Algorithm Recommendation for Data Streams

Jáder M. C. de Sá
Department of Computer Science
University of Brasilia
Brasília, Brazil
jader.martins@ipea.gov.br

Andre L. D. Rossi
São Paulo State University (UNESP)
Itapeva, SP, Brazil
andre.rossi@unesp.br

Gustavo E. A. P. A. Batista
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
g.batista@unsw.edu.au

Luís P. F. Garcia
Department of Computer Science
University of Brasilia
Brasília, Brazil
luis.garcia@unb.br

***Abstract*—In the last decades, many companies have taken advantage of knowledge discovery to identify valuable information in massive volumes of data generated at high frequency. Machine learning techniques can be employed for knowledge discovery since they can extract patterns from data and induce models to predict future events. However, dynamic and evolving environments usually generate non-stationary data streams. Hence, models trained in these scenarios may perish over time due to seasonality or concept drift. Periodic retraining can help, but a fixed hypothesis space may no longer be appropriate. An alternative solution is to use meta-learning for regular algorithm selection in time-changing environments, choosing the bias that best suits the current data. In this paper, we present an enhanced framework for data stream algorithm selection based on MetaStream. Our approach uses meta-learning and incremental learning to actively select the best algorithm for the current concept in a time-changing environment. Different from previous work, we use a rich set of state-of-the-art meta-features, and an incremental learning approach in the meta-level based on LightGBM. The results show that this new strategy can improve the recommendation accuracy of the best algorithm in time-changing data.**

I. Introduction

Clicks in web pages, user interaction in streaming services and IoT devices typically generate data at a high frequency. In the era of Big Data [1], [2], many companies are taking advantage of massive databases to aggregate valuable information to their users, such as movie recommendation and content filtering in social media. Machine Learning (ML) is one of the research areas that stands out for knowledge discovery. ML extracts patterns from data through model induction, enabling the prediction of future events [3].

As every inductive method, traditional ML techniques have two fundamental assumptions: (i) the future must resemble the past and (ii) the future events must be independent of past events [4]. However, in continuous data flow (also known as a data stream), the underlying data distribution may change over time. Thus, these assumptions could not stand, and the accuracy of induced models could diminish as time passes, causing poor experiences to users of those systems [5], [6].

Most of the existing data stream approaches rely on a single model or an ensemble of models with the same base classifier. These approaches have a fundamental assumption that a unique hypothesis space is best for all the concepts. However, we have more predictive power by learning different concepts with different learning paradigms [7]. A central question is how to efficiently choose the most appropriate learning algorithm for a given data sample.

Meta-Learning (MtL) is a prominent technique to learn with changing distributions by detecting concept drifts and recommending algorithms to improve prediction [8], [9], [10]. MtL requires meta-data [11] in the form of a set of descriptive characteristics, named meta-features, of the problem at hand [12]. These meta-features are extracted and combined into a meta-example. The central idea of MtL is to apply different ML algorithms to the meta-data and have their performances recorded as labels of the meta-examples. This procedure is performed for a data stream at different time points. Finally, we use such labelled meta-dataset to induce a meta-classifier that predicts which is the most appropriate algorithm for future events. MtL avoids the use of expensive model selection processes, common in the ML literature [13].

In this paper, we enhance MetaStream [14], [7], an MtL framework based on periodic algorithm selection on data streams. Our contributions are two-fold: (i) We augment the meta-feature set with a broader set of state-of-the-art features [12], and, (ii) We include an incremental learner in the MtL level using LightGBM [15] as meta-classifier. Therefore, this investigation answers whether the use of state-of-the-art meta-features allied to incremental learning provided by LightGBM can predict more accurately than a baseline method and improve the general performance of the learning system.

This paper is organized as follows. Section II discusses the relevant literature, including MetaStream and the enhancements proposed in this paper for this framework. Section III presents the hyperparameters, meta-features and datasets used to evaluate the proposed methodology. Section IV shows the experimental measures and analyses the evaluation results. Finally, Section V concludes our work and present directions future research.

II. Related Work

This section presents the background information necessary to describe the proposed approach: Section II-A explains about data streams problems and how to deal with concept drifts and retraining. Section II-B presents the MtL framework, including the process of building meta-data and how to recommend

algorithms. Section II-C describes the MetaStream, a recommendation system based on MtL for data streams.

A. Data Streams

Most ML techniques assume data are independent and identically distributed (*i.i.d.*). However, this assumption usually does not hold in data stream environments [16]. Data in the form of a stream arrive sequentially and can present temporal dependency [17] and concept drifts. As data streams are potentially infinite, they impose restrictions on computational resources, such as memory and CPU. For instance, we cannot expect to store the entire stream in memory as in batch learning, and we need to process the events quickly to cope with fast-paced streams [18], [19]. These challenges have motivated an entire community to propose a diverse set of methods and strategies. The most prominent ones are forgetting mechanisms, statistical testing and adaptive algorithms.

Forgetting mechanisms make recent data more relevant than outdated data. A well-known example is sliding windows [20], a popular technique applicable to any ML technique. Statistical tests based systems act monitoring a predefined measure, such as models predictive performance, and alarms when this quality is below some tolerated threshold, demanding some actions. Examples of this approach are Page-Hinkley Test and Statistical Process Control [21]. Based on these two solutions, ADaptive WINdowing (ADWIN) [16] uses a statistical test to set each window size, adjusted to the "concept size". Although these methods can alarm a reduction in models' performance and adapt the sliding window, it is not possible to identify the algorithms or models that have become more suitable than the current one for the most recent data.

Another approach is the development of adaptive learning algorithms, such as VFDT [22]. VFDT was introduced as an online learning algorithm with low memory consumption, but not capable of adapting to concept drifts. Later, several enhancements addressed this issue, such as CVFDT (a direct improvement over VFDT) [23], HAT [24] and ARF [25], which can handle varying concepts by applying sliding windows in the training process. However, these methods have a fundamental assumption that a fixed hypothesis space (Decision Tree (DT) in the case of the mentioned approaches) is adequate for all concepts in the stream. In this paper, we can address this drawback by using change detectors based on MtL recommendation systems, targeting hypothesis spaces as learning tasks [7].

B. Meta-Learning

The algorithm selection problem was initially addressed by Rice (1976) [26] with the main goal to predict the best algorithm to solve a given problem when more than one algorithm is available. The components of this model are: the problem instances space (P), which is composed by datasets in MtL; the instance features space (F), which are the meta-features used to describe the datasets; the algorithms space (A), which contains the pool of ML algorithms for recommendation; and the evaluation measures space (M), responsible for assessing the performance of the ML algorithms in solving the problem instances contained in P. The MtL system implements an algorithm that maps a dataset $p \in P$, described by the meta-features $f \in F$, into one (or more) algorithm $\alpha \in A$ able to solve the problem with a good predictive performance according to $m \in M$, i.e., with maximum $m(\alpha(p))$.

Smith-Miles (2008) [27] improved this abstract model by proposing generalizations that can also be applied to the algorithm design problem. This proposal adds some components: the set of MtL algorithms; the generation of empirical rules or algorithm rankings; and, the examination of the empirical results, which may guide theoretical support to refine the algorithms.

One crucial component of the previous models is the definition of the set of meta-features (F) used to describe the general properties of datasets. These meta-features should provide evidence about the future performance of the algorithms in A [28], [29] and discriminate the performance of a group of algorithms with a low computational cost. We can divide the meta-features commonly used in the MtL literature into five groups: General, Statistical, Information-theoretic, Model-based and Landmarking.

General features are easily extracted from data [30], with low computational cost [29]. The statistical meta-features capture the main indicators of data localization and distribution, such as average, standard deviation, correlation, and kurtosis. Information-theoretic meta-features are based on information theory [31], usually entropy estimates [32], which capture the amount of information in (subsets of) a dataset [27]. Model-based meta-features are properties obtained from a model; usually, a DT [33], [34] induced from the data under analysis [30]. The Landmarking meta-features use the performance of simple and fast learning algorithms to characterize the datasets [27].

The definition of the set of problem instances (P) is another concern. The ideal scenario is to use a large number of diverse datasets to induce a reliable meta-model. To reduce the bias of this choice, datasets from several data repositories, such as UCI[1] [35] and OpenML[2] [36], can be used.

The algorithm space A represents a set of candidate algorithms to be recommended in the algorithm selection process. Ideally, these algorithms should also be sufficiently different from each other and represent all regions in the algorithm space [13]. Different measures can evaluate the models induced by the algorithms. For classification tasks, most of the MtL studies use accuracy. However, other indices, such as F_β, AUC and kappa coefficient, can also be used.

The next step is labelling each meta-example in the meta-data. Brazdil et al. (2009) [37] summarize the four main properties frequently used to label the meta-examples in MtL: (i) the algorithm that presented the best performance on the dataset (a classification task); (ii) the ranking of the algorithms according to their performance on the dataset (a ranking

[1] https://archive.ics.uci.edu/ml/index.php
[2] http://www.openml.org/

classification task); (iii) the performance value obtained by each evaluated algorithm on the dataset (a regression task) and; (iv) the model description, which is in general based on clustering and association rules.

In data stream environments, the problem instance space is composed of only one problem p. At the same time, the meta-features F are extracted from the data of each time window to form a meta-example. The meta-features extraction process should have a low computational cost and high information degree. Besides that, the performance of the algorithms A is assessed periodically, usually for each sliding window. Therefore, the meta-model replaces the current algorithm as soon as it predicts a different one is more suitable for the examples of the next sliding window [38], [39], [8].

In this paper, we present an MtL based method based on the MetaStream framework [7]. The basic idea is selecting the best algorithm for time-changing environments. MetaStream regularly induces a meta-classifier able to map the characteristics extracted from past and incoming data to the performance of classifiers on these data. Different from previous works, we use a set of state-of-the-art meta-features based on A. Rivolli et al. (2018) [12] and an incremental learning approach in the meta-level based on LightGBM [15].

C. Metastream

The literature presents alternatives to algorithm (or model) selection based on MtL for concept drift. One of the first to present it is Klinkenberg (2005) [40], where characteristics obtained from the learning process, such as sliding window sizes, were used to induce meta-models. Gama and Kosina (2014) [41] and Anderson et al. (2019) [8] investigated a different approach by reusing previously learned models but using the same algorithm to induce those models in the data stream. A third alternative is given by van Rijn et al. (2018) [42], where they use ensembles of models, having a high computational cost of inducing models for every algorithm and keep the weights updated.

Next, we present the MetaStream framework based on Rossi et al. (2014) [7], which has "offline" and "online" phases. The offline stage performs hyperparameter tuning, validation and training data generation with a small initial sample of data. The online phase acts in the dynamic environment recommending an algorithm for a given window of the data. Both stages are based on MtL recommendation systems.

1) Offline Phase: This phase starts after a given initial amount of training data has arrived. With this batch of data, Metastream induces the base-algorithms through k-fold cross-validation for hyperparameter tunning. With the same initial data, the algorithm continues by swiping a sliding window through the data, as shown in Figure 1. In this setting, Metastream induces base models using the base-level algorithms and extracts meta-features x^m for each window ω_{bi}.

Then, Metastream evaluates the induced models on the examples of the η_{bi} window. The best performing model becomes the label of meta-example (y^m). The same process repeats N steps further, where N is a specified number of

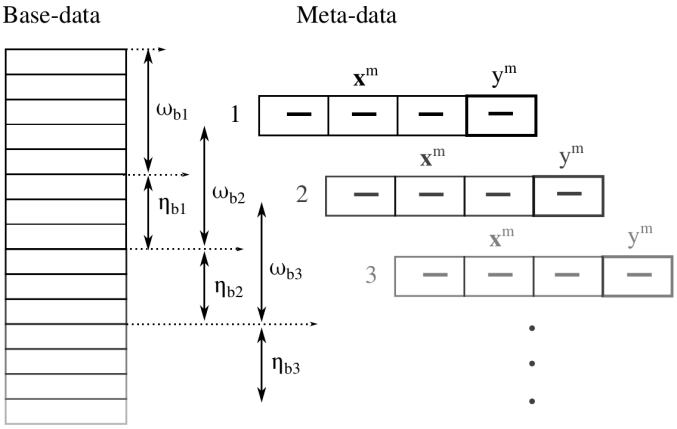

Fig. 1. Meta-feature extraction from ω_b and label obtaining from η_b windows.

instances to generate the initial meta-data, which Metastream uses to create the first meta-model.

2) Online Phase: In the online phase, the algorithm receives a continuous stream of data. At first, it gets a feature vector $\boldsymbol{x}_b = (x_1, ..., x_p)$, and with some delay, the target attribute $y_b \in \{1, 2, .., k\}$ for classification, where k is the number of classes.

Figure 2 shows the time t of the online phase. It has a window of fixed size ω_b that is used to induce the model and a window of fixed size η_b where the model induced on ω_b is evaluated. When Metastream processes all examples in η_b, it shifts the ω_b and η_b windows η_b instances to the right. Afterwards, Metastream induces a new model for this window.

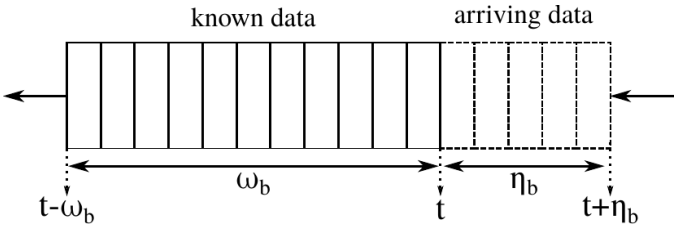

Fig. 2. Window discretization at base-level data stream.

MtL comes to play in the second level of processing, named meta-level. Figure 3 shows the meta-features extracted from ω_b windows. Metastream generates a meta-example without the target value, which is updated later.

The learning algorithm (meta-model) uses previously labelled meta-examples in the meta-base to induce a meta-classifier. The meta-classifier recommends an algorithm to induce a model using the ω_b window. Assuming the meta-model is accurate, it will likely suggest the fittest classifier to predict the labels of the η_b examples.

III. Methodology

As described in Section II-B, our problem of algorithm selection is given by choosing the most suitable hypothesis space according to a recent data window. Thus, in the presence

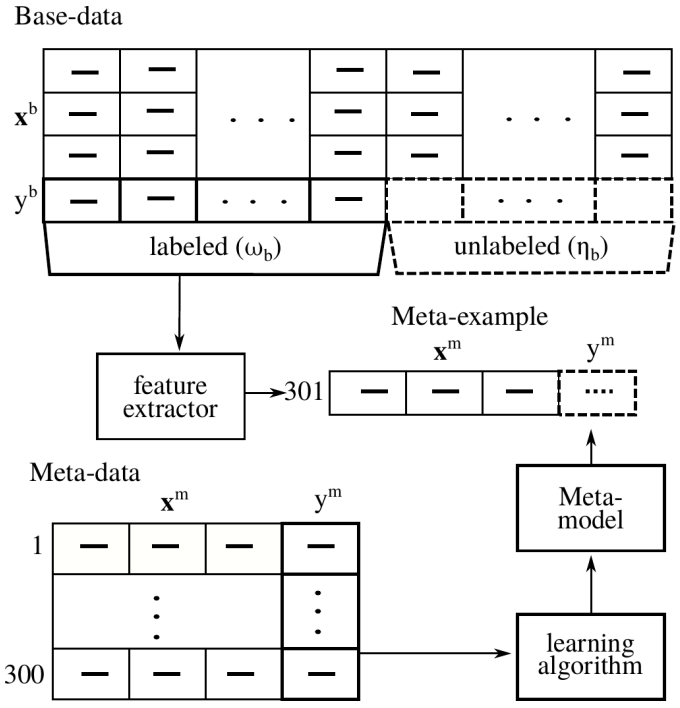

Fig. 3. Meta-feature extraction from ω_b and η_b windows at meta-level.

of concept drifts, we expect our proposal will recommend an alternative and more fit classifier to replace the current one.

Many theories have been developed to address the analysis of hypothesis spaces [4], [43], pointing directions to guide our choice. Another aspect to consider is the computational costs, memory and training/prediction time. This concern has led us to select Random Forest (RF) [44] and an efficient approximation of RBF-SVM through Nyströem method [45] as the base-level learning algorithms.

Therefore, our meta-dataset has labels according to the predictive performance of the models induced by RF and SVM for the examples in the η_b window. At the same time, the meta-classifier seeks to which one of these two classifiers is the most appropriate for the next η_b window.

As the meta-classifier, i.e., the recommender, we chose LightGBM for three main reasons. First, LightGBM presents state-of-the-art predictive performance for different problems [15]. Second, this algorithm automatically deals with missing features, which commonly occur for the extracted meta-features. Third, it has incremental learning capacity and low requirements for processing and memory[3], encouraging this choice for online applications. Incremental and batch learning has a significant distinction. While the first maintains trees and updates the values on the leaves, the second creates trees from scratch. Not only the training time for both differ, but also the incremental learning fixes the feature importance *apriori*.

For the meta-features, three groups from Section II-B were analyzed, namely statistical, model-based and landmarking. The statistical meta-features capture indicators of localization and distribution, the model-based extract properties of the DT models, while the landmarking use the performance of simple and fast learning algorithms to characterize the datasets [12]. In general, they have high discriminatory property and an average computational cost. For those experiments, we used the meta-features available in the Python Meta-Feature Extractor (pymfe) package [46] publicly available at GitHub[4].

We fixed the size of the windows across all experiments, in both base and meta-levels, $\omega_{m;b} = 300$ and $\eta_{m;b} = 10$. The training window step is the same as η. We perform a hyperparameter tuning in the base-level algorithms through k-fold cross-validation. We kept the hyperparameter values fixed for every future induction. Although this decision may not lead to optimal results, we believe that those hyperparameters values are a better choice than performing no selection at all (*i.e.*, using the default values). We use accuracy, Kappa score and Geometric Mean to assess the predictive performance of the induced models in both offline and online phases. , Kappa and Geometric Mean are more appropriate than accuracy to evaluate unbalanced class distributions, which is frequently the case of meta-data.

For the experiments, we selected three popular classification datasets in the data stream literature that present concept drift [39], [38], *Electricity* [47], *Power Supply*[48] and *Covertype* [49].

- *Electricity:* The Electricity dataset was collected from the Australian New South Wales Electricity Market. The prices fluctuate according to demand and supply, presenting seasonal factors. It contains 45,312 instances dated from 7 May 1996 to 5 December 1998 (set every 5 minutes). The target value is if the demand was higher or lower compared to the previous 24 hours and as predictive attributes the day of the week, the time stamp, the New South Wales electricity demand, the Victoria electricity demand, the scheduled electricity transfer between states.
- *Covertype:* This dataset contains the cartographic variables for 30 x 30-meter cells obtained from US Forest Service, it originally had 581,012 instances, but we reduced to 19,999 to minimize processing time. The target value is the cell's forest cover type (classes ranging from 1 to 7), and as predictive attributes, it has 53 properties of the cell including elevation, slope, soil type, etc.
- *PowerSupply:* Power Supply dataset contains records from 1995 to 1998, the task is to predict which hour (class) the current power supply belongs to (1 to 24), which is related to seasonal factors and trends, as predictive attributes we have the energy production in the main and auxiliary power grid. It has 29,928 instances, but we reduced to only 19,999.

The experimental analysis has an offline and online phase. The offline stages analyze the framework in the meta-level, that is, how well it can predict the best algorithm given the descriptive meta-features. The online phases analyze, in the base-level, the gains obtained by selecting the recommended

[3]https://lightgbm.readthedocs.io/en/latest/Experiments.html

[4]https://github.com/ealcobaca/pymfe

base algorithm, compared against a baseline method, namely Default, which represents the majority performance. For that, we look at the cumulative sum of performance gains over time, which is given by the difference in performance between using the recommended algorithm and using the Default one.

IV. Results

Similar to Section III, the results are divided into offline and online phases. In the offline evaluation, we analyze the meta-classifier to assess how well it discriminates classes at the meta-level. In the online assessment, we measure the gains regarding predictive performance at the base-level when employing the algorithms recommended by the meta-classifier.

A. Offline Analysis

Table I lists the distribution of algorithms (RF and SVM[5]) that presented the best performance for each batch of the data streams (*Electricity*, *CoverType* and *PowerSupply*). Under the perspective of MtL, this table shows the class distribution of the meta-examples for the offline meta-data.

TABLE I
ALGORITHM DISTRIBUTION IN META-DATA PER DATA STREAM PROBLEM.

	Electricity	CoverType	PowerSupply
SVM	0.752	0.718	0.535
RF	0.248	0.282	0.465

The algorithms RF and SVM presented distinct performances. For both *Electricity* and *CoverType* datasets, the class distribution is uneven, and, thus, the meta-classifier was trained to balance the loss for target proportion. On the other hand, the *PowerSupply* dataset is highly balanced.

We assessed the meta-classifier predictive performance using a time-series cross-validation [50] in the initial data. The model is induced using the first window of $\omega_m = 300$ train examples and predicts the label of the test window, composed of the $\eta_m = 10$ examples, then both windows slides by 1 example for the next evaluation. Table II presents the average and standard deviation for these metrics. Gains of meta-recommendation comes from non-trivially recommending the algorithms. Since the Default method represents a Kappa value equals to zero, $Kappa = 0$, $Kappa > 0$ means a prediction gain.

TABLE II
PREDICTIVE PERFORMANCE OF THE META-CLASSIFIER FOR THE OFFLINE PHASE.

	Kappa	G.Mean	Accuracy
Electricity	0.132±0.408	0.164±0.346	0.704±0.186
CoverType	0.322±0.517	0.330±0.470	0.752±0.177
PowerSupply	0.308±0.477	0.364±0.447	0.523±0.170

Table II presents the predictive performance in the offline phase assessed by three different measures using sliding window setting. The accuracy is estimated under $\eta_m = 10$, which is comparable to the meta-data distribution from Table I. For the Kappa and Geometric Mean measures, we removed the examples in η_m where there was no meaningful difference between the performance of the algorithms in the base level, i.e., we computed these measures for base examples where $|\text{Acc(RF)} - \text{Acc(SVM)}| > 0.1$. This approach was employed aiming to produce a more reliable estimate of the meta-classifier, i.e. when there is no clear distinction between models performance, any one of them could be chosen.

For all datasets, Kappa is at least slightly positive, showing that the recommender can select the correct algorithm when there is a patent difference between their predictive performance, outperforming the default method. Accuracy is close to dataset majority distribution, but later we argue that some examples present better gains to predict than others, as in base-level it has a significant difference.

To assess how the meta-features contribute to the recommendation problem, Figure 4 shows the 5 top-ranked meta-features, for each dataset, selected by LightGBM. In this figure, the x-axis represents the LightGBM feature importance in the meta-models, and the y-axis shows the top-ranked meta-features. The vast majority of meta-features selected to split nodes in the growing trees were from the landmarking group, followed by model-based and statistical meta-features. Those groups have the most informative and discriminative power meta-features according to the literature [12].

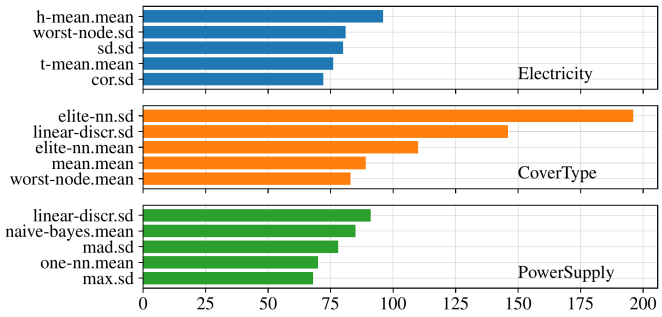

Fig. 4. Feature importance for the meta-algorithm in the offline data.

B. Online Analysis

In the online phase, we evaluate the meta-classifier (LightGBM) using a non-incremental and an incremental strategy. In the former evaluation, we train the meta-classifier using the procedure described in Section II-C2. When new meta-examples are available, the meta-classifier is retrained from scratch, and this procedure can lead to a model that is entirely independent of the previous models. In the incremental strategy, LightGBM updates the previous model, adapting the DTs to fit the meta-examples from the most recent window. Table III reports the results.

Although Table III presents relatively small Kappa values, we note that these values are positive for every combination of dataset-strategy. These results indicate that the meta-classifier can select the most appropriate algorithm compared to the majority class. This outcome is more evident for the

[5]An approximation for the SVM described in the methodology section.

TABLE III
PREDICTIVE PERFORMANCE OF THE META-CLASSIFIER FOR THE ONLINE PHASE.

Dataset	Strategy	Kappa	G.Mean	Accuracy
Electricity	Non-Incremental	0.013	0.326	0.670
	Incremental	0.023	0.510	0.645
CoverType	Non-Incremental	0.085	0.425	0.689
	Incremental	0.034	0.315	0.695
PowerSupply	Non-Incremental	0.067	0.556	0.565
	Incremental	0.006	0.466	0.524

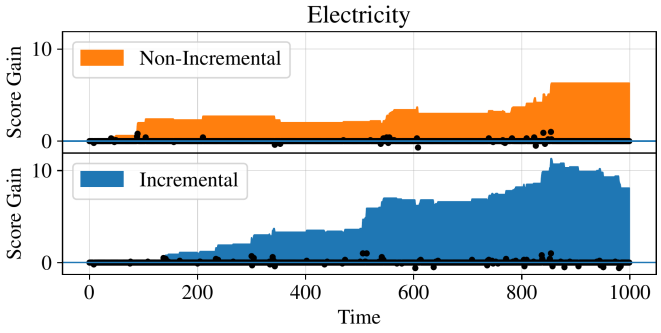

Fig. 5. Cumulative score gain over time for Electricity.

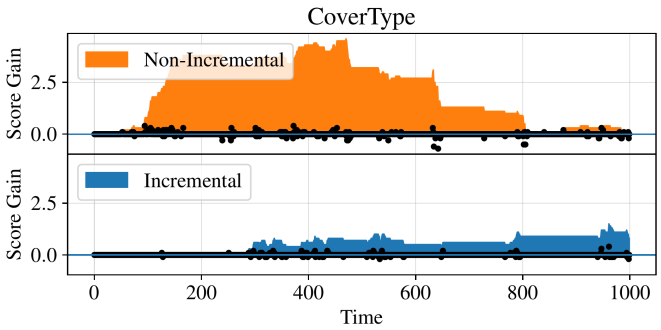

Fig. 6. Cumulative score gain over time for CoverType.

PowerSupply dataset, which presented the highest Geometric Mean and positive Kappa. This consistency reflects later in the cumulative gains of this dataset. The incremental strategy presented better performance than the non-incremental one for *PowerSupply* and *CoverType*, but for the *Electricity* it was worse for this metrics.

Figures 5-7 show the cumulative gain, which is the difference between the recommended algorithm accuracy and the Default method accuracy. The filled area is the cumulative sum of those score differences over time while the colours orange and blue represent incremental and non-incremental algorithms, respectively. Each black dot represents the score gain for time t.

In these figures, we observe a positive gain for the *Electricity* and *PowerSupply* datasets considering the recommendations of both strategies. However, the incremental strategy presents better performance for *Electricity* as opposed to *PowerSupply*, where the non-incremental is slightly better. In

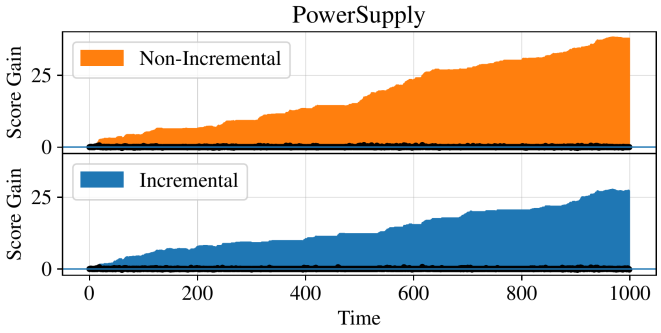

Fig. 7. Cumulative score gain over time for PowerSupply.

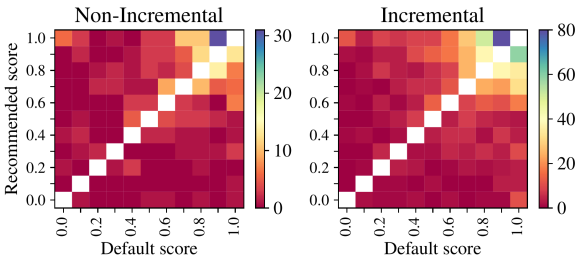

Fig. 8. Comparison between recommended and Default for Electricity.

the *CoverType* dataset we notice an interesting aspect. The non-incremental strategy presents a greater gain compared to the incremental one for lower points in time, but around $t = 450$, it starts to make incorrect recommendations. On the contrary, the incremental strategy is more constant, keeping gains slightly over time after $t-300$. This is partly determined by the difficulty of the dataset, mainly after $t - 400$, since the label that should be learned to infer the next window is not present in the training data. This is particularly difficult for the non-incremental strategy since the incremental one keeps the past learned target information.

In Figures 8, 9 and 10 the scores from Default are plotted against the scores from the recommended algorithms in a 2-dimensional histogram heatmap. Thus, each square in the heatmap represents the accuracy of the algorithm recommended by the Default method (x axis) and the meta-classifier (y axis) for the same window. The colour intensity as shown in the colour bar represents the number of points in that position. The points in the diagonal, where the recommended algorithm and the default algorithm had the same accuracy score, were removed and coloured as white to emphasize gains or losses of the framework. If the recommended classifier is always worse than the Default, i.e., recommending the minority class when it is not better than the majority, then all points will be below the white diagonal. Similarly, these points will be above if it is effectively selecting the algorithm from minority class when it is better than the majority.

In Figure 8, it is visible in the upper right region of the heatmap that the incremental algorithm is recommending the minority class more often than the incremental one, also the

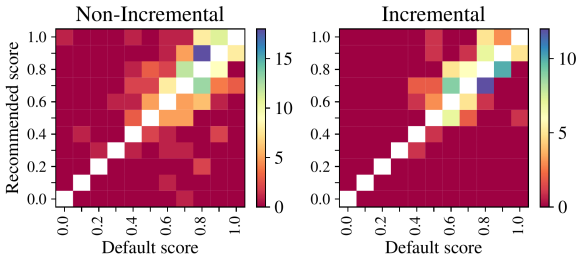

Fig. 9. Comparison between recommended and Default for CoverType.

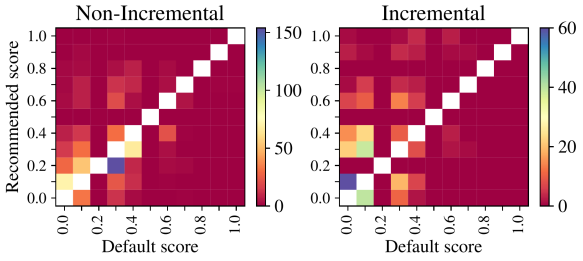

Fig. 10. Comparison between recommended and Default for PowerSupply.

blue square above the white diagonal has many more points (80 compared to 30 in the non-incremental), this reflects in gains as shown in Figure 5.

For the *Covertype* (Figure 9), we observe a different behaviour, there is a blue square above the diagonal for the non-incremental plot and a blue square below the diagonal in the incremental one. This points out that the incremental recommendation is performing worse than the default, as could be seen in Figure 6. The non-incremental also presents some light coloured squares below the diagonal, which resulted in reduced overall performance for later points in time.

The *PowerSupply* dataset, which has 24 classes, is the hardest one to classify, since most points are below 0.8 accuracy as shown in Figure 10. However, in the meta-level, was easier to discriminate algorithms as the meta-data is balanced, reflecting in a less symmetrical distribution for the diagonal (the upper region is more populated), it is also confirmed in Figure 7, with the highest cumulative score gain amongst the datasets.

The feature importance for the non-incremental strategy is given by Figure 4, as it keeps the same nodes obtained by training in the offline meta-data. The main difference between incremental and batch learning for this problem is that the feature importance is fixed in incremental learning. However, as shown in Figure 11, the importance varies for each trained window. It is believed that fixing feature importance acts as a regularization. Still, alternatively, concept drift could also occur in the meta-level, being this possibly beneficial to have a flexible tree choice.

V. Conclusion

In this paper, we presented a MtL method based on the MetaStream framework. We enhance MetaStream by extending the meta-features to modern and more informative ones, by including the incremental learning in the MtL level and by proposing LightGBM as meta-classifier. Although both strategies performed similarly, the incremental one had a significant lesser consumption of memory and processing time. The experimental results showed that the meta-classifier can consistently recommend the best algorithm for a given window in the data stream, leading to an increased gain of performance over time.

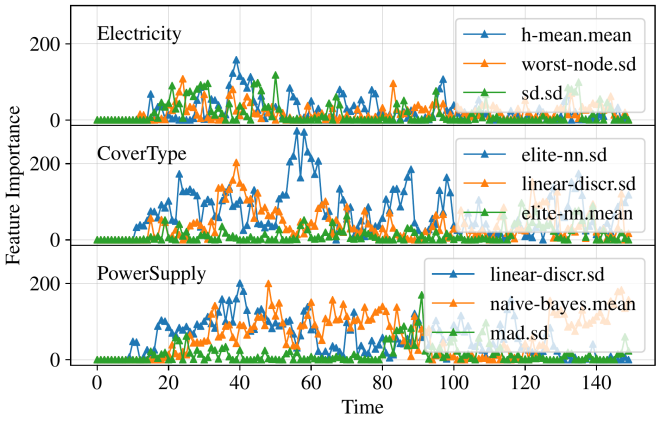

Fig. 11. Variation in feature importance over time for three meta-features.

As future work, we would like to rearrange the binary task to multi-class classification, enabling selecting between multiple hypothesis spaces in the base-level. Another idea is implementing a module for incremental meta-features, allowing to add more informative cutting edge meta-features. Also, as the original article proposed [7], time-series features could have significant discriminatory power. A recent approach is to perform recommendations based on regressing algorithms' accuracy which can be applied in this framework extending this study to more datasets.

Acknowledgements

The fourth author would like to thank FAPDF for the financial support (grant 40/2020). We also would like to thank E. Alcobaça and F. Siqueira, developers of the pymfe package, for implementing additional features in the package which help us in this study.

References

[1] B. Tarnoff. (2018) Big data for the people: it's time to take it back from our tech overlords. [Online]. Available: https://www.theguardian.com/technology/2018/mar/14/tech-big-data-capitalism-give-wealth-back-to-people

[2] L. Finger. (2014) 3 data products you need to know. [Online]. Available: https://www.forbes.com/sites/lutzfinger/2014/08/19/3-data-products-you-need-to-know

[3] T. M. Mitchell, *Machine Learning*, ser. McGraw Hill series in computer science. McGraw Hill, 1997.

[4] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[5] J. Gama and M. M. Gaber, *Learning from data streams: processing techniques in sensor networks*. Springer, 2007.

[6] U. Johansson, C. Sönströd, H. Linusson, and H. Boström, "Regression trees for streaming data with local performance guarantees," in *IEEE International Conference on Big Data (Big Data)*, 2014, pp. 461–470.

[7] A. L. D. Rossi, A. C. P. L. F. de Carvalho, C. Soares, and B. F. de Souza, "MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data," *Neurocomputing*, vol. 127, pp. 52–64, 2014.

[8] R. Anderson, Y. S. Koh, G. Dobbie, and A. Bifet, "Recurring concept meta-learning for evolving data streams," *Expert Systems with Applications*, vol. 138, pp. 1–18, 2019.

[9] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "Having a blast: Meta-learning and heterogeneous ensembles for data streams," in *IEEE International Conference on Data Mining (ICDM)*, 2016, pp. 1003–1008.

[10] M. N. Zarmehri and C. Soares, "Using metalearning for prediction of taxi trip duration using different granularity levels," in *14th International Symposium on Intelligent Data Analysis (IDA)*, 2015, pp. 205–216.

[11] J. Vanschoren, "Meta-learning: A survey," *eprint arXiv*, no. 1810.03548, pp. 1–29, 2018.

[12] A. Rivolli, L. P. F. Garcia, C. Soares, J. Vanschoren, and A. C. P. L. F. de Carvalho, "Towards reproducible empirical research in meta-learning," *eprint arXiv*, no. 1808.10406, pp. 1–41, 2018.

[13] M. A. Muñoz, L. Villanova, D. Baatar, and K. Smith-Miles, "Instance spaces for machine learning classification," *Machine Learning*, vol. 107, no. 1, pp. 109–147, 2018.

[14] A. L. D. Rossi, A. C. P. L. F. Carvalho, and C. Soares, "Meta-learning for periodic algorithm selection in time-changing data," in *Brazilian Symposium on Neural Networks (SBRN)*, 2012, pp. 7–12.

[15] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *31st International Conference on Neural Information Processing System (NIPS)*, 2017, pp. 3149–3157.

[16] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *International Conference on Data Mining (SIAM)*, 2007, pp. 443–448.

[17] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, "Evaluation methods and decision theory for classification of streaming data with temporal dependence," *Machine Learning*, vol. 98, no. 3, pp. 455–482, 2015.

[18] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[19] J. Gama, "A survey on learning from data streams: current and future trends," *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 45–55, 2012.

[20] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *ACM SIGMOD Record*, vol. 34, no. 2, pp. 18–26, 2005.

[21] J. Gama, *Knowledge discovery from data streams*. CRC Press, 2010.

[22] P. Domingos and G. Hulten, "Mining high-speed data streams," in *6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 71–80.

[23] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 97–106.

[24] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *8th International Symposium on Intelligent Data Analysis (IDA)*, 2009, pp. 249–260.

[25] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[26] J. R. Rice, "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[27] K. A. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Computing Surveys*, vol. 41, no. 1, pp. 1–25, 2008.

[28] C. Soares, J. Petrak, and P. Brazdil, "Sampling-based relative landmarks: Systematically test-driving algorithms before choosing," in *10th Portuguese Conference on Artificial Intelligence (EPIA)*, 2001, pp. 88–95.

[29] M. Reif, "A comprehensive dataset for evaluating approaches of various meta-learning tasks," in *1st International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, 2012, pp. 273–276.

[30] M. Reif, F. Shafait, M. Goldstein, T. Breuel, and A. Dengel, "Automatic classifier selection for non-experts," *Pattern Analysis and Applications*, vol. 17, no. 1, pp. 83–96, 2014.

[31] C. Castiello, G. Castellano, and A. M. Fanelli, "Meta-data: Characterization of input features for meta-learning," in *2nd Modeling Decisions for Artificial Intelligence (MDAI)*, vol. 3558, 2005, pp. 457–468.

[32] S. Segrera, J. Pinho, and M. N. Moreno, "Information-theoretic measures for meta-learning," in *3rd Hybrid Artificial Intelligence Systems (HAIS)*, 2008, pp. 458–465.

[33] H. Bensusan, C. Giraud-Carrier, and C. Kennedy, "A higher-order approach to meta-learning," in *10th International Conference Inductive Logic Programming (ILP)*, 2000, pp. 1–10.

[34] Y. Peng, P. A. Flach, C. Soares, and P. Brazdil, "Improved dataset characterisation for meta-learning," in *5th International Conference on Discovery Science (DS)*, vol. 2534, 2002, pp. 141–152.

[35] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[36] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.

[37] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning - Applications to Data Mining*, 1st ed., ser. Cognitive Technologies. Springer, 2009.

[38] J. Read, A. Bifet, B. Pfahringer, and G. Holmes, "Batch-incremental versus instance-incremental learning in dynamic and evolving data," in *11th International Symposium on Intelligent Data Analysis (IDA)*, 2012, pp. 313–323.

[39] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "Algorithm selection on data streams," in *17th International Conference on Discovery Science (DS)*, 2014, pp. 325–336.

[40] R. Klinkenberg, "Meta-learning, model selection, and example selection in machine learning domains with concept drift," University of Dortmund, Tech. Rep., 2005.

[41] J. Gama and P. Kosina, "Recurrent concepts in data streams classification," *Knowledge and Information Systems*, vol. 40, no. 3, pp. 489–507, 2014.

[42] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "The online performance estimation framework: heterogeneous ensemble learning for data streams," *Machine Learning*, vol. 107, no. 1, pp. 149–176, 2018.

[43] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[44] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[45] C. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, 2001, pp. 682–688.

[46] E. Alcobaça, F. Siqueira, A. Rivolli, L. P. F. Garcia, J. T. Oliva, and A. C. P. L. F. de Carvalho, "MFE: Towards reproducible meta-feature extraction," *Journal of Machine Learning Research*, pp. 1–5, 2020.

[47] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *17th Brazilian Symposium on Artificial Intelligence*, 2004, pp. 286–295.

[48] X. Zhu, "Power supply: Stream data mining repository," 2010. [Online]. Available: http://www.cse.fau.edu/~xqzhu/stream.html

[49] J. A. Blackard, D. J. Dean, and C. W. Anderson, "Covertype data set, UCI machine learning repository," 1998. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/covertype

[50] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.