Using meta-learning to predict performance metrics in Machine Learning problems

Davide Carneiro[1,2], Miguel Guimarães[1], Mariana Carvalho[1], and Paulo Novais[2]

[1] CIICESI,ESTG, Politécnico do Porto, Portugal
{dcarneiro,8150520,mrc}@estg.ipp.pt
[2] Centro ALGORITMI, Universidade do Minho, Portugal
pjon@di.uminho.pt

Abstract. Machine Learning has been facing significant challenges over the last years, much of which stem from the new characteristics of Machine Learning problems, such as learning from streaming data or incorporating Human feedback into existing datasets and models. In these dynamic scenarios, data change over time and models must adapt. However, new data do not necessarily mean new patterns. The main goal of this paper is to devise a method to predict a model's performance metrics before it is trained, in order to decide whether it is worth it to train it or not. That is, will the model hold significantly better results than the current one? To address this issue we propose the use of meta-learning. Specifically, we evaluate two different meta-models, one built for a specific Machine Learning problem, and another built based on many different problems, meant to be a generic meta-model, applicable to virtually any problem. In this paper we focus only on the prediction of the rmse. Results show that it is possible to accurately predict the rmse of future models, event in streaming scenarios. Moreover, results also show that it is possible to reduce the need for re-training models between 60% to 98%, depending on the problem and on the threshold used.

Keywords: Interactive Machine Learning, Meta-Learning, Error Prediction

Conflict of Interest

The authors declare that they have no conflict of interest to declare.

1 Introduction

Due to technological development and the growth of data use over the years, organizations are currently dealing with very demanding and competitive scenarios on a daily basis. Organizations feel the pressure for rapid evolution and need to invest and resort to new applications, technologies and tools in order to gain a competitive advantage. In this context, Machine Learning has been increasingly used to allow organizations to achieve competitive advantage.

Due to this increased and diverse use, very different application scenarios and requirements for Machine Learning can be found, which also led to different approaches to Machine Learning. A particularly interesting approach is that of Active Learning (AL) Aggarwal et al. (2014). The key idea behind AL is that, if allowed to choose from the training data, a ML algorithm can achieve higher accuracy with fewer training samples. This is particularly important in domains in which labeled data are hard or costly to get. When the process of obtaining labeled data is implemented through the interaction with Human experts, this form of Machine Learning is also often called interactive Machine Learning (iML) Fails & Olsen Jr (2003).

iML can be defined as a continuous learning process that involves the interaction of human experts: they analyze and evaluate its progress and integrate some feedback, for example, by adding relevant variables to the model or even removing uninteresting ones, whenever necessary. As the human experts are responsible for monitoring and evaluating the entire learning process, this interaction usually leads to improved model outcomes and performance.

When iML approaches are used, one common concern is the occurrence of significant changes in the structure, content or even the size of datasets over time. As a consequence, trained models can quickly become outdated, loose relevance, or even fail to represent the data patterns correctly. However, these problems are not exclusive to iML systems, as they can also happen in applications which resort to learning with streaming data Krawczyk et al. (2017) and also when dealing with concept drift Widmer & Kubat (1996).

Knowing *how* and *when* to update a given model is thus crucial in these scenarios. The *how* was already addressed by the research team in previous work Ramos et al. (2020, 2019). This paper focuses on the *when*. Indeed, there is a tradeoff between the usability or adequacy of the model, and the computational resources and time that are needed to keep it up to date with the most recent data.

The main goal of this paper, which extends the work published in Carneiro et al. (2021), is to propose a method to minimize the need for retraining models in ML use cases in which data change over time. The use case is a fraud detection system that maintains multiple machine learning models, that are updated regularly as new labeled data is available. However, most of the times, the new models are not significantly better than the previous ones, and their training could have been avoided, thus saving computational resources and time.

To be able to avoid the unnecessary training of models, we propose the use of a meta-model whose function is to predict the performance metrics of a given model if it is trained on data with certain statistical properties. Performance metrics include training time and well-known indicators such as rmse, mae or r^2. The statistical properties of data are obtained and represented through meta-features: features that describe the original features of the dataset, such as missing data or the way data are distributed. The key idea is that a model for a given ML problem for which there are new data is only retrained if there are new and unknown patterns in the data that justify it. Indeed, new data does not

necessarily mean new patterns. This will allow organizations to save valuable resources and time.

Specifically, in this paper we evaluate two different meta-models and their ability to predict the performance metrics of models trained in the domain of fraud detection. The first meta-model, deemed M_1 was developed only with data from a proprietary fraud detection dataset. The second, deemed M_2, was developed with data from 50 different datasets (including the fraud detection dataset), and can be seen as a general meta-model, whose aim is to predict performance metrics in virtually any ML problem.

We thus seek to answer the following research questions:

1. Is it possible to predict the performance metrics of a given model before it is trained?
2. If so, will a specific meta-model developed for this purpose perform better than a general-purpose one?

The significance of 1) resides on the ability to predict model performance, and thus minimize the amount of models that need to be trained in interactive learning scenarios. The potential advantages of this can be furthered if it can be shown that a general-purpose meta-model, built from the statistical properties of many datasets, can have better performance than a specific one, as in this case there is no need to collect data and train a meta-model for each specific application case. Moreover, a specific meta-model must be trained from scratch for each ML problem, and there is a setup time in which an initial amount of data is being collected to build the meta-model, so the meta-model can still not be used. On the contrary, a general meta-model is trained and exists beforehand, so it could be user right away at the onset of a ML project.

The paper thus starts with a description of the problem and context, and then sets out to find the answers for these two main research questions. It is organized as follows. Section 2 describes some related work in the field of interactive Machine Learning and its multiple application domains. Section 3 describes the use case that sparked the development of this work: an interactive machine learning system for financial fraud detection in which human auditors contribute with labeled data over time. Next, Section 4 describes the proposed approach for predicting model performance, which is used to minimize the necessity for updating or retraining models. Section 5 details the process through which the proposed approach was validated, and the corresponding results. Finally, the conclusions, limitations of the work, and future research directions are discussed in Section 6.

2 Related Work

In recent years, iML has drawn a lot of attention from the research community and evolved into several domain fields Jiang et al. (2019). In Holzinger & Jurisica (2014), the authors contribute with an overview on integrative and interactive solutions for knowledge discovery in bioinformatics. The authors refer to the

importance of automated and interactive machine learning solutions given the fast-paced growth of data and complex datasets.

The authors later follow up on their work in the health informatics domainHolzinger (2016) showing that using iML can bring advantages when compared to the use of automatic machine learning algorithms when dealing with small datasets, or datasets with not enough training samples. The authors later addressed the effectiveness of the IML-"human-in-the-loop", which consists in model simulation with human interaction Holzinger et al. (2019). Specifically, the authors focus on a case study on Ant Colony Optimization along with the Traveling Salesman Problem.

In Berg et al. (2019), the authors present *ilastik*, which is an interactive tool that allows the analysis of medical imaging using iML. This tool allows for image segmentation, object classification tracking and counting, which are implemented through the use of workflows that the users can adapt according to the problem at hand.

iML has also been successfully applied in education: Kime et al. (2019) show how one can use this technique to estimate the student mastery of calculus skills using an online Problem-Solving Learning Environment. In Suh et al. (2019) the authors present the AnchorViz, which is a tool that facilitates the discovery of prediction errors and previously unseen concepts through human-driven semantic data exploration.

The authors of Khan et al. (2019) show how one can overcome some of the performance limitations of machine learning algorithms when dealing with biomedical images, namely problems such as interactivity, dependence of large datasets, and class imbalance. In particular, the authors address the advantages of turning to iML when performing biomedical image visualization (Direct Volume Rendering).

The authors in Visi & Tanaka (2020) present an overview of the use of iML techniques for analyzing and designing musical gestures. The authors address the needs and challenges that one can face when working with sound synthesis systems, and use iML techniques to human body gestures recognition and how existing ML algorithms can be used in different tasks like interaction with complex synthesis techniques and exploration of interaction possibilities.

More recently, Wu et al. (2021) propose a tool that combines iML and active learning to high-impact bug report prediction. The authors suggest this combination since the results of a supervised learning application may be inaccurate due to the need for a large number of labeled data.

The applications of interactive Machine Learning are thus numerous and diverse. In this paper we described one such application, in the domain of financial fraud detection. However, the focus is not on the accuracy of the system itself but rather on how a typical iML system can be improved by minimizing the need for retraining models as new data becomes available.

3 An interactive Machine Learning system for Fraud Detection

The main problem addressed in this paper is that of deciding *when to update a specific machine learning model* in an interactive learning scenario. The problem arose during the implementation of an iML system for Fraud Detection, which is briefly described in this section (Figure 1).

The data used in this system is extracted from SAF-T (PT) files (Standard Audit Files for Tax)[1] from an organization. This file is an XML (eXtensible Markup Language)[2] file and contains information about national organizations' billing and accounting data that must be shared regularly with the Portuguese Tax Authority. Asides from this obligation, the data is also often used to perform internal or external audits, especially in larger organizations.

In a first phase, the described data is transformed using a feature extraction process, which results in the creation of the main features that are relevant for auditing. Next, a Rule-based system is used to enrich the dataset by adding a group of additional features, based on audit guidelines and other norms. The resulting data is designated as *unlabeled dataset*.

After this feature extraction process, the interactive phase takes place: the auditor initiates a new audit project by selecting specific instances from the previously defined dataset, and then proceeds to do the audit while providing structured and semi-structured feedback. This feedback can take can be diverse but includes changes in the values of certain variables or natural or semi-structured text used to justify certain decisions or audit rationale.

The instances reviewed by the auditors are then added to the *labeled dataset*, which consists of the input for the training models that can predict the likelihood of fraud of a certain instance. Additionally, when performing the analysis of the unlabeled dataset, the auditor can also suggest new features or changes in existing features. This allows auditors to add higher-level and more abstract features, that often cannot be derived from the raw data through automated processes, but that have meaning for an auditor. These features then go through a curation process and, if approved, are added to the pipeline.

Since these user-defined features cannot be calculated directly from the raw data, machine learning models are trained to predict them, so that they can be automatically filled in the unlabeled dataset, so that the role of auditors gradually becomes one of validating data rather than of providing feedback. Thus, a pool of models exists in the system. One is considered the *main* model: the model that predicts the likeliness of fraud. Additionally, one model is maintained for each user-defined variable.

As the labeled dataset changes over time, with the contribution of the feedback provided by the auditors, the models eventually become outdated, which decreases the accuracy of the system. The simplest solution would be to regularly

[1] https://info.portaldasfinancas.gov.pt/pt/apoio_contribuinte/Pages/news-saf-t-pt-vers-inglesa.aspx

[2] https://www.w3.org/XML/

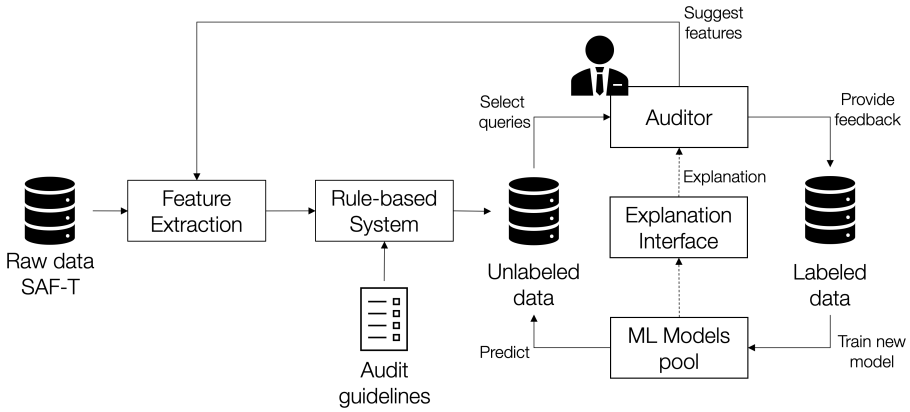


Fig. 1: Overview of the interactive learning system for fraud detection.

re-train the models. However, these model retrainings, which consume considerable amounts of computational resources and time, do not always result in significantly better models as new data does not necessarily mean new patterns. Knowing if and when to re-train each of the models may significantly reduce the resources used to maintain the models up to date. The process through which this is implemented is described in the following section.

4 Predicting Model Performance

In interactive machine learning scenarios, as described before, the user can interact with the system and contribute with feedback in the form of new data or changes to existing data. One of the main consequences of using this type of system is that the original dataset changes over time, not only due to the users' feedback but also due to new unlabeled data arriving. Therefore, a key decision in such systems is to determine the most appropriate time to retrain the model or update parts of it.

There are two possible scenarios, which represent a trade-off between resources consumption and accuracy: a system with more frequent updates and a system with less frequent updates. A system with smaller intervals between model updates will be a more dynamic system and adapt quicker to data changes. But it can turn out to be more sensitive to noise and use more computational and time resources for training the models. On the other hand, a system with large intervals between models (or less frequent updates) is a less dynamic system with slower responses to data changes. But a system with less frequent updates has the advantage of using less computational and time resources, and also of being less sensitive to noise in the data. A statically defined time interval for updating models is also not a viable solution as data and patters arrive or change at irregular intervals, thus making it hard if not impossible to find an adequate

interval. Hence the proposed solution of predicting models' performance metrics before training each model in order to determine if it is worth to update them.

Specifically, we aim at predicting the error metrics of each model if trained on the most recent data. With this approach, the model is effectively only trained if the statistical properties of the new data point to a better performance of the model. Thus, the computational (and time) resources are only put in use when required to perform the retraining/update of the model with a significantly lower error.

To do this, we propose an approach based on meta-learning. Specifically, we use a meta-model to predict the expected error of training a specific model on a dataset with given statistical properties. The input of the meta-model is the set of features that describe the statistical properties of the datasets, which we call *meta-features*. The meta-features are in turn stored in the so-called *meta-dataset*: one row for each dataset (or version of a dataset) for which a model was trained. The meta-features, which are the independent variables of the meta-model, consist of the characteristics of the dataset (number of rows and columns, and respective ratio), statistical properties of the data (mean kurtosis and mean skewness), information-theoretic (class entropy, mean entropy of variables, noise signal ration), among many others. The meta-dataset also stores the dependent variables, which consist of the performance metrics of each model trained with the respective dataset, including the rmse, mae, F1-score or training time.

Figure 2 contains more information of the components of the system involved in the prediction of the performance metrics. When the meta-dataset contains enough information, the meta-model can be trained. First, the system checks if there is any trained meta-model. If it does not exist, the first version of the meta-model is trained with the available meta-data and it is added to the model pool.

On the other hand, if there is already a meta-model, the system uses the outcome of the meta-model, which is the expected error metric of a given model when trained on a dataset, to decide if a model ought to be trained at a specific moment. If the new model is not expected to have better performance metrics than the previous model, then it is not worth it to re-train the model. If multiple error metrics are required, one meta-model is trained for each error metric, all based on the same meta-dataset but using the respective dependent variable. These meta-models are also updated at regular intervals, as new meta-data is available from the retraining of the models in the pool.

So, based on the outcome of the meta-model, the system decides if it worth it to train a new model. Every time a new model is trained, the information about the meta-features is added to the meta-dataset. With this, the capability of the system to perform predictions on the quality of future models can be improved over time.

The validation and results of this approach are discussed in the following section.

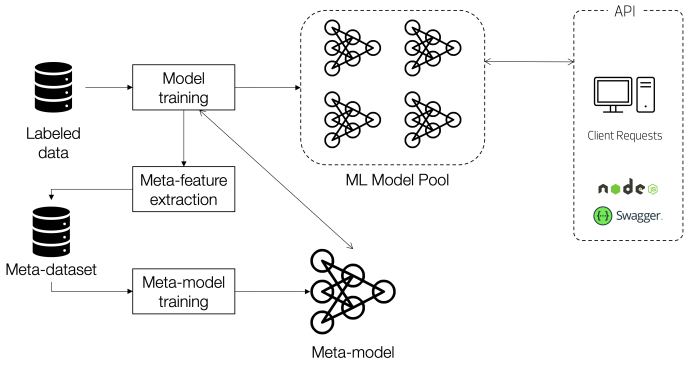


Fig. 2: Detail of the components in the system involved in the training and use of the meta-model.

5 Methodology and Results

This section details the methodology through which the proposed approach was validated and the corresponding results. A twofold approach was followed. On the one hand, we use a proprietary fraud detection dataset to create a first meta-model M_1 that is specific to this domain. On the other hand, we use a collection of 50 publicly available datasets to create a second, more general meta-model M_2 that can, in principle, be used in any domain. We assess and compare the performance of both meta-models at predicting the performance of new models trained for predicting the level of fraud for this particular problem.

5.1 Data

Two sets of data are used in this experiment. The first is composed of a proprietary fraud detection dataset consisting of more than 20.000 instances and 28 attributes. Of the 28 attributes, 4 are enumerations proposed by human auditors and represent high-level/abstract concepts that cannot be directly extracted from the SAF-T files but have meaning in the domain. These are deemed f1 to f4. Another attribute is the main dependent variable and it represents the likeliness of an instance being fraud, from the auditors' perspective. This is represented as a number between 0 and 9. These 5 attributes are contributed by the auditors. The remaining 23 attributes are either numeric or enumerations and are automatically obtained from the SAF-T files through the feature extraction process described before.

The second set of data is composed of 53 datasets, covering both regression and classification problems, as well as streaming and batch scenarios. Table 1 describes some of them. Of these, 50 datasets were used to train the meta-model whereas the last 3 depicted in the Table were used for testing it.

Table 1: Characterization of some of the 53 datasets used for building M_2: the first 50 were used as training sets (excerpt) while the last 3 were used for testing the meta-model. (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

5.2 Creation of the Meta-models

The process followed to create both meta-models, albeit with some particularities, can be generalized to what is represented in Figure 3. The training datasets (or parts of it, as described below) are first processed to standardize them. Next, the meta-features are extracted. At the same time, a model is trained with the dataset, and its resulting performance metrics are also extracted. These performance metrics (which are dependent variables for the meta-model) are added, together with the meta-features (the independent variables), as a new instance to the meta-dataset. At regular intervals, a new meta-model is trained. The performance of the meta-model is evaluated using 10-fold cross-validation. Once the new version of the meta-model is available, it can be used to make predictions about the performance of future models. All models and meta-models were trained using a Random Forest algorithm, with 20 trees, each with a maximum depth of 20 levels. While better algorithms/configurations may exist, we did not yet focused on optimizing meta-model performance: so far, we rather focused on the validation of the proposed approach, and will improve meta-model performance in future work.

Meta-datasets are 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 meta-models, one to predict each of these individual metrics. However, in this paper we focus on the prediction of the rmse.

Two meta-models were used in these experiments. The first, deemed M_1, was built solely with data from the proprietary fraud detection dataset. The second,

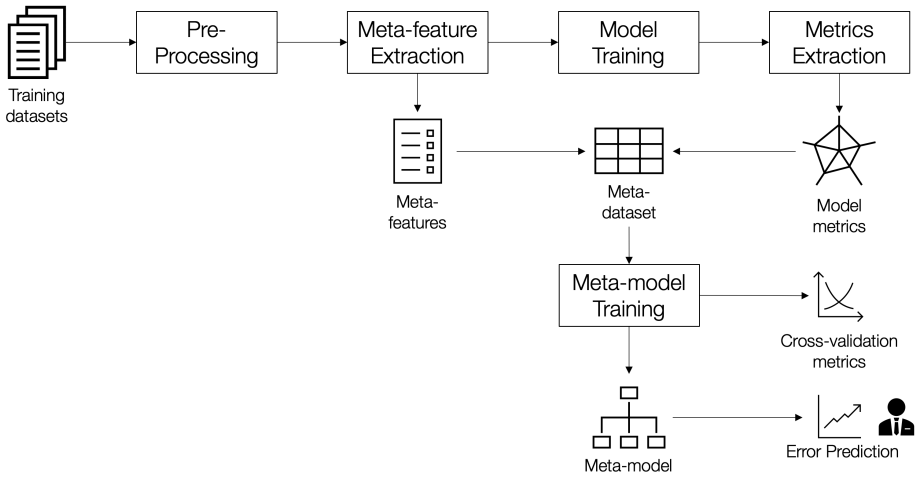


Fig. 3: Methodology followed to create the meta-models.

deemed M_2, is a more general meta-model that the team has been improving over time, based on 50 datasets, as described in the previous section. There are two main reasons for this twofold approach: 1) we want to determine if a specific and domain-dependent meta-model built on-the-fly, as new data arrive, is good enough to predict the performance of new models; and 2) we want to determine if a general meta-model, build with data from many different problems, can be used for a specific domain and eventually outperform the specific meta-model. If the second case holds true, it may open the door to the use of a general meta-model that can be used successfully in multiple domains, without the need to train specific meta-models, thus avoiding the initial warm-up phase in which data must be collected before predictions of the meta-model can be performed.

M_1 was created solely with data from the proprietary dataset. The goal is to assess how the proposed approach would perform in a real scenario, in which the meta-model is trained only with the data of the domain, as it arrives. To this end, a new instance was added to the meta-dataset at each 200 new instances of labeled data. This also means that when the labeled data starts to arrive, there is yet no meta-model that can be used to make predictions. For this reason, the first version of the meta-model M_1 was only trained after 2000 labeled instances were available. Since a new instance is added to the meta-dataset at each 200 instances, the first version of M_1 was trained with 10 instances of meta-data. From this point on, M_1 can start to be used to predict future models performance. Moreover, a new meta-instance continues to be added at a 200-instances interval (and the meta-model is updated), so that the meta-model continues to improve over time.

Concerning M_2, the goal is that we can have a general meta-model, built based on many different datasets, that can virtually be applied to any specific Machine Learning problem and still make good predictions regarding the per-

formance of its models. The main challenge is thus to have a sufficient number of diverse datasets, so that the meta-model knows as much different problems as possible. Diversity is related to the statistical properties of the datasets: datasets that, albeit from different domains, have similar meta-features (statistical properties) will not add value to the meta-dataset.

For the sake of diversity, the list of datasets used to build M_2 contains both regression and binomial classification problems. In order to include them in a single meta-dataset, all binomial classification problems were converted into regression problems by replacing the binary labels with the values 0 and 1. We also acknowledge that, if only one meta-instance were added to the meta-dataset for each dataset, we would have a rather small meta-dataset (with only 50 meta-instances). Thus, in order to increase its size, each dataset was also streamed in blocks of 200 instances. For each block, a model was trained, and the corresponding meta instance added to the meta-dataset. For example, for a dataset with 650 instances, 4 models are trained: the first with 200 instances, 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. Following this approach, the meta-dataset that was used to train M_2 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.

Table 2 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 M_2.

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 M_2 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, a 10-fold cross-validation approach was followed, as depicted in Figure 3. 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 $mae = 0.007$.

5.3 Results

This section details the results achieved when using both meta-models M_1 and M_2 to predict the rmse of the fraud detection model over time. The experiments described in this section were performed on a system with 2.7 GHz quad-core processor and 16GB 2133 MHz RAM.

To evaluate both meta-models, the following methodology was implemented. The fraud detection dataset was streamed in blocks of 200 instances, in the order in which the data was labeled by the auditors, to simulate a real use case scenario. For each block, a new fraud detection model was trained. At the same time, each meta-model M_1 and M_2 was used to predict the rmse of each model trained, and the predicted value of the rmse was compared against that of the observed rmse after training the model.

The only difference is that, as detailed in the previous section, the M_1 meta-model was only available after the first 10 blocks of data. This happens because M_1 is trained with the data, as it arrives, so there is the need for a setup period in which a minimum amount of meta-data is necessary. This does not happen with M_2, which was previously trained using data from 50 datasets, as previously described. This means that for M_1 101 models were trained, while for M_2 this number amounts to 111.

Moreover, for each model trained, we analyze whether using the prediction of M_1 and/or M_2 to avoid unnecessarily training a model would have been a good decision. To this end the following rationale was used: if the rmse predicted by a meta-model changes by at least a given amount, the model is retrained, otherwise we abstain from doing so. Nonetheless, for validation purposes we still train every model, to ascertain the accuracy of the meta-models.

Specifically, we ran 4 experiments: one for each meta-model and for each of two thresholds (5% and 10%), deemed A to D. That is, in experiment A and B the meta-model M_1 is used, but in A the fraud prediction model is re-trained if M_1 predicts a change of 5% or more in the rmse, while in B the model is retrained only if M_1 predicts a change of 10% or more. Experiments C and D follow the same rationale, but use M_2 as the meta-model instead.

The performance of each experiment is measured through several indicators: the differences between the predicted and observed rmse (also calculated through the rmse), the accuracy (measured as the percentage of times in which the meta-model predicted that the rmse would decrease by at least the corresponding threshold and it actually did), and the number of models whose training would have been avoided if the proposed approach was being used. The results of the four experiments are detailed in Table 3.

From Table 3 it is possible to conclude that M_2 behaves fairly better at estimating the rmse of each successive model that is trained: the rmse is significantly

Experiment	RMSE	Accuracy	#Avoided models
A	0.122	93.07% (94/101)	79.21% (80/101)
B	0.122	94.06% (95/101)	85.15% (86/101)
C	0.001	99.10% (110/111)	71.17% (79/111)
D	0.001	100% (111/111)	83.78% (93/111)

Table 3: Summary of the performance indicators of each experiment performed.

lower while the accuracy is higher and the number of models whose training was avoided is more or less in line with that of M_1.

Figure 4 shows the plots of the predicted vs. observed rmse for each of the experiments. Data points are color-coded according to the order in which the models were trained, from red to blue. That is, instances that correspond to earlier models are represented in red, and the later models are represented in blue. This shows, first, that the rmse generally tends to decrease as the system is provided with more labeled data. Moreover, it also shows that M_2 is far better at predicting the rmse of the models.

It should also be taken into consideration that one of the datasets used for building M_2 is the fraud detection dataset that was used to build M_1. However, while M_1 is based exclusively on this dataset, M_2 is enriched with 50 other datasets, which provide it with a significantly better ability to predict the performance metrics of the models.

It is also worth noting that experiments A and B are very similar in terms of results, as well as experiments C and D. This is because each of these pairs of experiments was run with the exact same data, hence the similar results. What changed is only the threshold for deciding when to re-train a model.

The differences in the experiments given the two different thresholds are visible in Figures 5 and 6. In Figure 5 (5a), only 21 models would have to be trained (instead of 101) if we were to update models only when the rmse is predicted to change by at least 5%. If a threshold of 10% is used, the number of models trained would have been only 15 (out of 101).

Figure 6 shows a similar analysis but in this case for meta-model M_2. A major difference is that M_2 can be used as soon as the first block of data is available, while M_1 requires a set of initial data (2000 instances) to train the first version of the meta-model, as described previously. This explains the differences in the X-axis.

The Figure shows that there is a very significant variability in the observed rmse in the first 2000 instances of data: the rmse starts very low, and then quickly increases, to start decreasing gradually. This can be explained with the initial small volume of data, that leads to such variations. Nonetheless, M_2 is able to predict the rmse with a high degree of accuracy, even in these conditions. In the case of experiment C, if the proposed approach was used (with meta-model M_1) only 32 models would have been trained (instead of 111), while in experiment D only 18 models would have been trained.

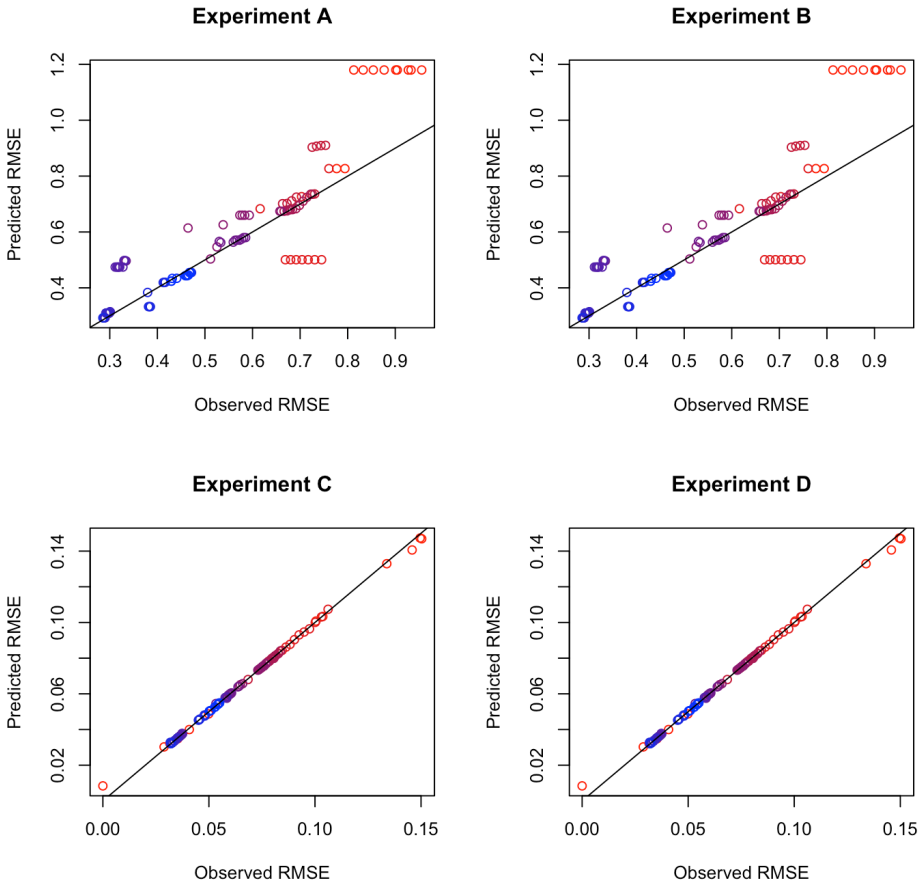
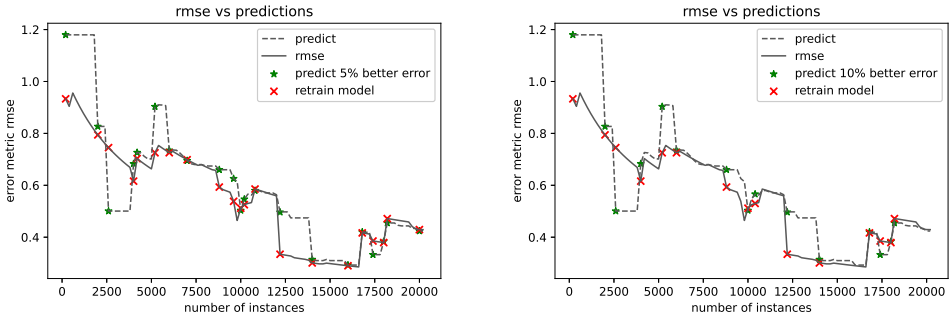


Fig. 4: Observed vs. predicted RMSE in each of the experiments.

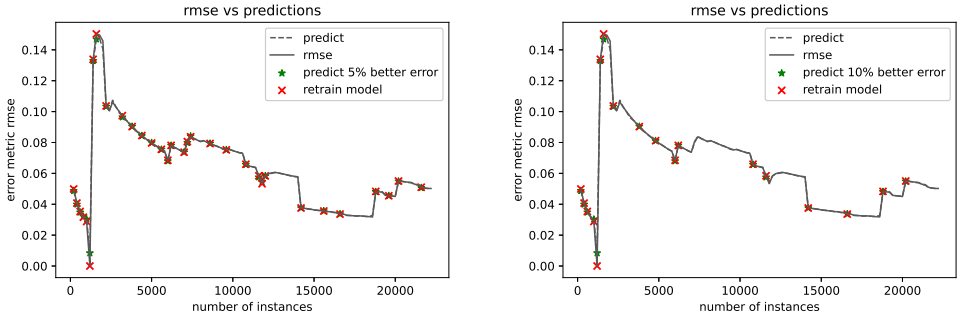
The more general meta-model M_2 was also validated on the 3 test datasets, whose details are provided at the end of Table 1. These datasets were not used to train the meta-model, so they can be used to assess the ability of the meta-model to generalize. Moreover, these are three of the well-known MOA datasets, a framework for data stream mining that also provides datasets with concept drift such as the three used in this work.

Table 4 summarizes the key performance indicators for each test dataset and threshold used (5% and 10%). The results show that M_2 is fairly good at guessing the rmse of models trained for each of the three datasets over time. Moreover, depending on the dataset and on the threshold, the decrease of number of models trained ranges between 63.27% and 97.96%, while the accuracy (the percentage of times that the meta-model correctly predicted a significant variation in the rmse) ranges between 81.63% and 100%.



(a) Experiment A (b) Experiment B

Fig. 5: Evolution of RMSE over time in Experiments A and B. The solid line represents the observed rmse while the dashed line represents the rmse predicted by M_1. Green stars show the points in which the meta-model predicted a significant decrease in the rmse (over the corresponding threshold), while red crosses represent the moments in which the models were trained.



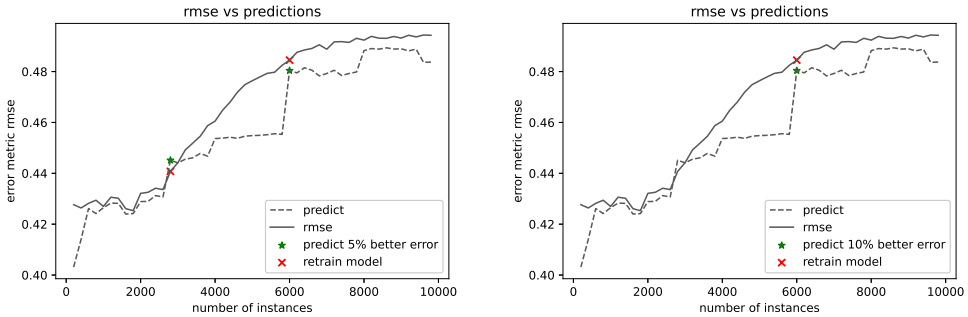
(a) Experiment C (b) Experiment D

Fig. 6: Evolution of rmse over time in Experiments C and D. The solid line represents the observed rmse while the dashed line represents the rmse predicted by M_2. Green stars show the points in which the meta-model predicted a significant decrease in the rmse (over the corresponding threshold), while red crosses represent the moments in which the models were trained.

Experiment	Threshold	RMSE	Accuracy	#Avoided models
Airlines	5%	0.012	100.00% (49/49)	95.92% (47/49)
Airlines	10%	0.012	97.96% (48/49)	97.96% (48/49)
AWS	5%	0.004	81.63% (40/49)	63.27% (31/49)
AWS	10%	0.004	95.92% (47/49)	87.76% (43/49)
Electricity	5%	0.005	100.00% (49/49)	95.92% (47/49)
Electricity	10%	0.005	100.00% (49/49)	95.92% (47/49)

Table 4: Summary of the performance indicators of each experiment performed for the test datasets.

Figures 7 to 9 provide additional details on the behavior of the meta-model over time. These Figures show that, in general, M_2 is able to predict the rmse of the models trained over time in a satisfactory way, with the exception of the beginning of the AWS experiment, in which the initial error is rather large, but quickly decreases.

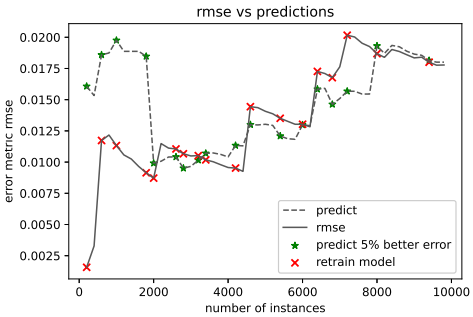


(a) M_2 used on the Airlines dataset, with a threshold of 5%.

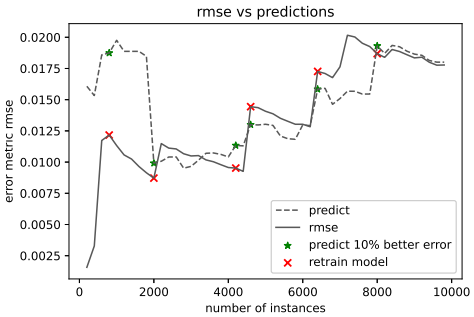
(b) M_2 used on the Airlines dataset, with a threshold of 5%.

Fig. 7: Evolution of rmse over time in the Airlines experiment. The solid line represents the observed rmse while the dashed line represents the rmse predicted by M_2. Green stars show the points in which the meta-model predicted a significant decrease in the rmse (over the corresponding threshold), while red crosses represent the moments in which the models were trained.

Figure 10 shows the plots of the predicted vs. observed rmse for each of the three experiments with the test datasets. Data points are color-coded according to the order in which the models were trained, from red to blue. That is, instances that correspond to earlier models are represented in red, and the later models are represented in blue. This shows, first, that the rmse generally tends to decrease as the system is provided with more labeled data. Moreover, it also shows how M_2 has the largest error in the first models trained for the AWS dataset, but

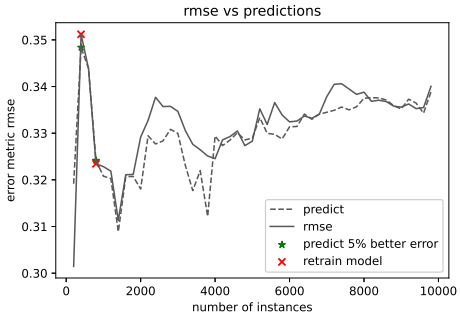


(a) M_2 used on the AWS dataset, with a threshold of 5%.

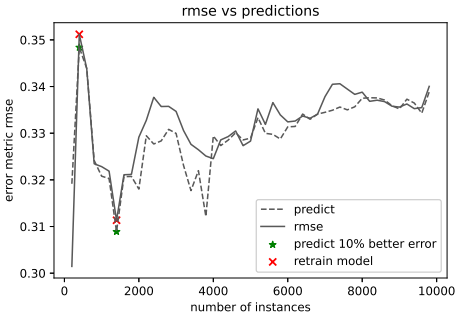


(b) M_2 used on the AWS dataset, with a threshold of 5%.

Fig. 8: Evolution of rmse over time in the AWS experiment. The solid line represents the observed rmse while the dashed line represents the rmse predicted by M_2. Green stars show the points in which the meta-model predicted a significant decrease in the rmse (over the corresponding threshold), while red crosses represent the moments in which the models were trained.



(a) M_2 used on the Electricity dataset, with a threshold of 5%.



(b) M_2 used on the Electricity dataset, with a threshold of 5%.

Fig. 9: Evolution of rmse over time in the Electricity experiment. The solid line represents the observed rmse while the dashed line represents the rmse predicted by M_2. Green stars show the points in which the meta-model predicted a significant decrease in the rmse (over the corresponding threshold), while red crosses represent the moments in which the models were trained.

behaves generally well other than that, and especially well in the Electricity dataset in which the error is minimum.

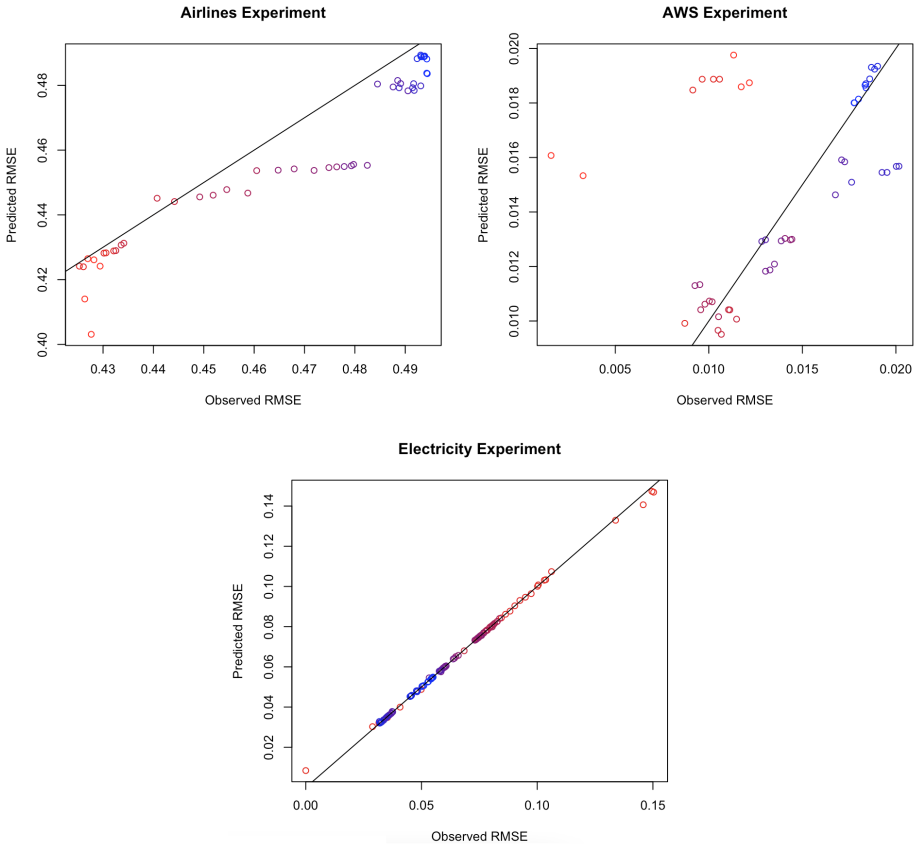


Fig. 10: Observed vs. predicted RMSE in each of the experiments with the test datasets.

While there are still limitations to the proposed approach, the results are satisfactory and encouraging. Especially because M_2, the more general meta-model, behaves better than M_1, the specific meta-model. This points out that it might be possible to develop a meta-model that is robust enough to predict the performance of any Machine Learning problem, as shown by the results of the meta-model when used in the three test datasets.

6 Conclusions, Limitations and Future Work

In this paper, we proposed a solution to a problem commonly found in interactive machine learning scenarios: to know whether or not is worth it to update a model at a given time. The proposed solution predicts the performance metrics of a model to be trained on data with certain statistical properties, represented as meta-features of the original dataset. The main goal of this approach is to minimize the necessary amount of computational resources and time in such systems, and make a more efficient management of model pools.

To validate the solution, several experiments were run. Some aimed at validating the approach for the specific use case of fraud detection, and others validated the approach for 3 publicly available streaming datasets. Moreover, two different meta-models were evaluated. The former, M_1, was build on-the-fly in the fraud detection use case, as new data arrived. The goal was to determine if it is possible to kickstart such a process from the ground up, with no initial meta-model, and start building the meta-model on-the-fly, only with the data of a specific problem. The latter, M_2, was trained previously, using data from 50 publicly available datasets. It was then tested on the fraud detection dataset and on 3 test datasets. The goal was to determine if such a meta-model would be able to generalize to any problem, and eventually behave better than a model trained for a specific problem.

Results show that M_2 behaves generally better when compared with M_1. This suggests that it might be possible to have a general meta-model, that is able to predict performance metrics for any Machine Learning problem. This was shown not only for the fraud detection dataset, in which M_2 outperformed M_1, but also in the three test datasets, in which M_2 showed that it can accurately predict the evolution of the rmse over time, despite not knowing such datasets in the training phase. Moreover, M_2 maintains or improves its accuracy over time even when testing on streaming datasets with concept drift, as was the case.

In terms of avoiding the retraining of models, M_2 shows that it can avoid between 60% to 90% of model retrainings, depending on the ML problem, when models are updated at 200-instances intervals. In scenarios of interactive machine learning in which multiple ML models are maintained in parallel, this may result in significant savings in computational resources and training time.

It should also be taken into consideration that one of the datasets used for building M_2 is the fraud detection dataset, that was the only one used to build M_1. However, while M_1 is based exclusively on this dataset, M_2 is enriched with 50 other datasets, which provide it with a significantly better ability to predict the performance metrics of the models. That is, the input of other ML problems contributes to improve the accuracy of the meta-model for a specific problem.

In future work we will evaluate the proposed approach regarding the remaining performance metrics, and also in what concerns training time. The goal is to determine if the meta-models are able to predict the remaining performance metrics with the same accuracy as rmse, including training time. This will result in a much more thorough solution, that will enable far robust policies for updating models in interactive learning scenarios. Based on these results, we will

also propose a better policy for updating models than that used in this paper, which is only based on the predicted change of a specific error metric. Finally, we will also increase the meta-dataset with data from additional ML problems, with the goal of making it as generalizable as possible, and also test it in more diverse ML problems.

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 Tecnologia within projects UIDB/04728/2020 and UIDB/00319/2020.

Bibliography

Aggarwal, C. C., Kong, X., Gu, Q., Han, J. & Philip, S. Y. (2014), Active learning: A survey, *in* 'Data Classification: Algorithms and Applications', CRC Press, pp. 571–605.

Berg, S., Kutra, D., Kroeger, T., Straehle, C. N., Kausler, B. X., Haubold, C., Schiegg, M., Ales, J., Beier, T., Rudy, M. et al. (2019), 'Ilastik: interactive machine learning for (bio) image analysis', *Nature Methods* pp. 1–7.

Carneiro, D., Guimarães, M., Carvalho, M. & Novais, P. (2021), Optimizing model training in interactive learning scenarios., *in* 'WorldCIST (1)', Springer, pp. 156–165.

Fails, J. A. & Olsen Jr, D. R. (2003), Interactive machine learning, *in* 'Proceedings of the 8th international conference on Intelligent user interfaces', pp. 39–45.

Holzinger, A. (2016), 'Interactive machine learning for health informatics: when do we need the human-in-the-loop?', *Brain Informatics* **3**(2), 119–131.

Holzinger, A. & Jurisica, I. (2014), Knowledge discovery and data mining in biomedical informatics: The future is in integrative, interactive machine learning solutions, *in* 'Interactive knowledge discovery and data mining in biomedical informatics', Springer, pp. 1–18.

Holzinger, A., Plass, M., Kickmeier-Rust, M., Holzinger, K., Crişan, G. C., Pintea, C.-M. & Palade, V. (2019), 'Interactive machine learning: experimental evidence for the human in the algorithmic loop', *Applied Intelligence* **49**(7), 2401–2414.

Jiang, L., Liu, S. & Chen, C. (2019), 'Recent research advances on interactive machine learning', *Journal of Visualization* **22**(2), 401–417.

Khan, N. M., Abraham, N., Hon, M. & Guan, L. (2019), Machine learning on biomedical images: Interactive learning, transfer learning, class imbalance, and beyond, *in* '2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)', IEEE, pp. 85–90.

Kime, K., Hickey, T. & Torrey, R. (2019), Refining skill classification with interactive machine learning, *in* '2019 IEEE Frontiers in Education Conference (FIE)', IEEE, pp. 1–8.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J. & Woźniak, M. (2017), 'Ensemble learning for data stream analysis: A survey', *Information Fusion* **37**, 132–156.

Ramos, D., Carneiro, D. & Novais, P. (2019), evorf: An evolutionary approach to random forests, *in* 'International Symposium on Intelligent and Distributed Computing', Springer, pp. 102–107.

Ramos, D., Carneiro, D. & Novais, P. (2020), 'Using a genetic algorithm to optimize a stacking ensemble in data streaming scenarios', *AI Communications* (Preprint), 1–14.

Suh, J., Ghorashi, S., Ramos, G., Chen, N.-C., Drucker, S., Verwey, J. & Simard, P. (2019), 'Anchorviz: facilitating semantic data exploration and concept dis-

covery for interactive machine learning', *ACM Transactions on Interactive Intelligent Systems (TiiS)* **10**(1), 1–38.

Visi, F. G. & Tanaka, A. (2020), 'Interactive machine learning of musical gesture', *arXiv preprint arXiv:2011.13487* .

Widmer, G. & Kubat, M. (1996), 'Learning in the presence of concept drift and hidden contexts', *Machine learning* **23**(1), 69–101.

Wu, X., Zheng, W., Chen, X., Zhao, Y., Yu, T. & Mu, D. (2021), 'Improving high-impact bug report prediction with combination of interactive machine learning and active learning', *Information and Software Technology* **133**, 106530.