2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 11-14, 2020. Toronto, Canada

# Combining Slow and Fast Learning for Improved Credit Scoring

Jean Paul Barddal, Fabrício Enembreck
Graduate Program in Informatics (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba, Brazil
{jean.barddal, fabricio}@ppgia.pucpr.br

Lucas Loezer, Riccardo Lanzuolo
4KST
Curitiba, Brazil
{loezer, riccardo}@4kst.com

***Abstract*—The financial credibility of a person is a relevant factor to determine whether a loan should be approved or not, and it is quantified by a credit score, which is computed using past performance on debt obligations, profiling, and other data available. Credit scoring becomes even a hotter topic in emerging countries, as interest rates and customer behavior swiftly vary, given the economic (in)stability of the country and as fintechs are chasing robust solutions for improved credit scoring solutions. Batch machine learning is often deployed for credit scoring, yet, they are tailored for static scenarios, i.e., they are not prepared to swiftly detect and adapt to changes in customer behavior, thus leading to slow recovery in such scenarios. In this paper, we bring forward an analysis on how batch machine learning can be combined with data stream mining techniques, thus leading to better recognition rates in credit scoring scenarios. We analyze three different real-world datasets from Brazilian financial institutions, whilst keeping their secrecy preserved, and show how batch and stream learning can be combined towards improved credit scoring systems, as well as highlighting relevant gaps that still require attention.**

***Index Terms*—Credit scoring, machine learning, data streams**

## I. Introduction

The financial credibility of a person is a factor used to determine whether a loan should be approved or not. It is quantified by a 'credit score,' which is computed using factors that include a person's information on past performance on debt obligations, profiling, main household, income, occupation, demographics, possessions (e.g., cars, and other residences, if any), and census information. The development and management of effective and reliable risk assessment credit scoring models are time-consuming, and over decades, multiple automatic credit scoring models have been created using machine learning techniques. Even with the help of machine learning, application papers such as [1] show that the development of a robust credit scoring model can range from 3 to 18 months. As a result, it is not rare for financial institutions and credit scoring operators to use the same credit scoring model for years without changes. As brought up by authors in [2], if a model is built on top of 2 or more years of historical data, while shifted 3 years away from the point they will be used, a 5-year shift is often exceeded.

In emerging countries, financial institutions suffer from increased default rates as the market is volatile, meaning that their predictive models are unable to keep up with the dynamics of the credit market. Therefore, the development of robust machine learning models that are able to evolve over time, i.e., using data stream learning techniques, has become a hot topic in which fintechs are working on.

In this paper, we bring forward an analysis of how slow (traditional batch machine learning) and fast (data stream mining techniques) approaches can be combined towards improved credit scoring systems. We analyze 3 different real-world datasets from Brazilian institutions and show how these can be combined following simple strategies that leverage the scoring rates. We also bring forward an analysis of existing gaps that are yet to be filled with research in diverse aspects of machine learning towards its application to credit scoring.

## II. Credit Scoring

Credit scoring is of the utmost importance for risk management in financial institutions as it works as a proxy to predict the risk of loan applications. Statistical models are used to estimate default probability upon customer information such as past performance on debt obligations, profiling, main household, income, occupation, demographics, possessions, e.g., cars, other residences, and census information. Over time, standard approaches such as *scorecards* were replaced by automated machine learning models since both the dimension and size of historical data are ever-growing [3].

The development of robust scoring models based on machine learning depends on several factors, including (i) the gathering of veracious historical data, (ii) the identification of key attributes from the customer and his/her past loans, and the (iii) proper construction and validation of the predictive model. Another relevant trait of predictive models for credit scoring is that all the stakeholders may impose constraints on which kind of data can be used, often to enable audits and prevent issues, e.g., discriminatory behavior. For instance, in the scope of our study, all credit scoring models in use in Brazil may be audited by the Central Bank of Brazil, so understanding how 'impactful' each variable is during prediction is of the utmost importance. In practice, this is one of the biggest reasons on why financial institutions often rely on Logistic Regression, as each variable can be analyzed individually, and their coefficients can be checked to see how they lean

978-1-7281-8526-2/20/$31.00 ©2020 IEEE

the classification towards creditworthy or non-creditworthy requests.

In this paper, we denote $(\vec{x}, y)$ to be a loan request, where $\vec{x}$ a vector of characteristics (features) detailing the loan request and the customer requesting it, and $y \in \{0, 1\}$ the target feature, such that 0 represents that the customer paid this specific loan in full, and 1 that the customer defaulted. The classification task targets the creation of a predictive model $h : \vec{x} \rightarrow y$ that accurately maps features and their values into classes. Naturally, different types of classification systems exist, including, for instance, decision trees, linear regression models, and ensembles; and these are discussed in Section III.

Instead of providing a boolean answer determining whether a customer loan request is expected to be fully paid or not, financial institutions and credit operators often work with scores. A score is usually bounded in the $[0; 1,000]$ interval, where 0 is the value that represents a customer that is undoubtedly going to default, while $1,000$ represents a customer that will surely pay his debts in full. In practice, these extreme values are nearly inexistent in real-world applications, so institutions should decide which threshold to use and discern between customers who should be granted a loan or not. To obtain this type of scores, we require classifiers that instead of only providing their classification outputs $h(\vec{x})$, also provide probabilities $P[h(\vec{x}) = 1]$ and $P[h(\vec{x}) = 0]$, which can be directly re-scaled to the $[0; 1,000]$ interval as $P[h(\vec{x}) = 0] + P[h(\vec{x}) = 1] = 1$.

The assessment of machine learning tailored for credit scoring requires specific metrics that highlight how well creditworthy and non-creditworthy customers are discerned. To perform the assessment of the ability of machine learning methods to discriminate customers that will pay their debts in full or not, we follow the Kolmogorov-Smirnov (KS) metric [4]. The KS statistic indicates the maximum distance between the cumulative probability distribution function (*cdfs*) obtained by customers that pay their debts in full and those who default [4]. Assuming that we are scoring $(n + m)$ customers, we denote that the $i$-th customer will default as $D_i = 1$, and $D_i = 0$ otherwise. Also, the empirical cumulative distribution function (*cdf*) of good and bad customers are given by Equations 1 and 2, respectively, where $n$ is the total number of good customers, $m$ is the number of bad customers, $L = \min s_i, 1 \leq i \leq (n + m)$ is the lower bounds of all the scores available, $H = \max s_i, 1 \leq i \leq (n + m)$ is the upper bound, and $a \in [L, H]$.

$$F_{\text{good}}(a) = \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 1, \text{if } s_i \leq a \wedge D_i = 1 \\ 0, \text{otherwise} \end{cases} \quad (1)$$

$$F_{\text{bad}}(a) = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} 1, \text{if } s_i \leq a \wedge D_i = 0 \\ 0, \text{otherwise} \end{cases} \quad (2)$$

The KS metric is given by $\text{KS} = \max_{a \in [L,H]} |F_{\text{bad}}(a) - F_{\text{good}(a)}|$, which is the maximum difference between the *cdfs* that describe the good and bad customers. When KS is zero, it means that the two credit-score distributions are the same and that the credit score fails to differentiate between defaulters and nondefaulters; a value equal to 100 indicates that the credit score perfectly differentiates defaulters from non-defaulters. As a rule of thumb, a KS score greater than 35% depicts a reasonable discriminative power of the predictive model to discern between the different types of customers.

In addition to KS, the population Stability Index (PSI) indicates changes in the population of loan applicants. It is important to note that this may or not be an indication of deterioration of the predictive model to predict risk, yet, PSI depicts changes in the environment that need to be further investigated by the bank experts to determine whether any macroeconomic conditions or lending policies are affecting the model outcomes [5]. To compute the PSI score, the probability distribution function (*pdf*) of the defaulting customers in two different time periods is calculated. First, the *pdfs* of these distributions are computed using a specified number of ranges $r$ so that each range has approximately the same number of defaulting customers. Here, we denote $n_i$ and $m_i$ to be the counters of defaulting customers in the two samples at the $i$-th bin, and that $\sum n_i = N$ and $\sum m_i = M$. Given these counters, it is possible to compute the PSI, which is given by $\sum_{i=1}^{r} \left[ \left( \frac{n_i}{N} - \frac{m_i}{M} \right) \times \left( \ln \frac{n_i}{N} - \ln \frac{m_i}{M} \right) \right]$. In our analysis, we report PSI rates obtained by comparing two subsequent months. As a rule of thumb often followed, PSI rates below 10% show that the population is reasonably stable and that the model is not unacceptably volatile.

## III. Machine Learning Models for Credit Scoring

Machine learning is a hot topic that aims at building and assessing predictive models. In this paper, we use both batch and stream learners. In batch machine learning, constraints on computational resources are roughly overlooked, and the ultimate goal is to favor models that generalize well a real-world phenomenon underlying in the dataset available. For instance, Random Forest [6], Extreme Gradient Boosting [7], and deep learning models [8] are popular machine learning choices that require large datasets, and often large memory consumption and processing time.

Conversely, machine learning from streaming data learns as data becomes available, and thus, storing data becomes unfeasible as the data stream is potentially infinite and is not sampled from an i.i.d. distribution. Therefore, models are incrementally trained and each arriving datum is analyzed and discarded right after. The reason for continuously adapting predictive models is that streaming data may drift over time, as the relationship between input data and the response (in our case, whether the customer is creditworthy or not) may change over time.

In this section, we describe the learning algorithms used in our analysis. These algorithms were chosen due to their availability in data mining frameworks, i.e. scikit-learn [9], RAPIDS AI [10], xgboost [7], and MOA [11]; by achieving state-of-the-art results [12], [13], and also per request of our

partners. Below, we describe these learners and divide them into batch and stream learners, where the former are trained over a static amount of training data, while the latter can be incremented over time. Upon availability of large datasets, batch learners can be used to learn complex and accurate models, while streaming approaches can adapt to changes faster as it takes into account more recent data. Therefore, our driving rationale is to show that batch and streaming data must not be replacing but complementary.

### A. Batch Learners

**Logistic Regression (LR).** Logistic regression is a linear model where the target variable (class) is categorical [14]. It creates a linear model based on a sigmoid function that is used to estimate the probability of a binary response based on the input features. This is, by far, the most widely used approach for credit scoring, and all 3 of our partners adopt it in their current models. The reason why LR is so popular with financial institutions is that they are human-readable, which means that each feature and its coefficient can be analyzed individually, thus determining how important it is to discern between creditworthy and non-creditworthy customers and that scorecards can be derived from it [3]. **Random Forest (RF).** The Random Forest classifier [6] is an ensemble of unpruned classification trees, which are induced from bootstraps of the training data [15]. In contrast to conventional decision trees, during the branching process, only a randomly selected subset of features is evaluated [16]. The final predictions scores are obtained by averaging the output probabilities given by each of the trees. **XGBoost (XGB).** XGBoost is an additive learning scheme where decision trees are learned sequentially with the goal of minimizing a loss function [7] while not overfitting. The implementation of XGBoost used in this paper learns decision trees sequentially while using both horizontal and vertical sampling.

### B. Stream Learners

**Leveraging Bagging (LB).** Leveraging Bagging [13] is an extension to Online Bagging [17] where the weights of training instances are randomized according to a Poisson distribution with a mean $\lambda = 6$, the ADWIN drift detector [18] is used to flag drifts and reset classifiers accordingly, and random output codes are used to improve the accuracy of the whole ensemble. As in the original paper, we have conducted our analysis by creating a leveraging Bagging ensemble of Hoeffding trees [19] with the default values available in the MOA framework [11], i.e., an ensemble size of 10 trees. Finally, probabilities are obtained by averaging the probabilities obtained in each of the subtrees available in the ensemble. **Adaptive Random Forest (ARF).** The Adaptive Random Forest algorithm (ARF) was introduced in [12] with the goal of allowing adaptive learning from data streams by extending the original Random Forests of Breiman [6]. ARF combines drift detection as in Leveraging Bagging, ensemble adjustments, limited tree sizes, and background learning to improve accuracy rates over concept drifting data streams. As in the other ensembles, the final output probability scores are obtained by averaging the outputs obtained from each subtree.

TABLE I: Hyper-parameters and values tested in the experiments.

| Learner | Parameter | Value |
|---|---|---|
| Logistic Regression | Penalty | L1, L2, ElasticNet |
| | Intercept | Yes, No |
| | Regularization factor | 0.1, 0.01, 0.001 |
| Random Forest | Number of learners | 50, 100, 150 |
| | Maximum depth | 3, 5, 10, Unlimited |
| | Number of features analyzed during split | $\sqrt{d}$, $\log_2 d$ |
| XGBoost | Number of learners | 50, 100, 150 |
| | Sample percentage | 60%, 80%, 100% |
| | Column sampling per tree | 60%, 80%, 100% |
| | Maximum depth | 3, 5, 10 |
| Adaptive Random Forest | Number of learners | 50, 100, 150 |
| | Maximum depth | 3, 5, 10, Unlimited |
| | Number of features analyzed during split | $\sqrt{d}$, $\log_2 d$ |
| Leveraging Bagging | Number of learners | 50, 100, 150 |
| | Resampling factor | 1, 6 |

### C. Parametrization

In the following experiments, batch learners, i.e., Logistic Regression, Random Forest, and XGBoost, were trained using a 10-fold cross-validation scheme using a grid search process for parameter tuning. On the other hand, tuning is a complex process when the data is processed as a stream. In practice, after the model is deployed, it is roughly impossible to guarantee that the hyper-parameters were set appropriately, as they may also drift along with the data. Still, a grid search has been conducted with the goal of determining a fair set of parameters' values for posterior testing in the specified time period. Even though 10-fold cross-validation is not possible in streaming scenarios as the order in which the data is made available is relevant, the goal of the tuning process was maximizing the average monthly prequential KS rates. The learners, hyper-parameters, and values tested throughout this process are given in Table I.

## IV. Experimentation

In batch machine learning, the dataset is often divided into training and test subsets, a process called holdout validation, with the goal of determining whether the predictive model built generalizes and performs well on unseen data. In the following experiments, we follow the holdout procedure for assessing batch learners. The definition of which data periods are used for training and testing depend on the dataset, and this information is given in Section IV-A. Yet, data stream mining techniques are tailored to take advantage of the arrival of new data, and thus, the traditional holdout process cannot be used. Therefore, we used a prequential-like validation process [20], where the data of each month is used for training right after its evaluation. In practice, the entire training set is still used solely for training, yet, the test data is passed to the learner for training as their label becomes available, i.e., at the end of each month. The rationale is to compare the results obtained by a learner that is constantly being updated with new data against a model that was learned until a certain

point and only evaluated after. By comparing these results, and also by combining them, we will be able to identify whether continuously updating predictive models significantly improves the KS results without jeopardizing PSI rates.

### A. Datasets

In this section, we present the main characteristics of the datasets analyzed. We use 3 different datasets during this study that were borrowed from [21], each obtained from a different financial institution, hereafter referred to as DS1, DS2, and DS3. We refrained from using credit scoring datasets available in the literature [22], [23] as they are either (i) too small in terms of instances available, and in some cases, (ii) unclear on what each feature stands for.

### B. Combination rules

In the following experiments, we report the results obtained by combining the predictions from batch and stream learners in different ways. Assuming that the classifiers described above are enlisted in $H = \{h_1, h_2, \ldots, h_n\}$, and that $P_i$ is the probability obtained by the $i$-th classifier that a loan request $\vec{x}$ will default ($P[h_i(\vec{x}) = 1]$), we tested extracted the minimum ($H(\vec{x}) = \min P[h_i(\vec{x}) = 1]$), maximum ($H(\vec{x}) = \max P[h_i(\vec{x}) = 1]$), and average probabilities ($H(\vec{x}) = \frac{1}{n}\sum_{i=1}^{n} P[h_i(\vec{x}) = 1]$) of the entire set of classifiers.

Minimum, maximum, and average combination rules have been applied twice. First, they were applied globally, as all learners were accounted for, while the second approach was to combine the probabilities obtained by the two best-performing classifiers (one batch and one stream). Hereafter, we will refer to the global rules using the MIN, MAX, and AVG, terms, whilst the best-performing models will be reported using the MIN-TOP2, MAX-TOP2, and AVG-TOP2 terms.

Besides the aforementioned combination rules, a meta-classification scheme was also tested, as the probabilities $P[h_i(\vec{x}) = 1]$ were used as input to a Logistic Regression model. An important aspect of this approach is that only the probabilities obtained during training data were used to create this model, and thus, it does not account for changes that might take place during the updates of streaming models. In the following experiments, we refer to this approach as META.

### C. Analysis

In this section we analyze the results obtained in terms of KS and PSI in the DS1, DS2, and DS3 experiments. The KS rates obtained in the DS1 experiment are given in Figure 1. From the Logistic Regression (LR) results, we observe that it is roughly in the middle of the plot, showing mild KS rates compared to all the other approaches tested. Focusing on the data stream algorithms, i.e., Adaptive Random Forest (ARF) and Leveraging Bagging (LEVBAG), we observe that ARF shows competitive results from MONTH 3 and beyond, while LEVBAG depicts poor results during the entire test set. On the other hand, the batch algorithms, i.e., XGBoost (XGB) and Random Forest (RF) depict a conflicting behavior. The MIN and AVG combination rules show that they are unable to

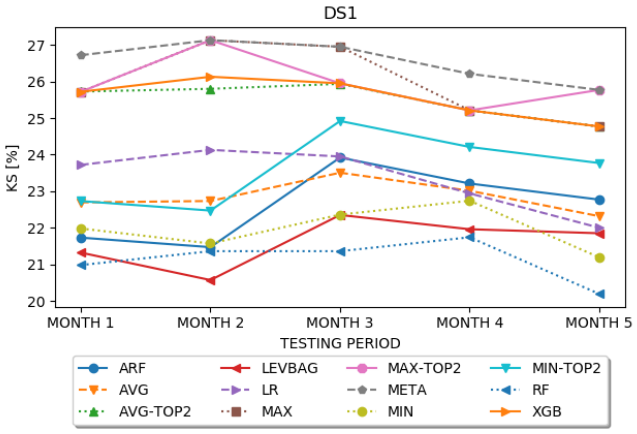


Fig. 1: KS analysis for the DS1 experiment.

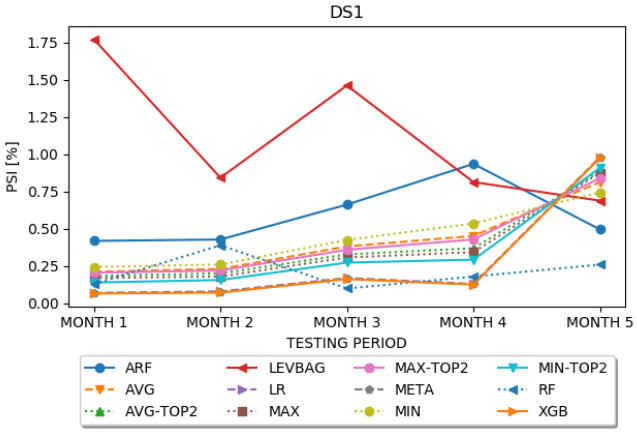


Fig. 2: PSI analysis for the DS1 experiment.

overcome Logistic Regression (LR), while MIN-TOP2 showed improvements only in the last 3 testing periods. Conversely, MAX, MAX-TOP2, and AVG-TOP2 yielded interesting improvements compared to LR and XGB. Finally, we emphasize the results obtained by META, as it depicted the best overall KS rates throughout the testing months. The PSI rates observed throughout the test data for DS1 are given in Figure 2. Overall, most of the learners, as well as their combination, yield PSI rates below 0.5%, while the stream learners (ARF and LEVBAG), have increased PSI that reach 1.75%. Despite such spikes and differences, all the PSI rates are low, so these values are not determinant for model selection.

The KS results for DS2 are given in Figure 3. As in the previous experiment, Logistic Regression (LR) shows mild KS rates compared to the remainder of the tested approaches. Clearly below or tied with it, we observe ARF, MIN, AVG, MIN-TOP2, and LEVBAG. On the other hand, the results for batch learners (RF and XGB), show consistent improvements to those seen by LR. These results show that both of the stream learners are below the LR baseline, yet, when combined with XGB, according to META, MAX, MAX-TOP2, and AVG-TOP2 approaches, even improvements to XGB are observed.

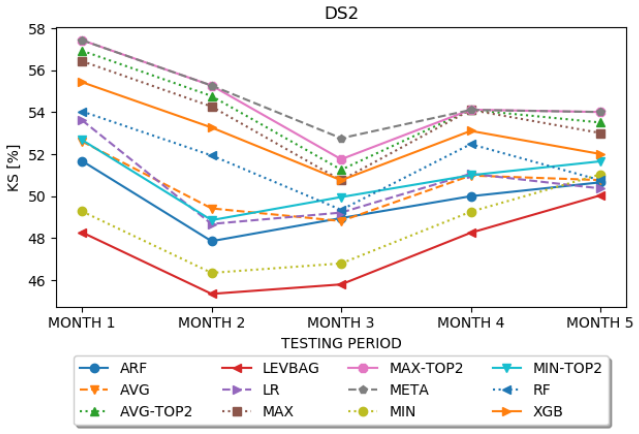


Fig. 3: KS analysis for the DS2 experiment.

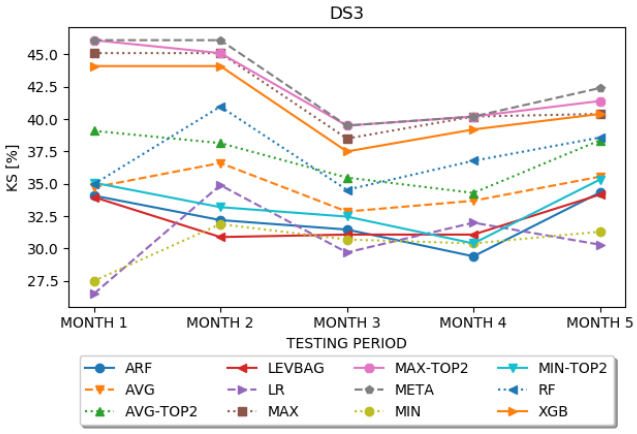


Fig. 5: KS analysis for the DS3 experiment.

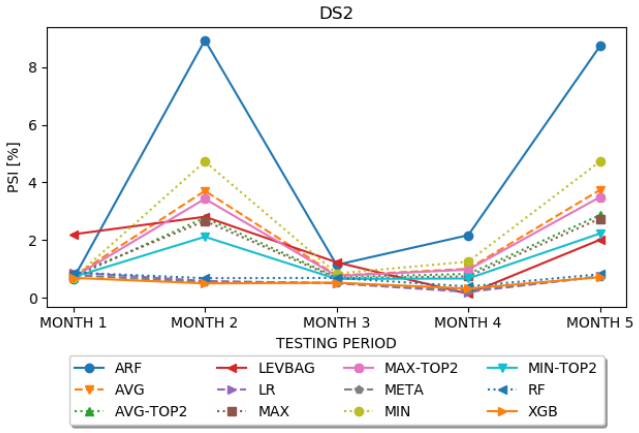


Fig. 4: PSI analysis for the DS2 experiment.

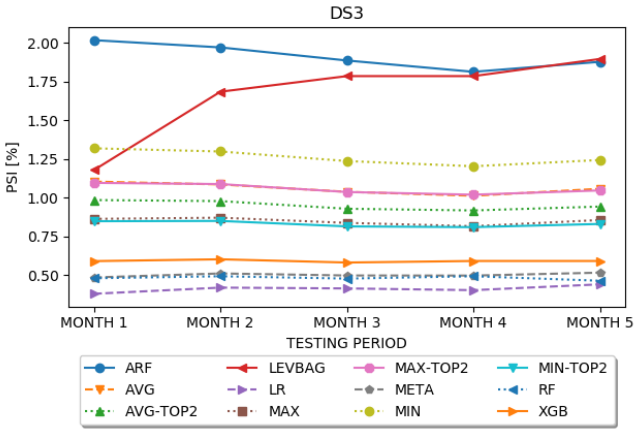


Fig. 6: PSI analysis for the DS3 experiment.

The PSI results for DS2 are given in Figure 4. In contrast to the previous experiment, the PSI values observed are higher, as most of the learners depict PSI rates that grow, on average, to 3.5%. An important behavior is observed by ARF, as its PSI nearly reaches 9% in the second month of the test dataset. Overall, these PSI rates are also inconclusive and do not affect the selection of a predictive model or even their combination.

The KS results for DS3 are given in Figure 5. In opposition to the previous experiments, the KS rates obtained by Logistic Regression are at the bottom part of the plot. With similar KS rates, we see MIN-TOP2, LEVBAG, ARF, and MIN. The Random Forest classifier (RF) obtained mild results, while XGBoost (XGB) achieved interesting KS rates compared to others. In opposition to the previous experiments, we see that MAX and MAX-TOP2 achieved very interesting rates, thus showing that the combination of XGB with ARF leverages the final KS rates, despite the existing gap w.r.t. their performance. Additionally, AVG and AVG-TOP2 showed that averaging the probabilities is unbeneficial, as the KS rates are much below the rates seen for XGB. We also observe that the META approach overcomes the remainder throughout the entire test set. Regarding PSI rates, given in Figure 6, we observe that this dataset is much more stable than others. In practice, most of the learners and their combinations yield stable PSI rates throughout the entire test set. The exceptions are again the stream learners, as both ARF and LEVBAG achieve slightly higher PSI rates, up to 2%.

Summing up the results, it is observed that the combination of batch and stream learning is fruitful, mainly in terms of KS. Regardless of whether drifts are present or not and whether there is a discrepancy between the KS rates of batch and stream learners, their combination yielded interesting KS improvements. In practice, assuming the XGB as the baseline and using the META combiner, the average KS improvements in the DS1, DS2, and DS3 experiments were of 1.01%, 1.80%, and 1.82%, respectively.

### D. Challenges

Despite the interesting results displayed in the previous section, it is important to clarify challenges that still exist and demand the attention of financial institutions and fintechs.

One important aspect shared amongst all datasets in the analysis is that labels are assumed to become available at the end of each month. Even though this is reasonable, financial

institutions often label their customers according to whether they paid their loan in the past 3 or even 6 months. This is a scenario where labels are delayed, and thus, techniques tailored for taking advantage of unlabelled instances could be deployed. For instance, semi-supervised learning [24], and co-training [25] could be applied, so that loan requests are used for model updates as soon as they arrive instead of waiting up to 3 months until their actual label is available.

Another important aspect behind the combination of batch and stream learning is *model explainability*. As mentioned in Section II, logistic regression is widely applied to credit scoring as they are human-readable and can easily derive scorecards. Similarly, more complex models such as Random Forests and XGBoost can also provide explainable predictions and feature importances, for instance, with the use of Shapley Values [26]. The issue relies on streaming models as they adapt to changes, and both feature importances are also expected to drift as well as the explainability behind each prediction. Therefore, it becomes relevant not only to propose feature ranking approaches such as in [27], but also make assure that each prediction is explained individually, which is untouched ground thus far for streaming models.

## V. Conclusion

In this paper, we discuss an important topic for fintechs: credit scoring. We showed how traditional batch machine learning algorithms can be combined with data stream mining approaches for improved recognition rates. We brought forward an analysis of the proposed combination scheme in two real-world datasets provided by partners from financial institutions in Brazil. Results show that, in emerging countries, where the customer behavior drifts over time, this combination can be fruitful and deserves further research and application. Therefore, we claim that financial institutions should account for data stream learning and assess whether improved credit scoring systems can be obtained by diversifying their frameworks and going beyond Logistic Regression. We also brought forward technical aspects that still require attention from both researchers and practitioners, as certain label availability and computational constraints are yet to be addressed.

## Acknowledgements

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the TITAN V GPU used for this research.

## References

[1] D. J. Hand and N. M. Adams, "Selection bias in credit scorecard evaluation," *Journal of the Operational Research Society*, vol. 65, pp. 408–415, Mar 2014.

[2] I. Žliobaitė, M. Pechenizkiy, and J. Gama, *An Overview of Concept Drift Applications*, pp. 91–114. Cham: Springer International Publishing, 2016.

[3] G. Dong, K. K. Lai, and J. Yen, "Credit scorecard based on logistic regression with random coefficients," *Procedia Computer Science*, vol. 1, no. 1, pp. 2463 – 2468, 2010. ICCS 2010.

[4] M. Rezac and F. Rezac, "How to Measure the Quality of Credit Scoring Models," *Czech Journal of Economics and Finance (Finance a uver)*, vol. 61, pp. 486–507, November 2011.

[5] G. Karakoulas, "Empirical validation of retail credit-scoring models," *The RMA Journal*, vol. 87, no. 1, pp. 56–60, 2004.

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

[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] RAPIDS Development Team, *RAPIDS: Collection of Libraries for End to End GPU Data Science*, 2018.

[11] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, Aug. 2010.

[12] 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, pp. 1469–1495, Oct 2017.

[13] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Machine Learning and Knowledge Discovery in Databases* (J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, eds.), (Berlin, Heidelberg), pp. 135–150, Springer Berlin Heidelberg, 2010.

[14] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 215–242, 1958.

[15] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, Aug 1996.

[16] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 832–844, Aug 1998.

[17] N. C. Oza, "Online bagging and boosting," in *International Conference on Systems, Man, and Cybernetics, Special Session on Ensemble Methods for Extreme Environments* (M. Jamshidi, ed.), (New Jersey), pp. 2340–2345, Institute for Electrical and Electronics Engineers, October 2005.

[18] A. Bifet, "Adaptive learning and mining for data streams and frequent patterns," *SIGKDD Explor. Newsl.*, vol. 11, pp. 55–56, Nov. 2009.

[19] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, (New York, NY, USA), pp. 71–80, ACM, 2000.

[20] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, vol. 90, pp. 317–346, Mar 2013.

[21] J. P. Barddal, L. Loezer, F. Enembreck, and R. Lanzuolo, "Lessons learned from data stream classification applied to credit scoring," *Expert Systems with Applications*, p. 113899, 2020.

[22] J. Abellán and J. G. Castellano, "A comparative study on base classifiers in ensemble methods for credit scoring," *Expert Systems with Applications*, vol. 73, pp. 1 – 10, 2017.

[23] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[24] T. Wagner, S. Guha, S. Kasiviswanathan, and N. Mishra, "Semi-supervised learning on data streams via temporal label propagation," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm Sweden), pp. 5095–5104, PMLR, 10–15 Jul 2018.

[25] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, (New York, NY, USA), p. 92–100, Association for Computing Machinery, 1998.

[26] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), p. 1135–1144, Association for Computing Machinery, 2016.

[27] J. A. P. Karax, A. Malucelli, and J. P. Barddal, "Decision tree-based feature ranking in concept drifting data streams," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, (New York, NY, USA), p. 590–592, Association for Computing Machinery, 2019.