001
EBM	lime_xgboost	mean_variables_used_per_day	3.25	1.45	0.001	2.16	1.55	0.001
EBM	lime_xgboost	mean_variables_used_per_group	7.74	2.53	0.001	1.38	1.09	0.001
EBM	lime_lightgbm	per_datapoints_explained	1.0	0.75	0.001	1.0	0.49	0.001
EBM	lime_lightgbm	n_variables_used	27.0	30.0	0.005	19.0	5.0	0.001
EBM	lime_lightgbm	mean_variables_used_per_day	3.25	1.52	0.001	2.16	1.71	0.001
EBM	lime_lightgbm	mean_variables_used_per_group	7.74	2.66	0.001	1.38	1.11	0.001
EBM	EBM_variation	per_datapoints_explained	1.0	1.0	0.129	1.0	1.0	1.0
EBM	EBM_variation	n_variables_used	27.0	29.0	0.002	19.0	19.0	0.317
EBM	EBM_variation	mean_variables_used_per_day	3.25	3.06	0.001	2.16	2.15	0.074
EBM	EBM_variation	mean_variables_used_per_group	7.74	7.34	0.01	1.38	1.37	0.001
shap_tree_xgboost	EBM_variation	per_datapoints_explained	0.98	1.0	0.026	1.0	1.0	0.317
shap_tree_xgboost	EBM_variation	n_variables_used	31.0	29.0	0.007	19.0	19.0	0.317
shap_tree_xgboost	EBM_variation	mean_variables_used_per_day	2.54	3.06	0.001	2.08	2.15	0.32
shap_tree_xgboost	EBM_variation	mean_variables_used_per_group	5.62	7.34	0.001	1.31	1.37	0.005
shap_tree_lightgbm	EBM_variation	per_datapoints_explained	0.96	1.0	0.001	1.0	1.0	0.157
shap_tree_lightgbm	EBM_variation	n_variables_used	27.0	29.0	0.001	19.0	19.0	0.06
shap_tree_lightgbm	EBM_variation	mean_variables_used_per_day	2.43	3.06	0.001	2.0	2.15	0.005
shap_tree_lightgbm	EBM_variation	mean_variables_used_per_group	5.16	7.34	0.001	1.27	1.37	0.001
linear_model	EBM_variation	per_datapoints_explained	0.0	1.0	0.001	0.03	1.0	0.001
linear_model	EBM_variation	n_variables_used	0.0	29.0	0.001	1.0	19.0	0.001
linear_model	EBM_variation	mean_variables_used_per_day	0.0	3.06	0.001	1.0	2.15	0.001
linear_model	EBM_variation	mean_variables_used_per_group	0.0	7.34	0.001	1.0	1.37	0.01
lime_xgboost	EBM_variation	per_datapoints_explained	0.75	1.0	0.001	0.52	1.0	0.001
lime_xgboost	EBM_variation	n_variables_used	30.0	29.0	0.124	6.0	19.0	0.001
lime_xgboost	EBM_variation	mean_variables_used_per_day	1.45	3.06	0.001	1.55	2.15	0.001
lime_xgboost	EBM_variation	mean_variables_used_per_group	2.53	7.34	0.001	1.09	1.37	0.001
lime_lightgbm	EBM_variation	per_datapoints_explained	0.75	1.0	0.001	0.49	1.0	0.001
lime_lightgbm	EBM_variation	n_variables_used	30.0	29.0	0.2	5.0	19.0	0.001
lime_lightgbm	EBM_variation	mean_variables_used_per_day	1.52	3.06	0.001	1.71	2.15	0.001
lime_lightgbm	EBM_variation	mean_variables_used_per_group	2.66	7.34	0.001	1.11	1.37	0.001

Table 5: XAI general metrics results for model comparison using data sets D1 and D2. Columns with ”D” indicate the median value for that combination (for instance, D1_m2 is the median value for model_2 with the metric considered at data set 1). P indicates the p-value for that data set.

fleet_group	model_1	model_2	median_1	median_2	metric	pvalue	data
D1	EBM	EBM_variation	0.797	0.813	per_rec_below	0.069	month_periods
D2	EBM	EBM_variation	0.725	0.758	per_rec_below	0.007	month_periods
D1	EBM	EBM_variation	0.99	0.99	per_rec_below	1.0	test_set
D2	EBM	EBM_variation	0.99	0.99	per_rec_below	1.0	test_set

Table 6: XAI contrastiveness metrics results for model comparison using both data sets and only EBM and ”EBM_variation”. Obtained both over the whole historical monthly period and the test set

fleet_group	model_1	model_2	mean_1	mean_2	median_1	median_2	pvalue
D1	EBM	EBM_variation	0.838	0.825	0.85	0.83	0.102
D1	EBM	lime_lightgbm	0.838	0.521	0.85	0.52	0.001
D1	EBM	lime_xgboost	0.838	0.519	0.85	0.51	0.001
D1	EBM	linear_model	0.838	1.0	0.85	1.0	0.001
D1	EBM	SHAP_tree_lightgbm	0.838	0.821	0.85	0.82	0.091
D1	EBM	SHAP_tree_xgboost	0.838	0.832	0.85	0.83	0.61
D1	EBM_variation	lime_lightgbm	0.825	0.521	0.83	0.52	0.001
D1	EBM_variation	lime_xgboost	0.825	0.519	0.83	0.51	0.001
D1	EBM_variation	linear_model	0.825	1.0	0.83	1.0	0.001
D1	EBM_variation	SHAP_tree_lightgbm	0.825	0.821	0.83	0.82	0.472
D1	EBM_variation	SHAP_tree_xgboost	0.825	0.832	0.83	0.83	0.495
D2	EBM	EBM_variation	0.996	0.996	1.0	1.0	1.0
D2	EBM	lime_lightgbm	0.996	0.999	1.0	1.0	0.083
D2	EBM	lime_xgboost	0.996	0.999	1.0	1.0	0.18
D2	EBM	linear_model	0.996	1.0	1.0	1.0	0.046
D2	EBM	SHAP_tree_lightgbm	0.996	0.992	1.0	0.99	0.059
D2	EBM	SHAP_tree_xgboost	0.996	0.996	1.0	1.0	1.0
D2	EBM_variation	lime_lightgbm	0.996	0.999	1.0	1.0	0.083
D2	EBM_variation	lime_xgboost	0.996	0.999	1.0	1.0	0.18
D2	EBM_variation	linear_model	0.996	1.0	1.0	1.0	0.046
D2	EBM_variation	SHAP_tree_lightgbm	0.996	0.992	1.0	0.99	0.059
D2	EBM_variation	SHAP_tree_xgboost	0.996	0.996	1.0	1.0	1.0

Table 7: XAI monotonicity metrics for ”per_monotonic_datapoint” for both data sets D1 and D2

fleet_group	model_1	model_2	median_1	median_2	metric	pvalue
D1	EBM	lime_lightgbm	0.191	0.558	mape_result	0.001
D1	EBM	lime_xgboost	0.191	1.361	mape_result	0.001
D1	EBM	SHAP_tree_lightgbm	0.191	0.231	mape_result	0.007
D1	EBM	SHAP_tree_xgboost	0.191	0.221	mape_result	0.032
D1	EBM	lime_lightgbm	32.866	22.093	max_error	0.001
D1	EBM	lime_xgboost	32.866	31.927	max_error	0.898
D1	EBM	SHAP_tree_lightgbm	32.866	31.835	max_error	0.638
D1	EBM	SHAP_tree_xgboost	32.866	29.494	max_error	0.413
D1	EBM	lime_lightgbm	23.576	44.177	mean_squared_error	0.024
D1	EBM	lime_xgboost	23.576	193.153	mean_squared_error	0.003
D1	EBM	SHAP_tree_lightgbm	23.576	27.124	mean_squared_error	0.007
D1	EBM	SHAP_tree_xgboost	23.576	23.257	mean_squared_error	0.054
D2	EBM	lime_lightgbm	0.193	0.293	mape_result	0.001
D2	EBM	lime_xgboost	0.193	0.317	mape_result	0.001
D2	EBM	SHAP_tree_lightgbm	0.193	0.15	mape_result	0.001
D2	EBM	SHAP_tree_xgboost	0.193	0.17	mape_result	0.001
D2	EBM	lime_lightgbm	29.008	30.583	max_error	0.52
D2	EBM	lime_xgboost	29.008	31.907	max_error	0.278
D2	EBM	SHAP_tree_lightgbm	29.008	19.673	max_error	0.001
D2	EBM	SHAP_tree_xgboost	29.008	26.27	max_error	0.002
D2	EBM	lime_lightgbm	59.095	111.356	mean_squared_error	0.002
D2	EBM	lime_xgboost	59.095	91.451	mean_squared_error	0.001
D2	EBM	SHAP_tree_lightgbm	59.095	14.299	mean_squared_error	0.001
D2	EBM	SHAP_tree_xgboost	59.095	27.088	mean_squared_error	0.001

Table 8: XAI fidelity-target metrics results for model comparison using both data sets and only EBM

fleet_group	model_1	model_2	median_1	median_2	metric	pvalue
D1	EBM	SHAP_tree_xgboost	0.778	0.403	faithfulness_error	0.0
D1	EBM	SHAP_tree_lightgbm	0.778	0.784	faithfulness_error	0.645
D1	EBM	lime_lightgbm	0.778	5.016	faithfulness_error	0.0
D1	EBM	EBM_variation	0.778	0.776	faithfulness_error	0.543
D1	EBM	lime_xgboost	0.778	10.814	faithfulness_error	0.0
D1	SHAP_tree_xgboost	EBM_variation	0.403	0.776	faithfulness_error	0.0
D1	SHAP_tree_lightgbm	EBM_variation	0.784	0.776	faithfulness_error	0.706
D1	lime_lightgbm	EBM_variation	5.016	0.776	faithfulness_error	0.0
D2	EBM	SHAP_tree_xgboost	1.188	0.599	faithfulness_error	0.0
D2	EBM	SHAP_tree_lightgbm	1.188	0.802	faithfulness_error	0.014
D2	EBM	lime_lightgbm	1.188	10.019	faithfulness_error	0.0
D2	EBM	EBM_variation	1.188	1.151	faithfulness_error	0.907
D2	EBM	lime_xgboost	1.188	20.029	faithfulness_error	0.0
D2	SHAP_tree_xgboost	EBM_variation	0.599	1.151	faithfulness_error	0.0
D2	SHAP_tree_lightgbm	EBM_variation	0.802	1.151	faithfulness_error	0.005
D2	lime_lightgbm	EBM_variation	10.019	1.151	faithfulness_error	0.0
D1	EBM_variation	lime_xgboost	0.776	10.814	faithfulness_error	0.0
D2	EBM_variation	lime_xgboost	1.151	20.029	faithfulness_error	0.0

Table 9: XAI fidelity-model metrics results for model comparison using both data sets and only EBM

fleet_group	model_1	model_2	median_1	median_2	metric	pvalue
D1	EBM	SHAP_tree_xgboost	0.516	0.689	stability_error	0.068
D1	EBM	SHAP_tree_xgboost	0.439	0.624	xai_stability_error	0.027
D1	EBM	SHAP_tree_lightgbm	0.516	0.363	stability_error	0.032
D1	EBM	SHAP_tree_lightgbm	0.439	0.253	xai_stability_error	0.0
D1	EBM	lime_lightgbm	0.516	1.56	stability_error	0.0
D1	EBM	lime_lightgbm	0.439	0.855	xai_stability_error	0.0
D1	EBM	EBM_variation	0.516	0.712	stability_error	0.097
D1	EBM	EBM_variation	0.439	0.596	xai_stability_error	0.363
D1	EBM	lime_xgboost	0.516	3.336	stability_error	0.0
D1	EBM	lime_xgboost	0.439	1.796	xai_stability_error	0.0
D1	SHAP_tree_xgboost	EBM_variation	0.689	0.712	stability_error	0.468
D1	SHAP_tree_xgboost	EBM_variation	0.624	0.596	xai_stability_error	0.176
D1	SHAP_tree_lightgbm	EBM_variation	0.363	0.712	stability_error	0.0
D1	SHAP_tree_lightgbm	EBM_variation	0.253	0.596	xai_stability_error	0.0
D1	lime_lightgbm	EBM_variation	1.56	0.712	stability_error	0.0
D1	lime_lightgbm	EBM_variation	0.855	0.596	xai_stability_error	0.0
D2	EBM	SHAP_tree_xgboost	0.112	0.174	stability_error	0.008
D2	EBM	SHAP_tree_xgboost	0.133	0.189	xai_stability_error	0.062
D2	EBM	SHAP_tree_lightgbm	0.112	0.201	stability_error	0.003
D2	EBM	SHAP_tree_lightgbm	0.133	0.186	xai_stability_error	0.062
D2	EBM	lime_lightgbm	0.112	0.057	stability_error	0.0
D2	EBM	lime_lightgbm	0.133	0.032	xai_stability_error	0.0
D2	EBM	EBM_variation	0.112	0.128	stability_error	0.93
D2	EBM	EBM_variation	0.133	0.127	xai_stability_error	0.18
D2	EBM	lime_xgboost	0.112	0.395	stability_error	0.0
D2	EBM	lime_xgboost	0.133	0.25	xai_stability_error	0.0
D2	SHAP_tree_xgboost	EBM_variation	0.174	0.128	stability_error	0.002
D2	SHAP_tree_xgboost	EBM_variation	0.189	0.127	xai_stability_error	0.003
D2	SHAP_tree_lightgbm	EBM_variation	0.201	0.128	stability_error	0.001
D2	SHAP_tree_lightgbm	EBM_variation	0.186	0.127	xai_stability_error	0.002
D2	lime_lightgbm	EBM_variation	0.057	0.128	stability_error	0.0
D2	lime_lightgbm	EBM_variation	0.032	0.127	xai_stability_error	0.0
D1	EBM_variation	lime_xgboost	0.712	3.336	stability_error	0.0
D1	EBM_variation	lime_xgboost	0.596	1.796	xai_stability_error	0.0
D2	EBM_variation	lime_xgboost	0.128	0.395	stability_error	0.0
D2	EBM_variation	lime_xgboost	0.127	0.25	xai_stability_error	0.0

Table 10: XAI stability metrics results for model comparison using both data sets and only EBM