Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Estatística
Programa de Especialização em *Data Science* e *Big Data*

Antonio Ricardo Lunardi

Streaming and Concept Drift: A Comparative Study

Curitiba
2024

Antonio Ricardo Lunardi

Streaming and Concept Drift: A Comparative Study

Monografia apresentada ao Programa de Especialização em *Data Science* e *Big Data* da Universidade Federal do Paraná como requisito parcial para a obtenção do grau de especialista.

Orientador: Prof. Paulo Ricardo Lisboa de Almeida

Curitiba
2024

Streaming and Concept Drift: A Comparative Study

Antonio Ricardo Lunardi[1], Paulo Ricardo Lisboa de Almeida[2]

[1]Data Science & Big Data Specialization Program Student, antonio.lunardi@ufpr.br
[2]Department of Informatics Professor - DInf/UFPR, paulo@inf.ufpr.br

Este estudo começa por discutir os princípios de dados em fluxo (*streaming*) e o fenômeno de desvio de conceito (*concept drift)*. Com o intuito de determinar o quão bons são algoritmos para detecção de desvio de conceito associados a modelos de aprendizado de máquina, vários testes são conduzidos com River, uma API para Python. Conjuntos de dados públicos são examinados a fim de verificar os possíveis comportamentos dos modelos. Algoritmos de detecção são usados como gatilhos para retreinar e reconstruir modelos, tornando o processo de aprendizagem adaptativo para lidar com desvio de conceito. Conjuntos de dados que aparentemente possuem desvio de conceito foram utilizados para treinar os modelos propostos, todo algoritmos de detecção se saiu melhor que os demais pelo menos para um conjunto de dados. Desse modo, todos os detectores provaram ser de alguma forma úteis, apesar de alguns deles terem a acurácia média menor do que a linha de base para decisão (*baseline*). Por outro lado, para os conjuntos de dados aparentemente sem desvio de conceito, os detectores pioraram o desempenho dos modelos, ou simplesmente não fizeram diferença nos casos de melhor resultado. Isso leva o presente estudo a concluir que em cenários de dados reais, deve-se utilizar uma linha de base que considera que o fluxo de dados não apresenta nenhum desvio.

This study starts by discussing the fundamentals of data streaming and concept drift. In order to verify how good are the drift detector algorithms combined with machine learning models, many tests were conducted with Python River API. Available public streaming datasets were divided into two groups aiming to verify the models possible distinct behaviors. Detecting algorithm were used as triggers to retrain and rebuild the models, making the process adaptive to deal with concept drifts. The results showed that when dealing with streaming datasets with probable concept drift, every tested drift detector was the relative best one at least for one dataset. Thus, all detectors proved themselves somehow useful. However, some models had the mean accuracy lower than the baseline mean. On the other hand, in the case of streaming datasets with apparently no drift, the detectors have worsen, or in better cases didn't improve the models at all. This leads the present study to conclude that in real world scenarios, it is crucial to have a baseline that supposes that the stream doesn't present any drift.
Keywords: streaming, concept drift, drift, classification, adaptive learning, machine learning, River, Python, Hoeffding Tree, ADWIN, DDM, EDDM, FHDDM, HDDM, HDDMA, HDDMW, KWSIN, Page Hinkley.

1. Introduction

Classical classification approaches consider frequently that data distribution probability is not time dependent, but constant. The concept drift phenomenon can be described as a data distribution change that occurs over time. It is of vital importance to consider when dealing with concept drifts what information is relevant in the present analysis and how should the transformations be treated [1].

Data stream data mining differs from batch in five aspects [2]:

1. Data pass a single time.
2. Time requirement needed is limited.
3. Required memory is limited.
4. There are more than one concept.
5. Obtained results are approximate.

Authors have proposed several different terminologies for concept drift such as concept shift, dataset shift, concept shift. Concept drift understanding requires to set the information about the moment that the phenomenon happened, its region and severity. There are many ways to identify a concept drift time stamp depending on the chosen algorithm. The main features of stream data are: Data streams are unbounded in size.

Data arrives at a high rate and varying velocity. Data may evolve over time [2][8].

In order to define mathematically the environment, the target variable can be described as $Y \in \mathbb{R}$. Time variable as $t \in [0, \infty[$. The input variable as $X \in \mathbb{R}^p$ considering p as the dimensions number. The (X, y) pair joint probability in a given instant of time t is denoted by $P_t(X, y)$. Formally, a concept drift can be described by happening between the time instances t_0 and t_1 as [16]

$$\exists X : \mathrm{P}_{t_0}(X, y) \neq \mathrm{P}_{t_1}(X, y) \tag{1}$$

Being a joint probability, it has two components:

$$P(X, y) = P(X) \cdot P(y \mid X) \tag{2}$$

The recent data streaming growing use demands for specific kinds of machine learning. Models that operate continuously in data streams must learn in a continuous way. Thus, this work has the purpose of investigate how do machine learning adaptive models respond to different kinds of streaming datasets, whether those datasets contain concept drift or not. All these considerations raise some questions: Are the current available drift detector algorithms useful to build machine learning models for streaming environments? Do these detectors improve the performance of classifiers in the presence of any kind of data stream? If they do improve, how much is expected of them to enhance models performances?

This work is divided into eight main sections. The present one **1. Introduction** contains fundamental concepts about the theme in order inform the reader about the broader discussion's essential issues. The following section **2. Problem definition** approaches the more specific theory concerning this this paper's used technologies. **3. State-of-the-art** presents modern technologies that configure the basis of this study. **4. Experimental protocol** describes how procedures were planned and the followed methodology. Then there come the self explanatory sections **5. Results** and **6. Conclusions**.

2. Problem definition

The region of concept drift can be described as the space in conflict between two concepts. There is a strong relation of the model to identify the drift's region and the one for drift's detection. A very limited amount of the existing techniques for drift detection are capable of showing a concept drift's location [8].

There are two categories of concept drift: real and virtual. A virtual concept drift occurs when the *a priori* probability $P_t(y)$ or the unconditional distribution $P_t(X)$ may change, although the best possible classes separation border is kept. Mathematically it means that $P_t(X) \neq P_{t+1}(X)$ and $P_t(y|X) = P_{t+1}(y|X)$. A real concept drift happens if the *a posteriori* probability changes over time. Real concept drifts present $P_t(y|X) \neq P_{t+1}(y|X)$. However, it is still a real one, whether or not having a change in $P_t(X)$, the *a priori* probability [3][1].

Aiming to identify the drift's timestamp algorithms usually have for safety a statistical significance measure that points whether a concept drift has occurred or not. This signal also drives the system to update its learning process. The algorithm should also perceive the moment when the drift ends, in order to define the period of concept drift as a whole. A good way to assess the more appropriate concept drift strategies is to evaluate its intensity, it is also called a concept drift's severity. It is frequently calculated as the difference between two data distributions. Although some techniques and tools already exist for detecting severity, nowadays it is not a widespread practice [8].

Requirements should be taken into consideration when developing a drift detection algorithm such as the memory requirement, the time taken on the learning phase with the last model and the expected generated errors [9]

There are three main strategies for drift adaptation: model adjusting, ensemble retraining and simple retraining. Simple retraining usually bases its working on a window strategy that keeps a part of the data out for statistical distribution change testing. The new data is naturally used to retrain the model. The larger the window is, the more training data will be provided to the model [8].

There are three kinds of measures that can be regarded to spot a concept drift. A change in the prior results in class imbalance. A change in the class distribution leads the system to a virtual concept drift. If the class probability posterior class changes, it reveals a real concept drift [3][7].

A concept can be defined in three fixed aspects: Label, intention and extension. Label is merely an identification. Extension is the affected entities of the analyzed system. Intention are the concepts inner characteristics [10][11][12].

The amount of steps for a change between two concepts is seen as the period that separates stability confi-

gurations. Concept drift speed is defined as the inverse of the instability period [13][4][5][14].

The idea of adaptive learning evolves naturally from the problem of a dynamic environment operating model. As well as the system receives new data, the model should be able to respond to concept drifts [1][16]. Detecting algorithms may be divided into three categories: Error rate-based drift detection, data distribution-based drift detection, and multiple hypothesis test for drift detection. Error rate-based drift detection activates a trigger when a modification in error rate proves itself to be statistically significant within a time window. A classifier may be needed to make error predictions. Statistical tests are used to assess the online error rate and hypothesis tests are responsible to calculate a drift threshold and set a warning level. Data distribution-based drift detection methods usually generate information about the drifts location and the time that drifts occurred. They use a distance function in order to compare historical and new data and divide it into categories if needed. Multiple hypothesis-test drift detection works accurately whether using parallel or hierarchical hypothesis tests [8].

There are many techniques to be applied on concept drifts. They can be summarized in six categories: window based, gradual forgetting-based, triggered-based, ensemble-based, local region-based, distribution analysis-based [1].

3. State-of-the-art

This section describes the drift detection algorithms and the classifier algorithm used to build the machine learning models of the tests.

The base classifier used for all drift detectors was Hoeffding Tree Classifier. Hoeffding Tree algorithm is built upon a decision tree with incremental capacity. Evaluation using it can be done rapidly and is easily explainable due to the tree design. Another interesting feature is that it deals well with combines kinds of data. The algorithm assumes that there is no drift in data stream [17].

The DDM (Drift Detection Method) algorithm monitors the error rate in a period of time during the training, also known as context window. Error rate is updated when every new example is processed, as well as the variance, which is calculated by a binomial function. There are two possible levels in the analysis the warning level and the drift level. The first one has a lower tolerance than the second one. If the warning level is reached, the system alerts for a possible change, but no action is taken. When drift level is reached an alert of drift is activated and a new context is declared [18].

ADWIN (Adaptive Windowing) includes the last part of the stream in a window of arbitrary size. This window is divided into two sub windows. The algorithm compares both sub windows statistically. Assuming a level of confidence $\delta \in (0,1)$, it can detect if both distributions are equivalent or if a drift occurred [19].

KSWIN (Kolmogorov-Smirnov Windowing) also works based on a windows strategy with two sub windows. As the name suggests, Komogorov-Smirnov test is used in the sub window R and W from the sliding window ϕ. Compares the distance of the empirical cumulative data distribution dist(R, W). The level of statistical significance is α, and r is the number of samples in the R concept [20]:

$$\text{dist}(R, W) > \sqrt{\frac{\ln \alpha}{r}} \tag{3}$$

Page Hinkley has this name because of Page-Hinkley Test (PHT). It necessitates a user-defined input parameter and threshold based on the characteristics of the data and the goals of the application. The two-sided PHT identifies both upward and downward shifts in the mean of a sequence. For real-time change detection, it conducts two parallel tests, utilizing a cumulative variable that represents the total difference between the observed values and their mean up to the current point. There is no warning zone, only a drift zone [21].

EDDM (Early Drift Detection Method) seeks to enhance the detection of gradual concept drift in DDM while maintaining effectiveness against abrupt drifts. It achieves this by monitoring the average distance between errors rather than just the error rate. It requires tracking the running average distance, the running standard deviation, the maximum distance, and the maximum standard deviation. Considering i the stream sample index number, it works with the running average error distance (p'_i) and the running standard deviation (s'_i), as well as p'_{max} and s'_{max}, which are the values of p'_i and s'_i when $(p'_i + 2 * s'_i)$ reaches its maximum.

$$\text{if } \frac{p'_i + 2 * s'_i}{p'_{max} + 2 * s'_{max}} < \alpha \rightarrow \text{Warning zone} \tag{4}$$

$$\text{if } \frac{p'_i + 2 * s'_i}{p'_{max} + 2 * s'_{max}} < \beta \rightarrow \text{Change detected} \tag{5}$$

α and β are set to 0.95 and 0.9, respectively [22].

FHDDM (Fast Hoeffding Drift Detection Method) operates by monitoring a model's error rate over time and dividing the data into small fixed-size windows, calculating the error rate for each window. It applies Hoeffding's inequality [25] to compute an upper bound on the difference between the current error rate and the expected error rate. If this difference exceeds the bound, it signals a concept drift, indicating that the model may need updating or retraining [23].

HDDM_M and HDDM_W are similar methods. The first one identifies drifts by comparing moving averages. The second one applies the a forgetting scheme to weight the moving averages before comparing them to detect drifts. In both methods Hoeffding's inequality [25] is used to establish an upper limit on the difference between the averages. The authors observed that the first method is well-suited for detecting abrupt drifts, while the second method is better for detecting gradual drifts [24][23].

4. Experimental protocol

All tests were conducted using River API. River is a free open-source software under the BSD 3-Clause Revised License[1]. With River developers can make online machine learning in Python, specially under data streams.

The used datasets were extracted from three different sources:

- River API[2]: Airlines, TrumpApproval, Bananas, Occupancy, CreditCard, TREC07, SolarFlare, SMTP, STAGGER, SEA and Anomaly Sine.
- USP Streaming Dataset Repository[3]: Chess, InsectsAbruptBalanced, InsectsAbruptImbalanced, InsectsGradualBalanced, InsectsGradualImbalanced and NOAA.
- UCI Machine Learning Repository[4]: Electricity, Phishing, ForestCoverType and Ozone.

The procedure consisted of testing a pair categories of streaming datasets: The ones that appear not to have concept drift according to previous literature and the ones that do appear.

Nine models were tested for every dataset. The baseline model is the Hoeffding Tree Classifier without any trigger. All others were associated to one of the following drift detectors: ADWIN, DDM, EDDM, FHDDM, HDDM_A, HDDM_W, KSWIN or PageHinkley.

Only datasets suitable for classification were selected. Some of them demanded multiclass while others demanded binary classification.

The used protocol to verify if models are adequate to classify the stream is to compare the Hoeffding Tree Classifier's performance triggered by detecting algorithms with the model without any detector. If the best accuracy is given by the classifier without any detection trigger, the result indicates that the detectors don't improve the model but deteriorate it. If the model is enhanced instead, the experiment suggests that the detection algorithm is useful for this purpose.

5. Results

The results were divided into two tables. The datasets that seem to have drift are positioned on Table 1, and the ones that seem not to have are located on Table 2.

The Table 1 shows that the model without any drift detector trigger is in general worst than the ones with triggers. Although there a single case that the none trigger model is tied in accuracy with EDDM as the best model. This data indicates that no trigger may be satisfying for modeling a streaming classifier with this dataset, or the stream may not actually contain any drift. In all other cases the triggerless model was not among the best ones.

On the contrary of all other drift detector algorithms, HDDM_A and HDDM_W have smaller accuracy means than the triggerless model (respectively 5,79% less and 6,16% approximately). Despite of this, HDDM_A generated the best accuracy in TrumpApproval dataset and HDDM_W in ForestCoverType. The model with the best mean accuracy KSWIN was approximately 4,34% better than the triggerless model mean accuracy.

The results of the Table 2 have contrariwise the none detector as the best classifying model. All triggers deteriorated the models performances or in best cases didn't affect them at all. This probably happened simply because they didn't point any drift. In Credit Card Dataset the improving obtained with DDM was so small that it was considered not significant. Thus, the dataset were kept in the driftless group.

6. Conclusion

About the obtained results of the datasets that apparently have concept drift, although drift detecting

[1] https://github.com/online-ml/river/blob/main/LICENSE
[2] https://riverml.xyz/latest/
[3] https://sites.google.com/view/uspdsrepository
[4] https://archive.ics.uci.edu/

Dataset	None	ADWIN	KSWIN	PageHinkley	DDM	EDDM	FHDDM	HDDM_A	HDDM_W
AnomalySine	89,04%	79,14%	96,32%	95,01%	98,83%	90,06%	98,63%	98,60%	79,81%
CDS	94,74%	96,91%	96,67%	95,58%	96,67%	94,65%	96,37%	93,36%	94,08%
SEA	94,21%	97,43%	94,92%	95,11%	97,79%	97,79%	95,08%	95,74%	94,50%
Airlines	60,97%	63,02%	60,83%	62,66%	63,66%	61,76%	62,10%	61,65%	60,27%
Chess	66,20%	69,98%	67,99%	65,61%	66,20%	66,20%	66,20%	66,20%	68,79%
TrumpApproval	17,98%	31,87%	37,36%	37,56%	35,95%	37,98%	32,77%	43,96%	42,86%
Electricity	79,56%	82,43%	83,23%	80,59%	81,89%	79,56%	83,25%	76,17%	76,17%
InsAbruptBalanced	51,69%	64,10%	61,73%	50,53%	55,73%	51,69%	51,46%	28,88%	32,28%
InsAbruptImbalanced	64,48%	61,64%	56,62%	45,84%	63,75%	64,48%	47,90%	28,78%	32,35%
InsGradualBalanced	58,62%	65,86%	66,56%	55,94%	58,62%	58,62%	56,31%	36,69%	38,59%
InsGradualImbalanced	56,34%	60,37%	58,77%	48,21%	60,01%	56,34%	48,23%	30,80%	34,97%
ForestCoverType	80,15%	81,96%	85,04%	82,16%	81,36%	82,81%	79,99%	83,60%	85,43%
Mean	**67,83%**	**71,23%**	**72,17%**	**67,90%**	**71,70%**	**70,16%**	**68,19%**	**62,04%**	**61,67%**

Figura 1: Seemingly drifted streams models accuracy.

Dataset	None	ADWIN	KSWIN	PageHinkley	DDM	EDDM	FHDDM	HDDM_A	HDDM_W
Bananas	64,21%	61,72%	61,68%	59,38%	60,83%	59,89%	64,21%	62,87%	59,53%
Occupancy	98,96%	97,90%	98,51%	97,95%	97,66%	96,53%	97,28%	97,88%	97,89%
Phishing	87,92%	85,76%	86,64%	86,48%	86,32%	85,28%	87,92%	87,76%	85,76%
CreditCard	99,83%	99,83%	99,83%	99,83%	99,85%	99,82%	99,83%	99,83%	99,82%
TREC07	81,48%	66,51%	67,06%	62,78%	61,35%	81,48%	66,01%	46,13%	46,13%
SolarFlare	99,44%	99,44%	99,44%	99,44%	99,25%	99,44%	99,44%	99,44%	99,44%
SMTP	99,97%	99,97%	99,97%	99,97%	99,96%	99,97%	99,97%	99,96%	99,95%
NOAA	73,55%	71,40%	70,86%	70,48%	72,02%	71,83%	70,04%	70,89%	72,94%
Luxemburg	97,48%	97,48%	97,48%	97,48%	97,48%	82,48%	97,48%	97,48%	97,48%
Ozone	93,64%	93,49%	93,53%	93,56%	93,53%	93,64%	93,60%	93,53%	93,37%
Mean	**89,65%**	**87,35%**	**87,50%**	**86,73%**	**86,82%**	**87,04%**	**87,58%**	**85,58%**	**85,23%**

Figura 2: Seemingly non-drifted streams models accuracy.

models didn't create enormous changes in models performances, they all proved proved themselves relatively good in one case at least. This drives to the conclusion that none of them could be considered useless handling with concept drift. In most cases they all surpassed the models without detectors.

Another thing that is very clear is that the datasets that seem not to have concept drift responded better to the models without drift detectors. On the other hand, the datasets that seemingly contain concept drift responded generally with the worst accuracy when didn't triggered by a concept drift detector algorithm. Despite the InsectsGradualBalanced dataset couldn't be modeled with better accuracy than the triggerless one, it doesn't change the general conclusion.

Due to the big difference of accuracy seen in the different datasets, without a clear and absolute superiority of any of them, a recommendation for future applications is to test various drift detection algorithms. Even the two detectors with the far worst accuracy means (HDDM_A and HDDM_W) figured as the best detector in at least one stream. In this way, all detectors proved themselves somehow useful.

An aspect that became clear very clear with this research is that due to the fact that one cannot be sure if a dataset has a drift or not, there is no guarantee that drift detector algorithms will improve the models performances. Thus, if it is decided to use streaming drift handler models, it is fundamental to have a baseline that considers the stream as non drifted. In this way, the effects of drift detectors will be clearer and performance worsening can be spotted.

7. Acknowledgments

I would like to extend my gratitude to Professor Paulo Almeida for his unwavering support and dedication throughout the journey of my specialization thesis. To my wife, Renata, her belief in me and constant encouragement have been the driving force behind my academic pursuits. Lastly, to my dear parents, whose examples have instilled in me the foundation of my academic journey.

Referências

[1] Almeida, Paulo RL, et al. "Adapting dynamic classifier selection for concept drift."Expert Systems with Applications 104 (2018): 67-85.

[2] Gama, Joao. Knowledge discovery from data streams. CRC Press, 2010.

[3] Gama, João, et al. "A survey on concept drift adaptation."ACM computing surveys (CSUR) 46.4 (2014): 1-37.

[4] Hoens, T. Ryan, Robi Polikar, and Nitesh V. Chawla. "Learning from streaming data with concept drift and imbalance: an overview."Progress in Artificial Intelligence 1 (2012): 89-101.

[5] Krawczyk, Bartosz, et al. "Ensemble learning for data stream analysis: A survey."Information Fusion 37 (2017): 132-156.

[6] Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression. John Wiley & Sons.

[7] Agrahari, Supriya, and Anil Kumar Singh. "Concept drift detection in data stream mining: A literature review."Journal of King Saud University-Computer and Information Sciences 34.10 (2022): 9523-9540.

[8] Lu, Jie, et al. "Learning under concept drift: A review."IEEE transactions on knowledge and data engineering 31.12 (2018): 2346-2363.

[9] Ditzler, Gregory, and Robi Polikar. "Incremental learning of concept drift from streaming imbalanced data."IEEE transactions on knowledge and data engineering 25.10 (2012): 2283-2301.

[10] Demšar, Jaka, and Zoran Bosnić. "Detecting concept drift in data streams using model explanation."Expert Systems with Applications 92 (2018): 546-559.

[11] Wang, Shenghui, Stefan Schlobach, and Michel Klein. "What is concept drift and how to measure it?."Knowledge Engineering and Management by the Masses: 17th International Conference, EKAW 2010, Lisbon, Portugal, October 11-15, 2010. Proceedings 17. Springer Berlin Heidelberg, 2010.

[12] Webb, Geoffrey I., et al. "Characterizing concept drift."Data Mining and Knowledge Discovery 30.4 (2016): 964-994.

[13] Gonçalves Jr, Paulo M., et al. "A comparative study on concept drift detectors."Expert Systems with Applications 41.18 (2014): 8144-8156.

[14] Minku, Leandro L., Allan P. White, and Xin Yao. "The impact of diversity on online ensemble learning in the presence of concept drift."IEEE Transactions on knowledge and Data Engineering 22.5 (2009): 730-742.

[15] Kolter, J. Zico, and Marcus A. Maloof. "Dynamic weighted majority: An ensemble method for drifting concepts."The Journal of Machine Learning Research 8 (2007): 2755-2790.

[16] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), 1–37. doi:10.1145/2523813.

[17] WEINBERG, Abraham Itzhak; LAST, Mark. EnHAT—Synergy of a tree-based Ensemble with Hoeffding Adaptive Tree for dynamic data streams mining. Information Fusion, v. 89, p. 397-404, 2023.

[18] GAMA, Joao et al. Learning with drift detection. In: Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17. Springer Berlin Heidelberg, 2004. p. 286-295.

[19] BIFET, Albert; GAVALDA, Ricard. Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2007. p. 443-448.

[20] RAAB, Christoph; HEUSINGER, Moritz; SCHLEIF, Frank-Michael. Reactive soft prototype computing for concept drift streams. Neurocomputing, v. 416, p. 340-351, 2020.

[21] SEBASTIÃO, Raquel; FERNANDES, José Maria. Supporting the page-hinkley test with empirical mode decomposition for change detection. In: International Symposium on Methodologies for Intelligent Systems. Cham: Springer International Publishing, 2017. p. 492-498.

[22] BAENA-GARCIA, Manuel et al. Early drift detection method. In: Fourth international workshop on knowledge discovery from data streams. 2006. p. 77-86.

[23] PESARANGHADER, Ali; VIKTOR, Herna L. Fast hoeffding drift detection method for evolving data streams. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16. Springer International Publishing, 2016. p. 96-111.

[24] FRIAS-BLANCO, Isvani et al. Online and non-parametric drift detection methods based on Hoeffding's bounds. IEEE Transactions on Knowledge and Data Engineering, v. 27, n. 3, p. 810-823, 2014.

[25] HOEFFDING, Wassily. Probability inequalities for sums of bounded random variables. Wiley StatsRef: Statistics Reference Online, 2014.