Current Trends in Learning from Data Streams

João Gama[1,3(✉)], Bruno Veloso[1,2,3], Ehsan Aminian[1,3], and Rita P. Ribeiro[1,3]

[1] LIAAD - INESC TEC, Porto, Portugal
jgama@inesctec.pt
[2] University Portucalense, Porto, Portugal
[3] University of Porto, Porto, Portugal

Abstract. This article presents our recent work on the topic of learning from data streams. We focus on emerging topics, including fraud detection, learning from rare cases, and hyper-parameter tuning for streaming data.

Keywords: Data streams · Fraud detection · Rare cases · Hyper-parameter tuning

1 Introduction

Learning from data streams is an hot-topic in machine learning and data mining. This article presents our recent work on the topic of learning from data streams. It is organized into three main sections. The first one, push up the need of forgetting older data to reinforce the focus on the most recent data. It is based on the work presented in [8]. The second topic refers to the problem of learning from imbalanced regression streams. In this setting, we are interested in predicting points in the fringe of the distribution. It is based on the work presented in [2]. The third topic, discuss the topic of hyper-parameter tuning in the context of data stream mining. It is based on the work presented in [7].

2 The Importance of Forgetting

The high asymmetry of international termination rates with regard to domestic ones, where international calls have higher charges applied by the operator where the call terminates, is fertile ground for the appearance of fraud in Telecommunications. There are several types of fraud that exploit this type of differential, being the Interconnect Bypass Fraud one of the most expressive [1,6].

In this type of fraud, one of several intermediaries responsible for delivering the calls forwards the traffic over a low-cost IP connection, reintroducing the call in the destination network already as a local call, using VOIP Gateways. This way, the entity that sent the traffic is charged the amount corresponding to the delivery of international traffic. However, once it is illegally delivered as national traffic, it will not have to pay the international termination fee, appropriating this amount.

S. N. Srirama et al. (Eds.): BDA 2021, LNCS 13147, pp. 183–193, 2021.
https://doi.org/10.1007/978-3-030-93620-4_14

Traditionally, the telecom operators analyze the calls of these Gateways to detect the fraud patterns and, once identified, have their SIM cards blocked. The constant evolution in terms of technology adopted on these gateways allows them to work like real SIM farms capable of manipulating identifiers, simulating standard call patterns similar to the ones of regular users, and even being mounted on vehicles to complicate the detection using location information.

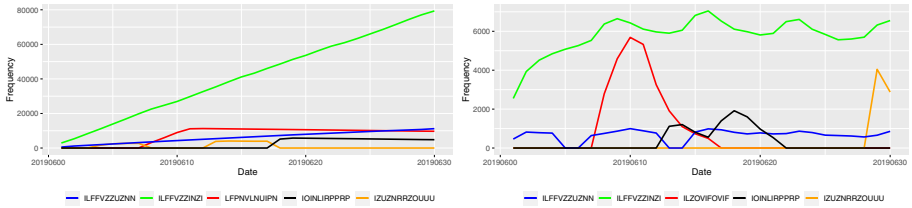
The interconnect bypass fraud detection algorithms typically consume a stream S of events, where S contains information about the origin number $A - Number$, the destination number $B - Number$, the associated timestamp, and the status of the call (accomplished or not). The expected output of this type of algorithm is a set of potential fraudulent $A - Numbers$ that require validation by the telecom operator. This process is not fully automated to avoid block legit $A - Numbers$ and get penalties. In the interconnect bypass fraud, we can observe three different types of abnormal behaviors: (i) the burst of calls, which are $A - Numbers$ that produce enormous quantities of $\#calls$ (above the $\overline{\#calls}$ of all $A - Numbers$) during a specific time window W. The size of this time window is typically small; (ii) the repetitions, which are the repetition of some pattern ($\#calls$) produced by a $A - Number$ during consecutive time windows W; and (iii) the mirror behaviors, which are two distinct $A - Numbers$ (typically these $A - Numbers$ are from the same country) that produces the same pattern of calls ($\#calls$) during a time window W.

Algorithm 1. The `Lossy Counting` Algorithm.

```
1: procedure LossyCounting(S: A Sequence of Examples; ε: Error margin; α: fast forgetting parameter)
2:     n ← 0; Δ ← 0; T ← 0;
3:     for example e ∈ S do
4:         n ← n + 1
5:         if e is monitored then
6:             Increment Count_e
7:         else
8:             T ← T ∪ {e, 1 + Δ}
9:         end if
10:        if ⌈n/ε⌉ ≠ Δ then
11:            Δ ← n/ε
12:        end if
13:        for all j ∈ T do
14:            if Count_j < δ then
15:                T ← T\{j}
16:            end if
17:        end for
18:    end for
19: end procedure
```

Algorithm 2. The `Lossy Counting` with Fast Forgetting Algorithm.

```
1: procedure LossyCounting(S: A Sequence of Examples; ε: Error margin; α: fast forgetting parameter)
2:     n ← 0; Δ ← 0; T ← 0;
3:     for example e ∈ S do
4:         n ← n + 1
5:         if e is monitored then
6:             Increment Count_e
7:         else
8:             T ← T ∪ {e, 1 + Δ}
9:         end if
10:        if ⌈n/ε⌉ ≠ Δ then
11:            Δ ← n/ε
12:        end if
13:        for all j ∈ T do
14:            Count_j ← α * Count_j
15:            if Count_j < δ then
16:                T ← T\{j}
17:            end if
18:        end for
19:    end for
20: end procedure
```



(a) Lossy Counting (b) Lossy Counting with Fast Forgetting

Fig. 1. Fraud detection on number of calls

3 Learning Rare Cases

Few approaches in the area of learning from imbalanced data streams discuss the task of regression. In this study, we employ the Chebyshev's inequality as an heuristic to disclose the type of incoming cases (i.e. frequent or rare). We discuss two methods for learning regression models from imbalanced data streams [2]. Both methods use Chebyshev's inequality to train learning models over a relatively balanced data stream once the incoming data stream is imbalanced. The mentioned inequality derived from Markov inequality is used to bound the tail probabilities of a random variable like Y. It guarantees that in any probability distribution, 'nearly all' values are close to the mean. More precisely, no more than $\frac{1}{t^2}$ of the distribution's values can be more than t times standard deviations away from the mean. Although conservative, the inequality can be applied to completely arbitrary distributions (unknown except for mean and variance). Let Y be a random variable with finite expected value $\overline{y}$ and finite non-zero variance σ^2. Then for any real number $t > 0$, we have:

$$\Pr(|y - \overline{y}| \geq t\sigma) \leq \frac{1}{t^2} \tag{1}$$

Only the case $t > 1$ is useful in the above inequality. In cases $t < 1$, the right-hand side is greater than one, and thus the statement will be "always true" as the probability of any event cannot be greater than one. Another "always true" case of inequality is when $t = 1$. In this case, the inequality changes to a statement saying that the probability of something is less than or equal to one, which is "always true".

For $t = \frac{|y-\overline{y}|}{\sigma}$ and $t > 1$, we define *frequency* score of the observation $\langle x, y \rangle$ as:

$$P(\mid \overline{y} - y \mid \geq t) = \frac{1}{\left(\frac{|y-\overline{y}|}{\sigma}\right)^2} \tag{2}$$

The above definition states that the probability of observing y far from its mean is small, and it decreases as we get farther away from the mean. In an imbalanced data streams regression scenario, considering the mean of target

values of the examples in the data stream ($\overline{y}$), examples with rare extreme target values are more likely to occur far from the mean. In contrast, examples with frequent target values are closer to the mean. So, given the mean and variance of a random variable, Chebyshev's inequality can indicate the degree of the rarity of an observation such that its low and high value implies that the observation is most probably a rare or a frequent case, respectively.

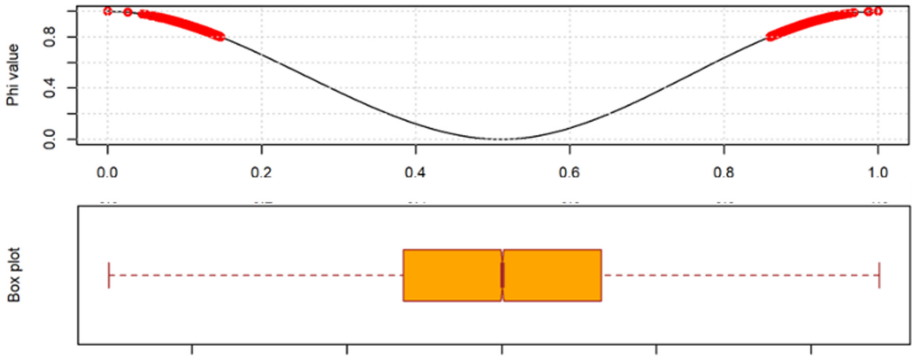


Fig. 2. Data-points relevance and the box plot for the target variable of Fried data set.

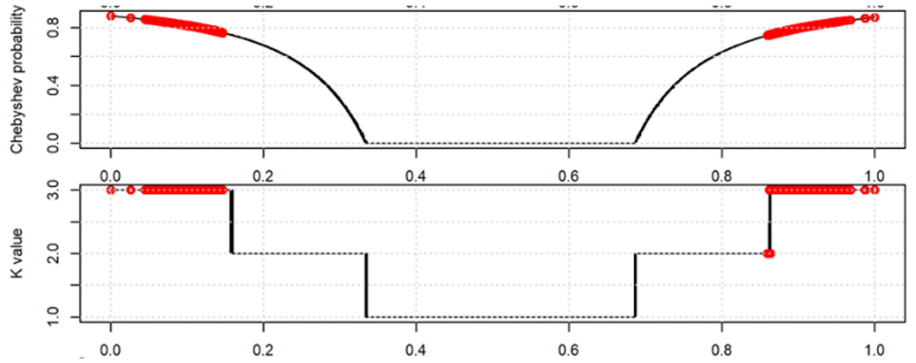


Fig. 3. Chebyshev probability used by the under-sample approach (top panel), and the K-value used in the over sample approach (bottom panel) for the target variable of Fried data set.

Figure 3 shows the probability values calculated from Eq. 2 for Fried data set, described in [5], along with the box plot of the target variable. As can be seen from the figures and as we expected, Chebyshev's probability value for examples near the mean is close to one. It decreases as we get far from the mean until it gets close to zero, for example, at the farthest distance to the mean. Accordingly,

interpretation of the output value of Eq. 2 for an example as its frequency score makes sense. Moreover, it meets the imbalance regression problem definition, w.r.t. rare extreme values of the target variable.

Having equipped with the heuristic to discover if an example is rare or frequent, the next step is to use such knowledge in training a regression model.To do that, ChebyUS and ChebyOS are the two methods proposed in this paper. They are described in detail in the next subsections.

3.1 ChebyUS: Chebyshev-Based Under-Sampling

The proposed under-sampling method is presented in Algorithm 3. This algorithm selects an incoming example for training the model if a randomly generated number in $[0, 1]$ is greater or equal to its Chebyshev's probability which is calculated as:

$$P(\mid y - \overline{y} \mid \geq t) = \begin{cases} \frac{\sigma^2}{|y-\overline{y}|^2}, & t > 1 \\ 1, & t \leq 1 \end{cases} \tag{3}$$

If the example is not selected, it is assumed that the example is probably a frequent case. Still, it receives a second chance for being selected if the number of frequent cases selected so far is less than that of rare cases. If so, the example is selected with a second chance probability (input parameter `sp`).

The descriptive statistics (μ and σ^2) of the target variable of examples can be computed through incremental methods [4]. The greater the number of examples, n, we have, the more accurate the estimation will be. For the first examples, mean and variance are not accurate, and therefore, Chebyshev's probability will not be accurate enough. But as more examples are received, those statistics (i.e. mean and variance) and, consequently, Chebyshev's probability are getting more stable and accurate.

At the end of the model's training phase, we expect the model to have been trained over approximately the same portion of frequent and rare cases.

3.2 ChebyOS: Chebyshev-Based Over-Sampling

Another way of making a balanced data stream is to over-sample rare cases of the incoming imbalanced data stream. Since those rare cases in data streams can be discovered by their Chebyshev's probability, they can be easily over-sampled by replication. Algorithm 4 describes our over-sampling proposed method.

For each example, a t value can be calculated by Eq. 4 which yields the result in $[0 \quad +\infty)$.

$$t = \frac{|y - \overline{y}|}{\sigma} \tag{4}$$

While t value is small for examples near the mean, it would be larger for ones farther from the mean and has its largest value for examples located in the farthest distance to the mean (i.e. extreme values). We limit the function in Eq. 4 to produce only natural numbers as follows:

Algorithm 3. ChebyUS: Chebyshev-based Under-Sampling Algorithm

```
1: procedure ChebyUS(S: Data stream, sp: second chance probability)
2:     H ← CreateEmptyModel()
3:     i ← 0
4:     RCounter ← 0                                   ▷ rare cases counter
5:     FCounter ← 0                                   ▷ frequent cases counter
6:     while true do
7:         ⟨x, y⟩ ← GetExample()
8:         ȳ, σ ← UpdateStatistics(y)
9:         p ← ComputeProbability(y, ȳ, σ)            ▷ Equation 3
10:        if RandomNumber ≥ p then
11:            H ← TrainModel(H, ⟨x, y⟩)
12:            RCounter ← RCounter + 1
13:        else
14:            if FCounter ≤ RCounterANDRandomNumber ≤ sp then
15:                H ← TrainModel(H, ⟨x, y⟩)
16:                FCounter ← FCounter + 1
17:            end if
18:        end if
19:        i ← i + 1
20:    end while
21: end procedure
```

Algorithm 4. ChebyOS: Chebyshev-based Over-Sampling Algorithm

```
1: procedure ChebyOS(S: Data stream)
2:     H ← CreateEmptyModel()
3:     i ← 0
4:     while true do
5:         ⟨x, y⟩ ← GetExample()
6:         ȳ, σ ← UpdateStatistics(y)
7:         k ← ComputeK(y, ȳ, σ)                      ▷ Equation 5
8:         for i ← 1 to k do
9:             H ← TrainModel(H, ⟨x, y⟩)
10:        end for
11:        i ← i + 1
12:    end while
13: end procedure
```

$$K = \left\lceil \frac{|y - \overline{y}|}{\sigma} \right\rceil \tag{5}$$

K is expected to have greater numbers for rare cases. In our proposed over-sampling method, we use K value computed for each incoming example and present that example exactly K times to the learner.

Examples that are not as far from the mean as the variance, are most probably frequent cases. They contribute only once in the learner's training process while the others contribute more times.

3.3 Experimental Evaluation

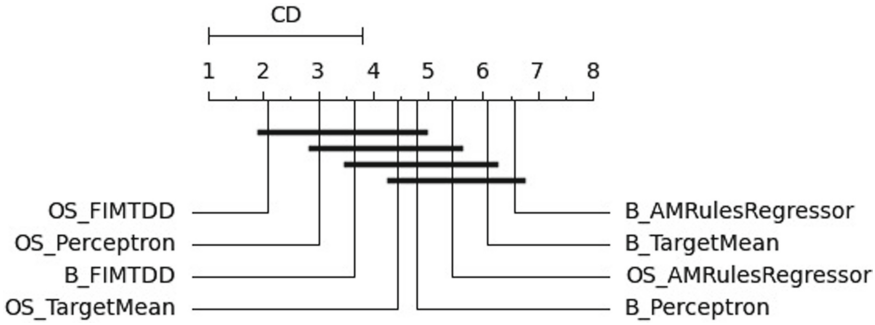


Fig. 4. Critical Difference diagrams considering both extreme rare cases ($thr_\phi = 0.8$), for four regression algorithms with no sampling (Baseline) and with the Chebyshev-based Over-Sampling (ChebyOS) strategy.

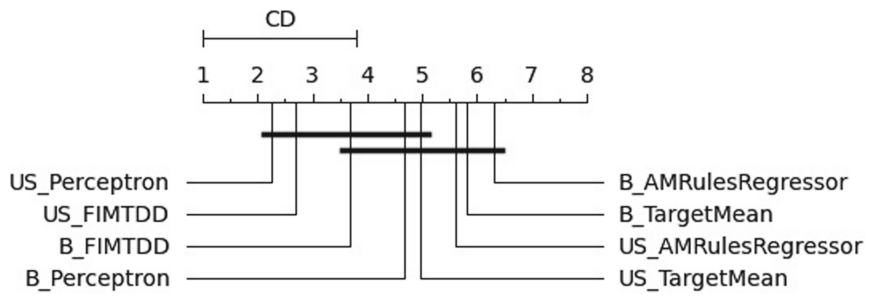


Fig. 5. Critical Difference diagrams considering both extreme rare cases ($thr_\phi = 0.8$), for four regression algorithms with no sampling (Baseline) and with the Chebyshev-based Under-Sampling (ChebyUS) strategy.

Figures 4 and 5 presents the critical difference diagrams [3] for four regression algorithms with no sampling (Baseline) and with the proposed sampling strategies: Chebyshev-based Under-Sampling (ChebyUS) andChebyshev-based Over-Sampling (ChebyOS).

4 Learning to Learn: Hyperparameter Tunning

This algorithm is a simplex search algorithm for multidimensional unconstrained optimization without derivatives. The vertexes of the simplex, which define a convex hull shape, are iteratively updated in order to sequentially discard the vertex associated with the largest cost function value.

The Nelder-Mead algorithm relies on four simple operations: *reflection*, *shrinkage*, *contraction* and *expansion*. Figure 6 illustrates the four corresponding Nelder-Mead operators R, S, C and E. Each vertice represents a model containing a set of hyper-parameters. The vertexes (models under optimisation) are ordered and named according to the root mean square error (RMSE) value: best (B), good (G), which is the closest to the best vertex, and worst (W). M is a mid vertex (auxiliary model). Algorithms 5 and 6 describe the application of the four operators.

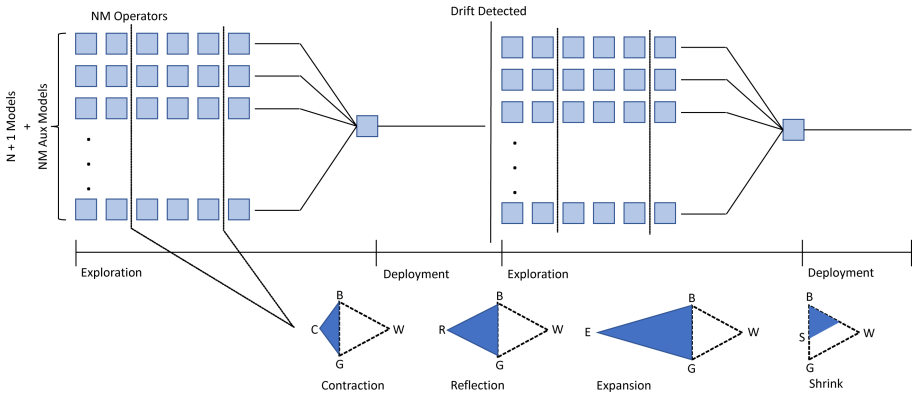


Fig. 6. SPT working modes and Nelder & Mead operators.

Algorithm 5 presents the reflection and extension of a vertex and Algorithm 6 presents the contraction and shrinkage of a vertex. For each Nelder-Mead operation, it is necessary to compute an additional set of vertexes (midpoint M, reflection R, expansion E, contraction C and shrinkage S) and verify if the calculated vertexes belong to the search space. First, Algorithm 5 computes the midpoint (M) of the best face of the shape as well as the reflection point (R). After this initial step, it determines whether to reflect or expand based on the set of predetermined heuristics (lines 3, 4 and 8).

Algorithm 5. Nelder-Mead - reflect (a) or expand operators (d).

```
1: M = (B + G)/2
2: R = 2M − W
3: if f(R) < f(G) then
4:     if f(B) < f(R) then
5:         W = R
6:     else
7:         E = 2R − M
8:         if f(E) < f(B) then
9:             W = E
10:        else
11:            W = R
12:        end if
13:    end if
14: end if
```

Algorithm 6. Nelder-Mead - contract (c) or shrink (b) operators.

```
1: M = (B + G)/2
2: R = 2M − W
3: if f(R) ≥ f(G) then
4:     if f(R) < f(W) then
5:         W = R
6:     else
7:         C = (W + M)/2
8:         if f(C) < f(W) then
9:             W = C
10:        else
11:            S = (B + W)/2
12:            if f(S) < f(W) then
13:                W = S
14:            end if
15:            if f(M) < f(G) then
16:                G = M
17:            end if
18:        end if
19:    end if
20: end if
```

Algorithm 6 calculates the contraction point (C) of the worst face of the shape – the midpoint between the worst vertex (W) and the midpoint M – and shrinkage point (S) – the midpoint between the best (B) and the worst (W) vertexes. Then, it determines whether to contract or shrink based on the set of predetermined heuristics (lines 3, 4, 8, 12 and 15).

The goal, in the case of data stream regression, is to optimise the learning rate, the learning rate decay and the split confidence hyper-parameters. These hyper-parameters are constrained to values between 0 and 1. The violation of this constraint results in the adoption of the nearest lower or upper bound.

4.1 Dynamic Sample Size

The dynamic sample size, which is based on the RMSE metric, attempts to identify significant changes in the streamed data. Whenever such a change is detected, the Nelder-Mead compares the performance of the $n+1$ models under analysis to choose the most promising model. The sample size S_{size} is given by Eq. 6 where σ represents the standard deviation of the RMSE and M the desired error margin. We use $M = 95\%$.

$$S_{size} = \frac{4\sigma^2}{M^2} \tag{6}$$

However, to avoid using small samples, that imply error estimations with large variance, we defined a lower bound of 30 samples.

4.2 Stream-Based Implementation

The adaptation of the Nelder-Mead algorithm to on-line scenarios relies extensively on parallel processing. The main thread launches the $n+1$ model threads

and starts a continuous event processing loop. This loop dispatches the incoming events to the model threads and, whenever it reaches the sample size interval, assesses the running models and calculates the new sample size. The model assessment involves the ordering of the $n + 1$ models by RMSE value and the application of the Nelder-Mead algorithm to substitute the worst model. The Nelder-Mead parallel implementation creates a dedicated thread per Nelder-Mead operator, totalling seven threads. Each Nelder-Mead operator thread generates a new model and calculates the incremental RMSE using the instances of the last sample size interval. The worst model is substituted by the Nelder-Mead operator thread model with lowest RMSE.

4.3 Experimental Evaluation

Figure 7 presents the critical difference diagram [3] of three hyper-parameter tuning algorithms: SPT, Grid search, default parameter values on four benchmark classification datasets. The diagram clearly illustrate the good performance of SPT.

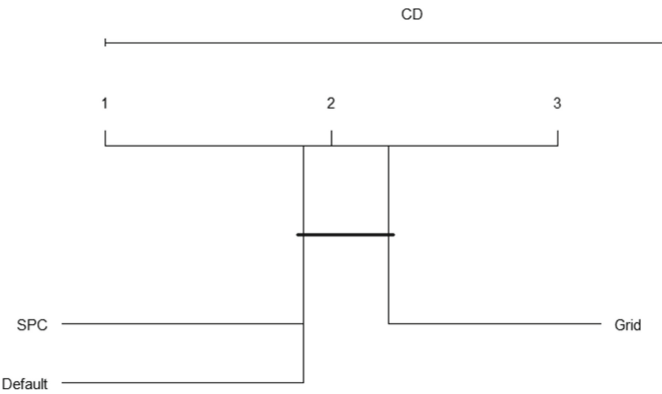


Fig. 7. Critical Difference Diagram comparing Self hyper-parameter tuning, Grid hyper-parameter tuning, and default parameters in 4 classification problems.

5 Conclusions

This paper reviews our recent work in learning from data streams. The two first works present different approaches to deal with imbalanced data: from applied research in fraud detection to basic research on using Chebyshev inequality to guide under-sampling and over-sampling. The last work presents a streaming optimization method to find the minimum of a function and its application in finding the hyper-parameter values that minimize the error. We believe that the three works reported here will have an impact on the work of other researchers.

Acknowledgements. This work was supported by the CHIST-ERA grant CHIST-ERA-19-XAI-012, and project CHIST-ERA/0004/2019 funded by FCT.

References

1. Ali, M., Azad, M., Centeno, M., Hao, F., Van Moorsel, A.: Consumer-facing technology fraud: economics, attack methods and potential solutions. Future Gener. Comput. Syst. **100**, 408–427 (2019)
2. Aminian, E., Ribeiro, R.P., Gama, J.: Chebyshev approaches for imbalanced data streams regression models. Data Mining Knowledge Discovery, pp. 1–78 (2021). https://doi.org/10.1007/s10618-021-00793-1
3. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
4. Finch, T.: Incremental calculation of weighted mean and variance. Technical Report, University of Cambridge Computing Service, Cambridge, UK (2009)
5. Friedman, J.: Multivariate adaptive regression splines. Ann. Statist. **19**(1), 1–67 (1991)
6. Laleh, N., Abdollahi Azgomi, M.: A taxonomy of frauds and fraud detection techniques. In: Prasad, S.K., Routray, S., Khurana, R., Sahni, S. (eds.) Information Systems, Technology and Management. ICISTM 2009. Communications in Computer and Information Science, vol. 31. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00405-6_28
7. Veloso, B., Gama, J., Malheiro, B., Vinagre, J.: Hyperparameter self-tuning for data streams. Inf. Fusion **76**, 75–86 (2021)
8. Veloso, B., Tabassum, S., Martins, C., Espanha, R., Azevedo, R., Gama, J.: Interconnect bypass fraud detection: a case study. Ann. Telecommun., **75**(9), 583–596 (2020). https://doi.org/10.1007/s12243-020-00808-w