A predictive and user-centric approach to Machine Learning in data streaming scenarios

Davide Carneiro[a,b], Miguel Guimarães[a], Fábio Silva[a], Paulo Novais[b]

[a] *CIICESI, ESTG, Politécnico do Porto, Portugal*
[b] *ALGORITMI/Department of Informatics, University of Minho, Portugal*

Abstract

Machine Learning has emerged in the last years as the main solution to many of nowadays' data-based decision problems. However, while new and more powerful algorithms and the increasing availability of computational resources contributed to a widespread use of Machine Learning, significant challenges still remain. Two of the most significant nowadays are the need to explain a model's predictions, and the significant costs of training and re-training models, especially with large datasets or in streaming scenarios. In this paper we address both issues by proposing an approach we deem predictive and user-centric. It is predictive in the sense that it estimates the benefit of re-training a model with new data, and it is user-centric in the sense that it implements an explainable interface that produces interpretable explanations that accompany predictions. The former allows to reduce necessary resources (e.g. time, costs) spent on re-training models when no improvements are expected, while the latter allows for human users to have additional information to support decision-making. We validate the proposed approach with a group of public datasets and present a real application scenario.

Keywords: Meta-learning, Explainability, Streaming Data, Big Data

1. Introduction

The recent growth of Machine Learning (ML) on business tasks and value-added operations is undeniable. Machine Learning algorithms have outpaced

human decision performance in many domains, and this has convinced decision makers to rely increasingly on them [1]. Thus, Machine Learning is nowadays used in virtually all spheres of our existence, controlling our routines in pervasive and transparent ways, and often used to take decisions that have a measurable impact in our lives[2, 3].

The increase in the use of Machine Learning has been accompanied by the increase in the volume of data used. Generally, the more complex the problem/domain is and the larger the dataset, the more complex the models learned are. This poses two main challenges: 1) models are harder to understand and interpret by human decision-makers; and 2) the training of models becomes more expensive in terms of time and resources.

The first challenge poses an interpretability problem: a human user obtains a decision/prediction from a model, but often lacks the necessary information to properly judge and evaluate the outcome. Namely, "how good is it?", "how good are neighboring decisions?", "what is the rationale behind it?", "how does the model behave and how can it's behavior be understood and predicted?". This is especially true when so-called black box models are used, such as Deep Learning, or ensembles such as Random Forests. Coincidentally, these are among the most popular models nowadays.

The second challenge is related to efficient resource management in Machine Learning [4]. Indeed, as the size of datasets increases, so do the necessary resources to train the models and the corresponding costs [5], both in terms of infrastructure/services, time and energy. These costs are also higher in data streaming scenarios or in domains with concept drift [6]. In these cases, frequent updates of the model are necessary, to ensure that it remains up to date with recent data. On the one hand, this encompasses significant costs as the training of models are computationally intensive tasks. On the other, it may ultimately be impractical, if the training of the model takes so much time that it is already outdated when it is finished.

In this paper we propose an integrated approach to deal with both these challenges. While the approach is generic, it is instantiated for a specific case

study on tax fraud detection. The key takeaways from this work are thus:

- We propose a way to build instance-level explanations for any Machine Learning model, using a proxy explainable model that does not require access to the original data but only to the model being explained;
- Explanations are symbolic in the sense that they are based on the concepts of the domain, thus being easier to understand by Human practitioners;
- We propose an approach for predicting model performance, that is especially useful for deciding when to update a model in data streaming scenarios.

The rest of the paper is organized as follows. Section 2 provides some background and related work on the two main topics addressed in this work: explainability and learning in streaming scenarios. Afterwards, Section 3 presents the methodology followed to address the two main challenges tackled in this work, already identified. This section starts by presenting the case study that inspires this work, and then details the methodology followed in each problem individually. Next, the results are detailed in Section 4. The paper ends with a discussion of these results, some concluding remarks, acknowledgement of some limitations, and pointers for future work.

2. Related Work

2.1. Explainability in Machine Learning

Machine Learning problems and algorithms have been growing in complexity in the past years, mostly due to the increase in the volume and complexity of data. As a consequence, models and their behavior are increasingly harder fathom by Humans.

This creates challenges for researchers, regulatory bodies of industry specific applications and other decision makers where there is the need to guarantee principles such as non-discrimination, to adhere to regulations, to comply with the scientific domain, or to follow certain legislation.

Thus, the ability of human agents to *understand* the decision process of models or the rationale behind their predictions is nowadays a common requirement in the field of Machine Learning. It has been addressed by a relatively recent field known as Explainable Artificial Intelligence (xAI)[7]. While xAI is a very broad field that can encompass multiple knowledge domains, in this paper we focus on the aspects more related to Machine Learning models. In this scope, it is important to first make the distinction between two important terms: explainability and interpretability.

While these concepts are often used interchangeably, they represent different notions according to the direction in which information flows [8, 9]. Explainability relates to the ability of the model to detail why and how a given prediction is being made, or how the model behaves internally. Interpretability, on the other hand, is the ability of the human user to understand the explanations provided. Asides from these, the research community has put forward other desirable properties such as transparency - the ability to visualize the inner workings of the model, the ability to use domain-knowledge in the explanation, or its scientific consistency [9].

Depending on the requirements, there can also be varying degrees of explanation. For instance, it is possible to explain a decision process without actually understanding the model which generated such decision, or the intricate relationships between cause and effect in the decision process [1]. An example is the use of saliency maps to explain the classification of an image: while such an explanation may be used to understand a prediction, it is not helpful to understand how a Deep Learning model works.

Building explanations is naturally easier in some models, namely those based on statistical or rule-based algorithms (e.g. Decision Trees). It is much harder and less intuitive in the so-called "black-box" models (e.g. Deep Learning), that are characterized by high complexity and abstraction levels.

Nonetheless, many different approaches are being undertaken in both explainable and black-box models. These approaches are sometimes specific to a given algorithm, or generic and applicable to a broad range of them.

One of the most interesting examples is the use of counterfactuals or evidence based on the interpertability of the model. These require a deep understanding of the Machine Learning model being used and how changes in the input may alter the outcome [10]. These decisions are characterized by the complete categorization of a specific decision and how the decision would be altered given some changes in the input.

This is a generic idea which may have different implementations depending on the algorithm being studied. In the literature this approach can be found in linear classification algorithms [11], where a linear Machine Learning algorithm is exploited to find how changes in coefficients or inputs change the final decision, as well as in black box models such as mutlilayer perceptrons [12].

Indeed, explanations are often more valuable when it comes to black box models, as there is here, clearly, a trade-off between interpretability and accuracy [13]: models that are generally more accurate, such as Deep Learning, are usually also harder to explain. As a consequence, when interpretability and/or explainability are critical project requirements, other less accurate models are often used.

New explainable methods can, however, allow for these complex models to be explained and thus allow their use even in domains where explanations are critical. A common strategy is to use an additional or external algorithm or methodology to provide the necessary explanation, while still using the underlying original model.

Examples of generic and domain-agnostic approaches that follow this strategy are LIME [14] and SHAP [15]. LIME approximates inputs to predictions using a local linear explanation model while SHAP uses metrics represented by SHAP values to denote the expectation of the prediction change in the model based on a determined feature. LIME and SHAP are not mutually exclusive and can be used together [15]. These approaches have been used to explain algorithms' decisions in fields such as intrusion detection [16] or anomaly detection [17].

Other approaches are dependent on the type of black box algorithms be-

ing explained. For instance, in the case of imagery and neural networks the DeepLIFT [18] approach tries to explain which are the most relevant pixels in a image through the analysis of the weight activation in a neural network alike algorithm. Layer-Wise Relevance Propagation [19] is yet another neural networks specific method that interprets the predictions in a similar way to DeepLIFT.

There are, however, additional concerns to be taken into consideration especially when it comes to streaming scenarios, in which ML algorithms must not be stationary in time but be updated. This creates additional concerns with model deployment and adds the need to monitor models and audit models throughout their lifecycle. Even more specifically, there a need to guarantee that local and global explainability and model trustworthiness is maintained between deployments [20]. These issues are analyzed in more detail in the following Section.

Explainable AI will always have to be a two-way road, in which models need to be able to explain decisions in a way that human users can understand them. While some types of explanation can be generic and apply to any problem, domain-dependent ones are probably more valuable as they will add insights that are specific to that problem, and thus more relevant to support decision-making. This is related with the notion of completeness, which is the extent to which an explanation allows a complete understanding of all the domains for each attribute in the decision-making process [21].

The general perception is that all models can be explained to some extent. However, some are easier to explain than others. In any case, explanations should consider the mental model of the user and the domain of the application, something that is not always considered by existing methods. This is the approach to explainability proposed in this paper, which will be further detailed in the following sections.

2.2. Learning in Streaming Scenarios

Current ML algorithms face several issues that stem from the new nature of data (e.g. volume, complexity, velocity, variety) [22]. One of the most significant is to learn from high-speed streaming data. In these scenarios, learning becomes

continuous, and models are often initially trained with small amounts of data, and are later updated or re-trained when new data are available.

There are, however, additional issues. Namely, there is the need to decide which data will be used to train future models or when will future models be trained or updated. There is here clearly a trade-off between the use of computational resources and how outdated a model is.

A small period between model updates or re-trains may lead to a system that adapts faster to changes in the data, but also to one more sensitive to noise. It will also be a more costly system in the sense that it will require more computational resources and time for the constant re-training of the models. Ultimately, depending on the complexity of the model and the velocity at which time changes, it may be impossible to update models in due time.

On the other hand, a system with a large period between updates will require less computational resources, but may perform worse over time as models become obsolete and no longer reflect the patterns in the data.

When learning from streaming data, a key assumption from traditional batch ML systems is violated: that the training and test data are drawn from the same feature space and have the same distribution.

One of the recently proposed techniques for dealing with this is Transfer Learning: the goal is to extract knowledge from a task and use it to improve learning on a different task. Different elements can be transferred, including instances, feature representation or parameters. Various approaches exist, including inductive, transductive and unsupervised transfer learning [23].

These challenges are more significant when there is concept drift [24]. That is, when data, their concepts, their variables or their patterns change over time. In these scenarios, algorithms must learn and forget concepts incrementally, and the act of forgetting becomes as relevant that of learning new concepts.

To deal with these scenarios, the notion of Evolving Ensemble has been proposed [25]. This denotes an Ensemble whose weights are fine-tunned using some optimization mechanism, generally of biological inspiration (e.g. genetic algorithm) [26]. However, when concept drift is too significant, models eventually

have to be replaced. Adaptive Random Forests address this issue by training a new model in the background when concept drift is detected, which later replaces the original model [27].

The approach proposed in this paper differs from the previous ones in the sense that our main goal is to predict the performance of a given model when trained on a given set of data. Thus, rather than updating models at regular intervals or when a certain amount of new data exist, we provide an additional decision layer, that is the predicted performance of the future model. The main goal is to avoid re-training a model if it is not expected to perform better than the previous one, thus more efficiently managing resources.

3. Methodology

3.1. Case Study

Section 1 detailed the relevance of the two main problems being addressed in this work: 1) to predict the performance of a Machine Learning model before its training; and 2) to explain the predictions of a model in a way that can be understood by a human. The importance of explaining decisions in an Intelligent Environment has already been addressed in Section 2. However, nowadays, explanations are not only desirable from a perspective of interpretability but are starting to become a legal requirement. In the context of the GDPR, the EU recently regulated on algorithmic decision-making and, specifically, addressed the issue of a "right to explanation"[28]. There are particularly sensitive domains in which algorithmic decisions significantly affect one's life, such as credit scoring, sentencing, or fraud detection.

This section describes the Case Study that motivates this work, which is inserted in these sensitive areas of application of Machine Learning. Specifically, this work is being developed in the context of the Neurat funded project (31/SI/2017 - 39900). One of the goals of the Neurat project is to develop a Machine Learning environment for tax fraud detection. There are however some particular characteristics, namely:

- There are two types of variable in this problem: static variables (which are obtained through the feature extraction process from the raw data), and dynamic variables (which are proposed by human users);
- Labeled data is scarce since most of the dynamic variables cannot be extracted from the raw data and must be provided manually by human users (some of them reflect subjective or abstract concepts);
- Users' actions change the dataset over time as they validate the models' predictions (contributing to the labeled dataset) or provide their own contribution, for example;
- New raw data is added regularly;
- Models must be updated frequently to adapt to new data. However, new data is not necessarily different from existing data;

The system can thus be described as a cooperative environment in which Machine Learning tools and human experts (auditors) work together to increase the efficiency of tax audits. However, the use of Machine Learning, and in particular of supervised methods, requires vast amounts of labeled data. The problem is that in this case data can only be labeled by auditors and, in this case, it comes at a high cost: auditors must undergo extensive training and their time is very limited. As a consequence, they are able to review but a small portion of the transactions of a company, usually by sampling, and thus provide a small amount of labeled data.

Thus, an Active Learning (AL) approach is being followed to implement it [29]. Generally, AL approaches aim to make Machine Learning less expensive by reducing the need for labeled data. To achieve this, a so-called *Oracle*, which may be a human expert or some automated artifact, is included in an cycle in which a Machine Learning model is continuously improved by training on a growing pool of labeled data. In this case, the Oracle are the auditors.

However, we introduce several major changes to the "traditional" AL scheme (Figure 1). First, we consider a pool of models rather than a single model. In

practice, one model is maintained for each dynamic variable in the problem. This is necessary since dynamic variables, as opposed to static ones, cannot be extracted from the raw data. Thus, one model is maintained to predict each variable. When models are updated, so are the predictions for these variables. Predictions are then validated or changed by the auditors, in what constitutes one way of incorporating human knowledge into the system.

Secondly, we add an additional input to the Oracle. When assessing one instance of data i, the auditor has also access to a prediction p for each dynamic variable, and a corresponding explanation e. The former is provided by each model f associated to each dynamic variable, while the latter is provided by the Explanation Interface as a result of $f(i)$. Now, when the auditor receives the instance to label (that is, when the auditor performs an audit action), he also receives the labels proposed by the system for the data not yet validated, as well as an intelligible explanation for it, tailored for this specific domain.

Finally, we also introduce a fork in the typical AL process in the moment of training new models. While, typically, models are updated at regular intervals or when a certain amount of new labeled data exist, we include a module for predicting the performance of a future model if it is trained with a given dataset. The goal is to avoid training new models if this is not expected to result in a significant performance improvement. The following two sub-sections detail these last two key aspects.

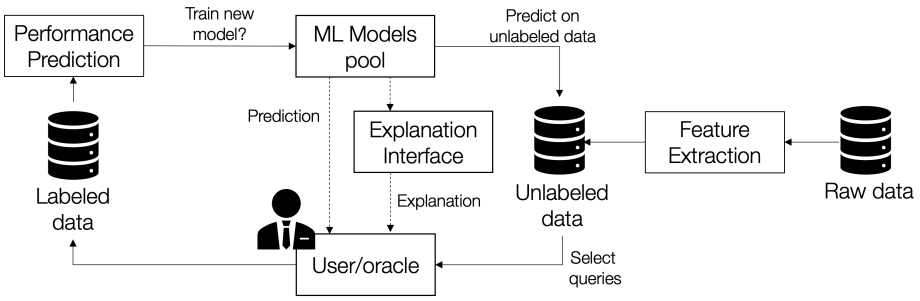

Figure 1: High-level view of the main elements of the proposed system.

3.2. Model Performance Prediction

The main goal of the Performance Prediction module is to predict one or more performance metrics of a future model, if it is trained from a dataset with given statistical properties. The underlying assumption is that certain features of a dataset can be associated to the quality of the resulting dataset. For instance, it is generally accepted that a dataset in which there is a significant amount of missing data leads to a poorer model. Likewise, a larger number of variables and instances tends to result in better models. Other examples could be given, related for instance with the complexity of the data, its distribution, the relationship between its variables, or numerous statistical and information-theoretic features that described the characteristics of a dataset.

This section describes the approach implemented for predicting the performance metrics of a future model. The process by which the approach was implemented and its results validated is detailed in Figure 2.

The process starts by consuming a large amount of datasets, called the training datasets. These datasets are first pre-processed, namely to normalize the values of the dependent variable of each one. Then, these datasets go through a process of meta-feature extraction. Meta-features describe important statistical and information-theoretic characteristics of the dataset, that are expected to be related in some degree to the quality of a model trained with the dataset.

Meta-features are often used in other applications of Machine Learning, such as in AutoML. In a sense, they are like hyperparameters in the search for an optimal model, used to control the learning process. This field, known as meta-learning [30] can be briefly described as *learning to learn*. The goal is to find the best model for a given problem by experimenting with different configurations.

The main different in the proposed work is that our goal is to estimate performance metrics without the need to train models. This may lead to a significant decrease in the necessary computational resources.

Meta-features were extracted using the pymfe library [31]. While this library allows the extraction of hundreds of meta-features, only a subset of 57 were used in this work.

Once the meta-features are extracted, an actual model is trained with the dataset, and its performance metrics are extracted (e.g. RMSE, mae, r^2). Models were trained using the scikit-learn library [32]. The meta-features of all the datasets are then combined with the performance metrics of the corresponding models, to generate what we call the meta-dataset. In this meta-dataset, meta-features are the independent variables while performance metrics and the model training time are the dependent variables. Typically, the meta-dataset has one line for each dataset/model combination, although we followed a different approach for validation purposes, as detailed in Section 4.

Once the dataset has a large enough dimension, the so-called meta-model is trained. This meta-model is trained from the meta-dataset by selecting all the independent variables and one of the dependent variables, corresponding to the performance metric that one intends to predict. The performance of the meta-model itself is obtained through cross-validation.

The previously described approach considers the training of a number of models that can be relatively large. However, once the meta-model is trained, it can be used to predict the performance metric of future models. Given that the models in the pool will only be re-trained if the meta-model predicts a performance improvement, the number of models trained in the long-term is expected to be smaller than if the models in the pool were updated on a regular basis. Moreover, if the meta-model is robust enough, it can be used to predict the performance on any kind of problem.

The goal is, indeed, that it can be used in any Machine Learning problem, independently of the nature of the problem. This can be done by simply extracting the same meta-features of the dataset of the new problem, and using the meta-model for predicting the intended performance metrics. This is detailed in the lower part of Figure 2, in which a group of datasets that were not used for training, deemed test datasets, are used to assess the performance of the meta-model.

The difference, in this case, is that the models are actually trained so that the predictions of the meta-model can be compared against the actual performance

metrics of the models. In a real use of the system, such as that described in the previous section, the meta-model would be used for predicting the performance of a model when trained on a specific set of data, and the model would only be trained if the prediction was for better performance metrics than that of the current model. Thus a reduction in the number of models trained, and consequently in the amount of necessary computational resources, is achieved. Section 4 details the process through which this approach was validated.

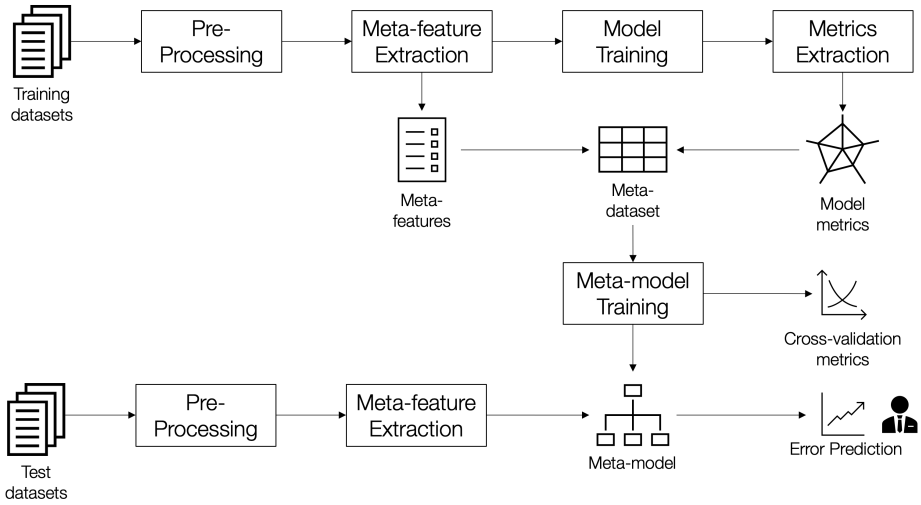

Figure 2: Methodology followed to validate the proposed approach for model performance prediction.

3.3. Explainable Interface

The main goal of the Explainable Interface is to generate elements that can be used to create different types of intelligible explanations for human users. This means that explanations should be domain-dependent and in-line with the users' conceptual models.

To achieve this, we are using a modified version of the CART algorithm[33], an algorithm of the Decision Tree category. A Decision Tree is, in itself, an explainable model: it can be analyzed visually to understand which variables and values are used at each level to take a decision. However, this may be

difficult for example if the tree is too large. There is also additional information that can be provided that is not explicitly in the tree's structure. In this section we detail the explainable elements that are generated by the system, to support the Human auditor in decision-making.

This algorithm allows to build a Decision Tree from a group of observations. Each node of the tree contains boolean rules about the observations (e.g. value of variable x is greater than y) and each leaf contains the result of the prediction for a given path in the tree. While the tree is being built, the training set is increasingly split at each node, leading to smaller sub-sets of the data. This splitting process ends when one or more stopping criteria are met, which may include a minimum size of the split or a minimum degree of variance/purity.

Variance denotes how much the values for the dependent variable of a split are spread around their mean value (in regression tasks), while purity considers the relative frequency of classes: if all classes have roughly the same frequency the node is deemed "impure". The Gini index is used in the CART algorithm to measure impurity [34].

Formula 1, as proposed by [35], describes the relationship between the outcome y and features x. Each instance of the training set is attributed to a single leaf node (subset R_m). $I\{x \in R_m\}$ is a function that returns 1 if x is in the subset R_m or 0 otherwise. In a regression problem the predicted outcome $\widehat{y} = c_l$ of a leaf node R_1 is given by the average value of the instances in that same node. The algorithm can be used for both classification and regression tasks.

$$\widehat{y} = \widehat{f}(x) = \sum_{m=1}^{M} c_m I\{x \in R_m\} \tag{1}$$

The core of the Explanatory Interface is thus a tree-based model, so-called *explanatory model*. A different explanatory model is trained to explain the predictions of each model in the pool.

If there is access to the original dataset, the explanatory model can be trained with the original data or with a subset of it.

However, if there is no access to the original dataset as in cases of proprietary

models, the explanatory interface is able to generate a synthetic dataset given some basic meta-data about the original dataset (e.g. minimum and maximum value for each variable). First, the data corresponding to the independent variables are generated, at random. The main goal is to cover the search space as much as possible. However, it may also happen that data instances that are unrealistic, in the sense that they would not happen in real life, are generated. While this may lead to an increased complexity of the explainable model, there are no other disadvantages in the sense that, if those instances never happen in real life, no explanations will ever be requested for them either. If they were, these would eventually be nonsensical. These random dataset is then fed to the original model, so that it can predict on each of its instances. The resulting synthetic dataset is then used to train the explainable model. Figure 3 details the process of training an explainable model when there is no access to the original dataset by the explanation interface. When this access exists, the original dataset is used directly to train the explainable model.

Whether the explainable model was trained with or without access to the original data, its goal is to be used in parallel with the original model. That is, whenever a new prediction is issued by the original model, its corresponding explainable model does a prediction as well, in order to generate the elements that are used to create explanations.

The algorithm to create the explainable models works as follows. When the tree is being built and each split generated, additional information is stored in the node which includes: the boolean rule that generates the split (mentioning the variable and the value interval), the prediction $\widehat{y}$ based on that split (i.e. the average or most frequent value, depending on the problem), measures of dispersion or purity (variance, standard deviation and Gini index), and the indexes of the instances in the split.

These values are then used to provide a notion of *confidence* and *support* to the decision-maker. Confidence is given by dispersion and purity measures: the lower the dispersion or the higher the purity, the higher the confidence on the decision is. Support is given by the number of instances in the split: the higher

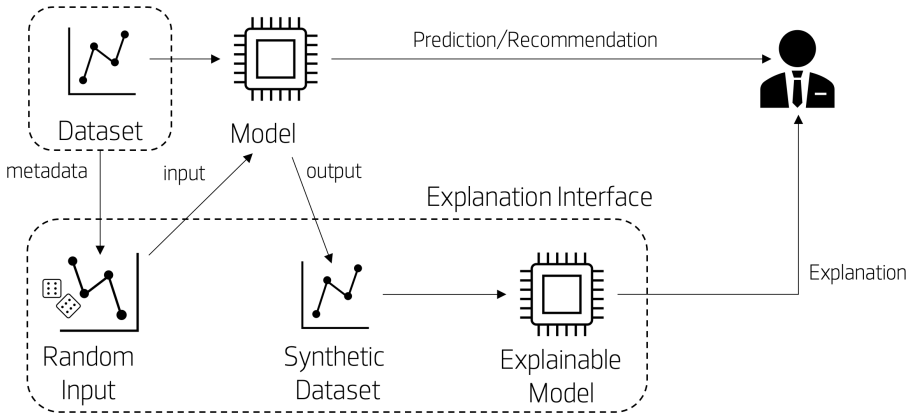

Figure 3: Process for generating explanations when there is no access to the original dataset.

the number of instances, the higher the support is.

This information on the nodes allows to incorporate a group of explainable elements in the user interface. Figure 4 shows a prototype of the graphical user interface that is used to provide explanations. When an auditor wants to analyze a specific instance she/he selects that instance and is redirected to this interface, which receives the data of the instance, the prediction, and an explanation. The user interface has three main areas, marked in the Figure as (a) - Explore, (b) - Decision path and (c) - Last results.

Area (a) allows the user to explore the search space and analyze each feature according to their relative importance. Features and values are collected from the internal nodes when traversing the tree to make a prediction. In this context, feature relevance is based on how much that split/feature decreases dispersion/purity. For each feature that the interface shows the following elements (depending on whether the variable is numeric or nominal): the domain of the feature (range/enumeration of possible values), the interval/values for which the prediction holds (blue bar or values highlighted in blue), and the value of the feature in the instance being audited (gray dot).

This allows the auditor to gain a sense of how *risky* the decision is. If the value of a given feature is very close to the upper or lower limits of the blue

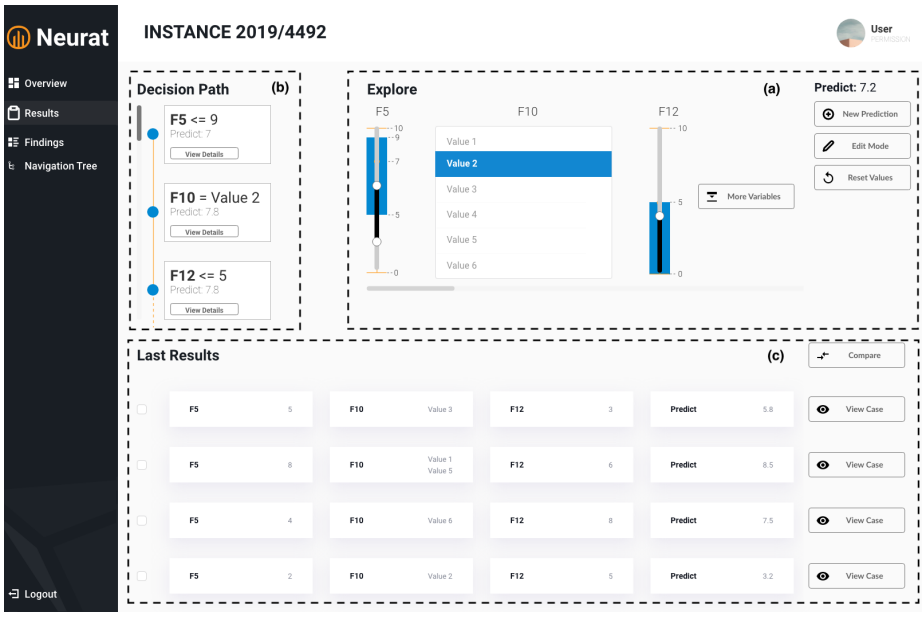

Figure 4: Prototype of the main screen of the application, with some of the explainable elements created, and three main areas highlighted: Explore (a), Decision Path (b) and Last Results (c).

bar, it indicates that a slight change of this feature towards the limit would significantly alter the prediction of the tree. Likewise, the size of the blue bar is also related to this sense of risk: the shorter the bar the more risky the decision is. In the case of a nominal feature, multiple values can be highlighted to show for which values of the enumeration the prediction holds. The risk of the decision grows with fewer highlighted values.

In Figure 4, the graphical interface is shown in "Edit Mode". This means that the user may change the values of the variables to perform a counterfactual analysis. That is, what would be the prediction if the value of a feature had been v_2 instead of v_1. These "what-if" scenarios allow the auditor to interact with the tree and to understand how predictions would change under different scenarios. This contributes significantly to the interpretability and interactivity of the explanation, as addressed in Section 2. The user does this by changing the value of the features by means of a slider, or by selecting a value from a list.

The scenarios created by the user can be added to area (c), to be compared. The user can also reset area (a), returning all the values and the associated prediction to the initial state of the instance being audited.

There is also a pagination mechanism that controls the amount of information provided to the user, to avoid overload. Indeed, depending on the training set, the number of levels/nodes/features on a tree may be too large to be efficiently analyzed by a Human. In that sense, in this interface we show only the n most relevant features. The user can then choose to request additional features (and the associated prediction) by clicking on the "More variables" button. These are gradually added upon request by decreasing relevance.

In the left side of the interface there is the area marked as (b). This area shows the path followed through the tree to make the prediction. Like in (a), this area may not show the whole path as it implements the same pagination mechanism: when features are added to (a) they are also added to (b). This element allows the user to understand (part of) the reasons for a given prediction: "because feature f_1 is smaller or equal than v_1 and feature f_2 equals v_2".

In this area the user may also click on a specific node to see its details (Figure 5). The details show, in the left side, the information for the feature that is also visible on (a). On the center and right, the "details" modal provides information regrading the *confidence* and *support* of the prediction. The graphical representation shows the prediction (blue dot) and the interval given by the standard deviation. A smaller interval indicates an increased confidence as instances in this split are more closely distributed around the mean, and vice-versa.

The central part of the modal shows values which include the support (number of instances in this split) and a button that allows the user to access the instances that fall into this split. The user may thus visualize the instances, which are shown sorted by similarity to the current instance in descending order. Similarity is calculated based on a weighted sum of differences, given by the euclidean distance for numerical variables and by the cosine similarity for the vector of nominal data (if any). While visualizing specific instances the user may add them to a list for comparison (area (c)).

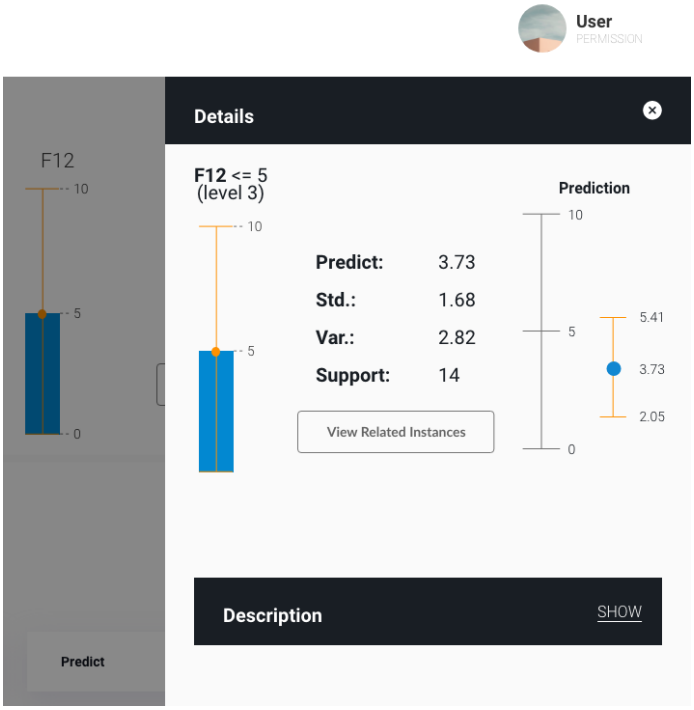

Figure 5: Details of a split node, with confidence and support measures.

As the user moves down the path, splits become smaller but confidence increases. It is up to the user to decide how far down to travel: an early stop may lead to a more general decision (with high support and potential low confidence), while going further down will lead to low support but high confidence. Finally, in area (c) the user has access to a list of previous prediction results (the scenarios that were simulated) and/or to actual instances that were visualized by the user and added for comparison. This allows to more easily compare a group of scenarios or real cases and their results.

4. Results

The methodology proposed for creating a meta-dataset for predicting model performance, detailed in Section 3.2, was instantiated to assess its suitability to solve the proposed challenge. This section details the results obtained.

A total of 53 datasets were used in the process, covering both regression and classification problems, as well as streaming and batch scenarios. Table 1 describes some of them. Of these, 47 were used for training the meta-model and 6 were used for testing it.

Concerning specifically the test datasets, the first 3 are batch datasets while the other 3 are streaming datasets. Given that streaming and batch ML problems are fundamentally different, the meta-model was evaluated differently for each case.

Table 1: Characterization of some of the 53 datasets used for the validation of the proposed approach. The first 17 are part of the 47 datasets used for training the meta-model while the last 6 are used for testing it (3 batch, 3 streaming). (R: Regression, B: Binary Classification)

Dataset	Source	Type	*N*	Features
School grades	https://www.mldata.io/dataset-details/school_grades/	R	649	33
Cardiovascular diseases	https://kaggle.com/aiaiaidavid/cardio-data-dv13032020	B	10000	12
Killed or Seriously Injured	https://kaggle.com/jrmistry/killed-or-seriously-injured-ksi-toronto-clean	B	12557	56
Contains Aditives	https://kaggle.com/jadeblue/openfoodfactsclean	B	774	13
Starbucks proteins	https://kaggle.com/jadeblue/openfoodfactsclean	R	243	7
McDonalds proteins	https://kaggle.com/jadeblue/openfoodfactsclean	R	260	6
Medical Cost	https://kaggle.com/mirichoi0218/insurance	R	1338	7
Car Price Prediction	https://kaggle.com/hellbuoy/car-price-prediction	R	205	26
Social Network Ads	https://kaggle.com/dragonheir/logistic-regression	B	400	5
Abalone	https://www.mldata.io/dataset-details/abalone/	R	4177	9
Auto mpg	https://www.mldata.io/dataset-details/auto_mpg/	R	398	9
Exercise Calories	https://kaggle.com/fmendes/fmendesdat263xdemos	R	9000	8
Computer Hardware	https://www.mldata.io/dataset-details/computer_hardware/	R	209	10
Forbes Billionaires	https://www.mldata.io/dataset-details/forbes_billionaire/	R	2043	6
House Price	https://kaggle.com/harlfoxem/housesalesprediction	R	4600	19
Fraud Detection	Proprietary	R	22225	13
Wine quality (red)	https://kaggle.com/uciml/red-wine-quality-cortez-et-al-2009	R	1599	12
Wine quality (white)	https://data.world/uci/wine-quality/workspace/data-dictionary	R	4500	12
Diabetes	https://kaggle.com/kandij/diabetes-dataset	B	768	9
Breast Cancer	https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/	B	699	10
Airlines	https://moa.cms.waikato.ac.nz/datasets/	B	10000	8
AWS Prices	https://moa.cms.waikato.ac.nz/datasets/	R	10000	8
Electricity	https://moa.cms.waikato.ac.nz/datasets/	B	10000	9

One challenge of this work is indeed to obtain a number of datasets that is large enough for building a sufficiently representative meta-dataset. This is related not only to the sheer number of datasets, but also to the variety of their statistical properties: datasets that are similar to others do not add value. The list of datasets considered has both regression and binomial classification

problems. However, these must be dealt with differently as they have different performance metrics and the training process is also different. To this end, all binomial classification problems have been transformed into regression problems by replacing the binary labels with the values 0 and 1.

We also acknowledge that 47 datasets for training is also a relatively small number, as it would result in a meta-dataset with only 47 instances. In order to increase the number of instances, the following approach was followed. Each dataset was streamed in blocks of 200 instances. A first model is trained with the first 200 instances, and additional models are trained by adding 200 instances to the previous dataset. That is, for a dataset with 650 instances, 4 models are trained: the first with 200 lines, the second with 400, the third with 600, and the last one with the whole 650. For training each of these models, a Random Forest algorithm was used. Each model is constituted by 20 trees, each with a maximum depth of 20 levels.

Given the number and size of the datasets used for training (partially detailed in Table 1), and the approach described, the resulting meta-dataset contains a total of 673 instances.

While the instances in the meta-dataset generated from a same dataset are not very diverse, which is a limitation, this allows to create a relatively large meta-dataset with a small number of input datasets. However, in future work we will continue to include additional datasets so that the meta-model learns to predict on a more variate range.

The meta-dataset is also composed of 111 columns. Of these, 105 correspond to the meta-features extracted from the datasets, while the remaining six correspond to relevant performance metrics: training time, RMSE, mae, mse, rmsle and r^2. This means that the meta-dataset can be used to train six different models, one to predict each of these individual metrics. However, in this paper we only trained one model to predict the RMSE and focus our analysis on this metric.

Table 4 describes some of the meta-features considered. The criteria for selecting a sample of the meta-features was based on the relative importance of

each one during the training of the meta-model.

Table 2: Ten most relevant meta-features, out of 105 used to build the meta-model.

Meta-feature	**Scaled Importance**	**Description**
linear_discr	100	Linear Discriminant classifier
mut_inf.sd	48.38	Standard deviation of mutual information
sparsity.sd	29.87	Standard deviation of sparsity metric
eq_num_attr	20.21	Attributes equivalent for a predictive task
one_itemset.sd	14.15	Standard deviation of one itemset
elite_nn.sd.relative	10.21	Performance of Elite Nearest Neighbor
one_itemset.mean	10.05	Mean of one itemset
ns_ratio	9.73	Noisiness of attributes
attr_ent.mean	8.29	Mean of Shannon's entropy
mut_inf.mean	7.25	Mean of mutual information

Once the meta-dataset was built, the meta-model was trained. The same algorithm (Random Forest) and configuration used in the training of each model was also used to train the meta-model. To assess the quality of the meta-model during the training phase, a 10-fold cross-validation approach was followed, as depicted in Figure 2. Metrics were computed for each holdout prediction, and averaged at the end, resulting in the following values: $RMSE = 0.000355$, $r^2 = 0.98$ and $mae = 0.007$.

Next, the performance of the meta-model was assessed on the test datasets. To this end, a different approach was followed depending on whether it was being evaluated on a batch or streaming dataset. In any case, the same meta-model already trained was used as the basis for evaluation.

For batch datasets, its meta-features were extracted and the performance of a model (RMSE) was predicted by the meta-model. An actual Random Forest was then trained with each batch dataset, and its actual RMSE value was obtained. Table 3 shows, for each dataset, the RMSE predicted by the meta-model and the actual RMSE of the trained Random Forest.

Table 3: Observed vs. predicted RMSE for the three test datasets.

Dataset	**Observed RMSE**	**Predicted RMSE**
Wine quality (white)	0.102	0.111
Diabetes	0.400	0.321
Breast Cancer	0.161	0.093

Streaming datasets were tested differently, to simulate a realistic streaming scenario. Specifically, for each dataset, data were streamed in blocks of 200 instances. For each block, its meta-features were extracted, and the meta-model was used to predict the performance of a model trained on that block. Then, a Random Forest was actually trained with the block. The RMSE of the resulting model was compared with the RMSE predicted by the meta-model (Figure 6), to assess the quality of the meta-model. For each model trained, a new instance was added to the meta-dataset, as previously described. The meta-model was updated at 10 block intervals (2000 instances of streamed data).

So, the main difference between the batch and the streaming scenario is that in the latter the meta-model is updated as new data is received.

Given the size of the Wine Quality dataset, it was tested following both the batch and the streaming approach.

Figure 6 shows the observed vs. the predicted RMSE for the 3 streaming datasets and the Wine Quality one. The data points are generally close to the diagonal, which indicates a relatively good predictive performance. However, in the AWS dataset there are some instances in which the prediction is slightly off, and in the Wine Quality dataset there seems to be a tendency to overestimate the value of the RMSE. Nonetheless, the values are close to the diagonal.

Table 4 shows how the performance of the meta-model evolved over time, as new versions of it were trained including the streaming data received so far. It can be seen that, even for streaming data, the performance of the meta-model is maintained or improves over time, as new data arrives and is incorporated into the meta-model.

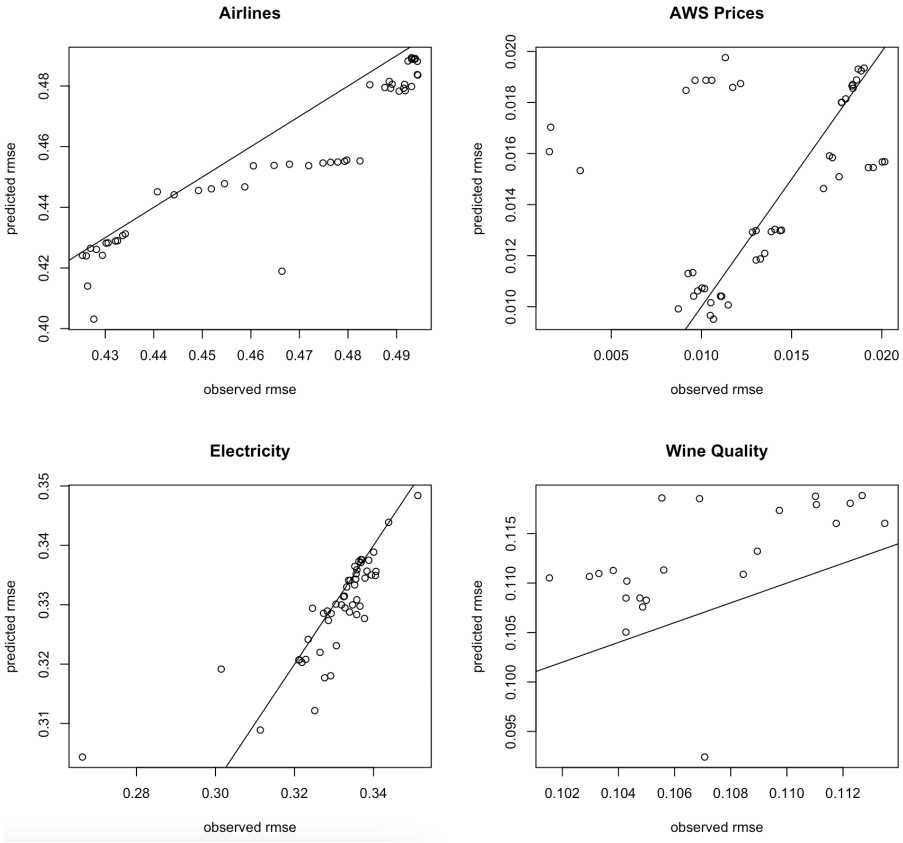

Figure 6: Observed vs. predicted RMSE for the 3 streaming datasets and the wine quality dataset, with models trained at 200-instance intervals. The solid line represents the diagonal.

Figure 7, on the other hand, shows the evolution of RMSE for the Wine Quality dataset over time, as each new model was trained using 200 additional instances of data. It shows a tendency of the RMSE of the models to decrease as more instances of data are included. The RMSE predicted by the model tends to follow this decrease, albeit with a positive bias, as described previously. The predictions are, nonetheless, in line and close to the observed values.

Finally, Figure 8 shows the evolution of a selected group of meta-features of each consecutive subset of data, and how it relates to the evolution of the RMSE. While the RMSE data is the same of Figure 7, the different scale makes it more difficult to observe its descent over time.

Table 4: Evolution of the RMSE of the meta-model as more instances of data are incorporated.

Dataset	N**=2000**	N**=4000**	N**=6000**	N**=8000**	N**=10000**
Airlines	0.018	0.005	0.020	0.0105	0.006
AWS Prices	0.0103	0.001	0.001	0.003	0.0003
Electricity	0.013	0.008	0.003	0.003	0.001

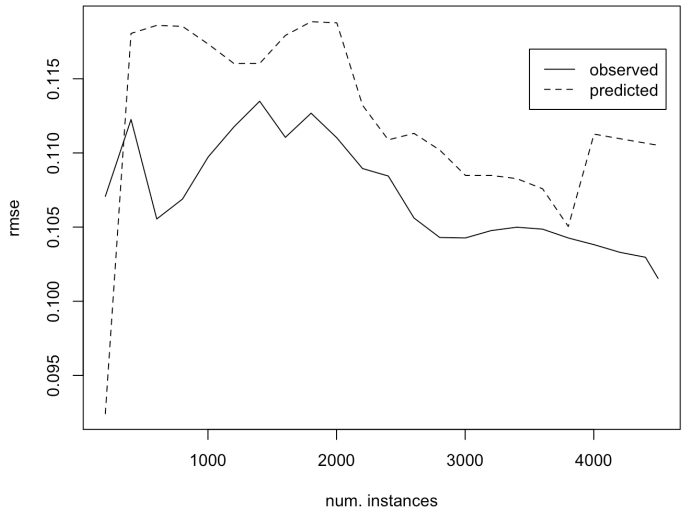

Figure 7: Evolution of the predicted and observed RMSE for the wine quality dataset, with models trained at 200-instance intervals.

5. Discussion and Limitation

In the last years, Machine Learning has seen a tremendous growth in its capabilities and applications. However, significant challenges still remain. These challenges stem mostly from the new characteristics of data: they are in unprecedented volume, and they are generate faster than ever. The resources to handle these data and to train models with them, for example, must thus also be greater. This is especially so in streaming scenarios, in which new data, with potentially different statistical properties, is added frequently, rendering the models potentially outdated.

At the same time, organizations rely increasingly on data and data-based

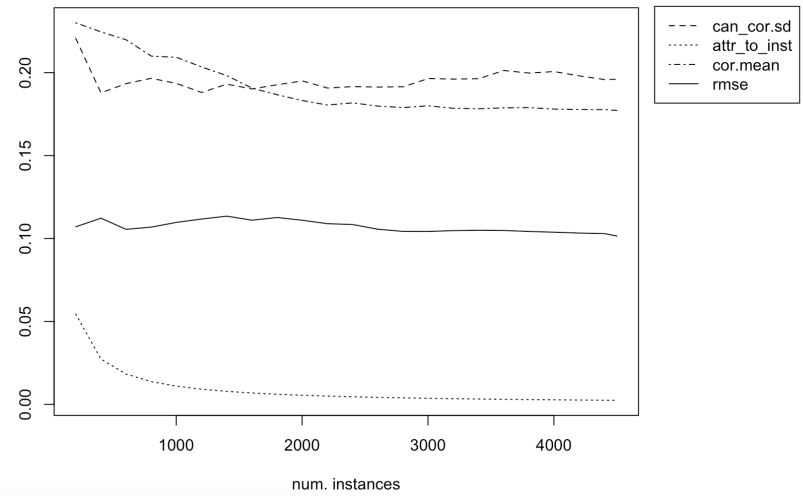

Figure 8: Evolution of the values of several meta-features and corresponding RMSE for the wine quality dataset, with models trained at 200-instances intervals.

techniques for decision support. While many decision problems are well structured and can be fully automatized, others require human decision capabilities and accountability. In these cases, Machine Learning models play a role of decision-support rather than decision-maker. In these cases, especially, it is important that human decision-makers understand how models work and how and why a given prediction is being made by a model, or what is the risk or certainty associated to that decision. This is especially so in domains that affect people's lives, such as in credit concession or in courtrooms. However, as current problems become more complex, so do Machine Learning models. Consequently, models and their predictions are also harder to explain.

In this paper we addressed these two main problems. One the one hand we propose an approach for minimizing the necessary resources to manage Machine Learning models in data streaming scenarios, by predicting the performance metrics of models before they are trained. Using this information, a model will only be re-trained if the statistical properties of the new data indicate that it will lead to a better model.

On the other hand, we propose an approach for generating explanations for

the predictions of a model, in the form of a so-called explanatory interface. The key element is a tree algorithm, that is able to generate elements during training that can be used to create intelligible explanations. These explanations are symbolic and domain dependent, based on features, their relevance, and their statistical properties. They are, therefore, easier to be understood by domain experts.

One of the key aspects of both approaches is that they are independent of the domain of application. That is, while we present a case study in the field of tax fraud detection, which is the domain of the problem that inspired this work, the proposed approaches can be applied in any problem. This is because the approach is based on properties of the dataset (the meta-features) and not the dataset itself. And the meta-features can be obtained from any dataset.

Nonetheless, it must also be stated that performance may vary significantly from one problem to the next, namely if the dataset of a new problem has very different properties. Nonetheless, once the new dataset is incorporated into the meta-dataset, it will contribute to improve the meta-model. Thus, the meta-model becomes increasingly better and more generalized as more datasets are added.

The number of datasets used to create the meta-dataset is indeed the main limitation of this work. This means that there is the risk of the meta-dataset not being representative enough to have a good performance on every problem that it is used on. A relatively small number of test datasets was also used. In order to overcome this limitation, the data in each dataset was divided into multiple subsets of increasing size, to allow for the training of more models and thus increasing the size of the dataset. To some extent this is, however, also a limiting factor as these sub-sets share many of the statistical properties of the entire dataset.

In principle, this would be more problematic in streaming scenarios, in which data and its patterns change over time. Nonetheless, we have shown that when the meta-model is updated at regular intervals with new data, it can adapt and maintain or even improve its performance.

The preliminary results are encouraging. We will continue to obtain and curate additional datasets in order to increase the robustness and scope of the meta-dataset.

Another limitation of this work is that only regression problems were considered. Indeed, regression and classification problems must be implemented differently. Given that this is an early validation of the proposed approach and a first step towards its full implementation, the decision was to start with regression problems. While some binomial classification problems were included, these were transformed into regression as well. In future work we will validate this approach for classification problems, in which we expect a similar degree of success.

Finally, in future work we will also use this approach while including multiple algorithms rather than Random Forest alone. This will allow the use of the meta-model for predicting the best algorithm to use, asides from the expected training time and performance metrics.

All in all, we believe that this meta-learning based approach can prove interesting towards a more efficient and intelligent management of the processes associated to model training and scoring, especially in scenarios in which there are large volumes of streaming data.

6. Acknowledgments

This work was supported by the Northern Regional Operational Program, Portugal 2020 and European Union, trough European Regional Development Fund (ERDF) in the scope of project number 39900 - 31/SI/2017, and by FCT - Fundação para a Ciência e a Tecnologia, through projects UIDB/04728/2020 and UIDB/00319/2020.

References

[1] F. K. Dosilovic, M. Brcic, N. Hlupic, Explainable artificial intelligence: A survey, 2018 41st International Convention on Information and Com-

munication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings (May) (2018) 210–215. doi:10.23919/MIPRO.2018.8400040.

[2] D. Ververidis, C. Kotropoulos, I. Pitas, Automatic emotional speech classification, in: 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 1, IEEE, 2004, pp. I–593.

[3] K. Crawford, Artificial intelligence's white guy problem, The New York Times 25 (2016).

[4] E. García-Martín, C. F. Rodrigues, G. Riley, H. Grahn, Estimation of energy consumption in machine learning, Journal of Parallel and Distributed Computing 134 (2019) 75–88.

[5] E. García-Martín, N. Lavesson, H. Grahn, E. Casalicchio, V. Boeva, How to measure energy consumption in machine learning algorithms, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2018, pp. 243–255.

[6] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, F. Petitjean, Characterizing concept drift, Data Mining and Knowledge Discovery 30 (4) (2016) 964–994.

[7] R. Goebel, A. Chander, K. Holzinger, F. Lecue, Z. Akata, S. Stumpf, P. Kieseberg, A. Holzinger, Explainable ai: the new 42?, in: International cross-domain conference for machine learning and knowledge extraction, Springer, 2018, pp. 295–303.

[8] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion 58 (2020) 82–115. arXiv:1910.10045, doi:10.1016/j.inffus.2019.12.012.

[9] R. Roscher, B. Bohn, M. F. Duarte, J. Garcke, Explainable Machine Learning for Scientific Insights and Discoveries, IEEE Access 8 (2020) 42200–42216. arXiv:1905.08883, doi:10.1109/ACCESS.2020.2976199.
URL https://ieeexplore.ieee.org/document/9007737/

[10] K. Sokol, P. Flach, Desiderata for interpretability: Explaining decision tree predictions with counterfactuals, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 10035–10036.

[11] B. Ustun, A. Spangher, Y. Liu, Actionable recourse in linear classification, in: FAT* 2019 - Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency, 2019, pp. 10–19. arXiv:1809.06514, doi:10.1145/3287560.3287566.

[12] F. Silva, C. Analide, Information asset analysis: credit scoring and credit suggestion, International Journal of Electronic Business 9 (3) (2011) 203. doi:10.1504/ijeb.2011.042542.
URL http://dx.doi.org/10.1504/ijeb.2011.042542

[13] S. Saisubramanian, S. Galhotra, S. Zilberstein, Balancing the tradeoff between clustering value and interpretability, in: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, 2020, pp. 351–357.

[14] M. T. Ribeiro, S. Singh, C. Guestrin, "Why should i trust you?" Explaining the predictions of any classifier, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 13-17-August-2016 (2016) 1135–1144. arXiv:1602.04938, doi:10.1145/2939672.2939778.

[15] S. M. Lundberg, S. I. Lee, A unified approach to interpreting model predictions (2017). arXiv:1705.07874v2.

[16] M. Wang, K. Zheng, Y. Yang, X. Wang, An explainable machine learning framework for intrusion detection systems, IEEE Access 8 (2020) 73127–73141. doi:10.1109/ACCESS.2020.2988359.

[17] L. Antwarg, B. Shapira, L. Rokach, Explaining Anomalies Detected by Autoencoders Using SHAP, arXiv (2019) 1–37arXiv:1903.02407.

[18] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, 34th International Conference on Machine Learning, ICML 2017 7 (2017) 4844–4866. arXiv:1704.02685.

[19] B. S., B. A., M. G., K. F., M. K-R., S. W., On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation, PLoS ONE 10(7) (2015).

[20] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. Moura, P. Eckersley, Explainable machine learning in deployment, FAT* 2020 - Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (2020) 648–657arXiv:1909.06342, doi:10.1145/3351095.3375624.

[21] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, L. Kagal, Explaining explanations: An overview of interpretability of machine learning, Proceedings - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA 2018 (2019) 80–89arXiv:1806.00069, doi:10.1109/DSAA.2018.00018.

[22] Q. Yang, Y. Zhang, W. Dai, S. J. Pan, Transfer learning, Cambridge University Press, 2020.

[23] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, X. Wang, A survey of deep active learning, arXiv preprint arXiv:2009.00236 (2020).

[24] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, IEEE Transactions on Knowledge and Data Engineering 31 (12) (2018) 2346–2363.

[25] D. E. Kvasov, M. S. Mukhametzhanov, Metaheuristic vs. deterministic global optimization algorithms: The univariate case, Applied Mathematics and Computation 318 (2018) 245–259.

[26] A. Chandra, X. Yao, Evolving hybrid ensembles of learning machines for better generalisation, Neurocomputing 69 (7-9) (2006) 686–700.

[27] H. A. Bashir, R. S. Neville, Hybrid evolutionary computation for continuous optimization, arXiv preprint arXiv:1303.3469 (2013).

[28] B. Goodman, S. Flaxman, European union regulations on algorithmic decision-making and a "right to explanation", AI magazine 38 (3) (2017) 50–57.

[29] B. Settles, From theories to queries: Active learning in practice, in: Active Learning and Experimental Design workshop In conjunction with AISTATS 2010, 2011, pp. 1–18.

[30] J. Vanschoren, Meta-learning: A survey, arXiv preprint arXiv:1810.03548 (2018).

[31] E. Alcobaça, F. Siqueira, A. Rivolli, L. P. Garcia, J. T. Oliva, A. C. de Carvalho, et al., Mfe: Towards reproducible meta-feature extraction, Journal of Machine Learning Research 21 (111) (2020) 1–5.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.

[33] S. Singh, P. Gupta, Comparative study id3, cart and c4. 5 decision tree algorithm: a survey, International Journal of Advanced Information Science and Technology (IJAIST) 27 (27) (2014) 97–103.

[34] R. I. Lerman, S. Yitzhaki, A note on the calculation and interpretation of the gini index, Economics Letters 15 (3-4) (1984) 363–368.

[35] C. Molnar, Interpretable machine learning, Lulu. com, 2019.