24th International Conference on Business Information Systems (BIS 2021)

Big Data

https://doi.org/10.52825/bis.v1i.45

Published: 02 July 2021

# Predict COVID-19 Spreading with C-SMOTE

Alessio Bernardo[1][https://orcid.org/0000-0002-3492-0345], Emanuele Della Valle[1][https://orcid.org/0000-0002-5176-5885]

[1]DEIB, Politecnico di Milano, Italy

**Abstract.** Data continuously gathered monitoring the spreading of the COVID-19 pandemic form an unbounded flow of data. Accurately forecasting if the infections will increase or decrease has a high impact, but it is challenging because the pandemic spreads and contracts periodically. Technically, the flow of data is said to be imbalanced and subject to concept drifts because signs of decrements are the minority class during the spreading periods, while they become the majority class in the contraction periods and the other way round. In this paper, we propose a case study applying the Continuous Synthetic Minority Oversampling Technique (C-SMOTE), a novel meta-strategy to pipeline with Streaming Machine Learning (SML) classification algorithms, to forecast the COVID-19 pandemic trend. Benchmarking SML pipelines that use C-SMOTE against state-of-the-art methods on a COVID-19 dataset, we bring statistical evidence that models learned using C-SMOTE are better.

**Keywords:** SML, Evolving Data Stream, Concept Drift, Balancing, COVID-19

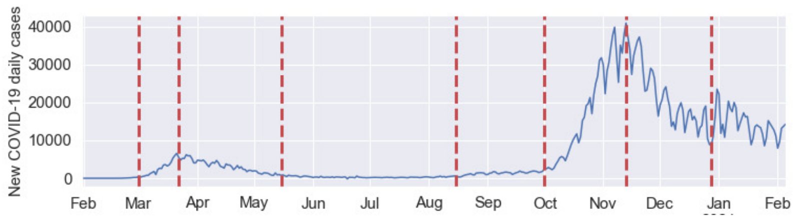
## Introduction

Nowadays, a multitude of smartphones, wearables, computers, and Internet of Things (IoT) sensors produce massive, continuous, and unbounded flows of data, namely *data streams*. They pose several challenges to Machine Learning (ML) since they are impossible to load as a whole in memory, and they are often non-stationary (i.e., they present concept drifts [1]). Moreover, when they are the input for a classification problem, they present class imbalance. This is the case of data streams related to the pandemic of COVID-19.

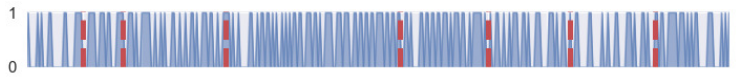
For instance, several ML models used for forecasting sales are failing during COVID-19 because people's behaviour keeps changing. Before the disease, a large part of the people spent all the day at the office, supermarkets, bars, restaurants while only a small part of the population (minority class) stayed at home. When COVID-19 spread, it was the opposite. The majority of the people stayed at home, and only a small percentage of them went outside (new minority class). During 2020 summer/autumn and 2021 winter, such a change in people's behaviour was often observed in many countries worldwide. Those sale forecasting solutions failed because they were unable to detect concept drifts and manage minority instances. Indeed, the traditional ML techniques are not designed to monitor concept drifts, so their models are prone to introduce classification errors when they happen.

In recent years, the *Streaming Approach* (a.k.a. *Data Stream Mining* or *Online Learning*) approach was introduced to tackle those problems. Streaming Machine Learning (SML) methods can detect when concept drifts occur and adapt the model accordingly.

This paper focuses on the COVID-19 case study, aiming at predicting the daily trend in the spreading of COVID-19. It is a *streaming binary classification problem* that requires addressing



**(a)** New daily COVID-19 cases.



**(b)** COVID-19 increment/decrement cases.

**Figure 1.** COVID-19 pandemic spreading in Italy. Red lines are concept drifts.

both concept drift and class imbalance [2]. Indeed, when the pandemic spreads, we may simply forecast an increment trend ignoring signs of decrement. On the over way round, when the pandemic contracts or is stable, we may ignore early signs of increment. Due to the concept drifts occurrences, classes can swap, i.e. all the samples labelled as minority (majority) class before a concept drift occurrence get labelled as majority (minority) class after it. Fig. 1 shows the COVID-19 spreading in Italy. In particular, Fig. 1a shows the number of new daily cases while Fig. 1b shows COVID-19 increment/decrement cases respect to the day before. *0* means that the COVID-19 cases decreased, while *1* means that the COVID-19 cases increased. The red lines represent the concept drifts occurrences[1]. Table 1 shows the percentage of COVID-19 increment/decrement cases in each concept drift. We can see that there is a continuous class swapping.

To address this problem, we applied our novel C-SMOTE [4] SML meta-strategy, inspired by SMOTE [5] class rebalancing algorithm. Considering that in literature there are SML algorithms *natively able* to rebalance streams in presence of concept drifts (let's denote them with SML+) and algorithms *unable* to do so (say, SML-), we formulated the following **research questions**:

Q1 *does prepending* C-SMOTE *to SML- algorithms improve their performances*?
Q2 *are there pipelines of* C-SMOTE *and a SML- algorithms that outperform SML+ models*?

In more detail, the *main contributions* of this paper are statistical evidence that, also in this particular case study, prepending C-SMOTE to SML- algorithms improves the minority class performances w.r.t. both the SML-'s and SML+'s performances, and, hence, better predicts the daily trend of COVID-19 spread.

The remainder of this paper is organized as follows. Section Sampling Techniques for Class Imbalance describes the investigated problem and presents techniques able to handle it. Section C-SMOTE describes the C-SMOTE meta-strategy. Section Related Work introduces the related works. Section Experimental Settings introduces the dataset, metrics, and algorithms used in the experiments on a COVID-19 dataset and presents our research hypotheses. Section Results and Discussion shows and discusses the evaluation results. Finally, Section Conclusions discusses the conclusions and outlines directions for future research.

**Table 1.** % of COVID-19 increment/decrement cases in each concept drift.

| **Case** | **CD1** | **CD2** | **CD3** | **CD4** | **CD5** | **CD6** | **CD7** | **CD8** |
|---|---|---|---|---|---|---|---|---|
| Increment (1) | 64.52% | 38.10% | 50.00% | 41.30% | 42.55% | 51.16% | 64.44% | 46.15% |
| Decrement (0) | 35.48% | 61.90% | 50.00% | 58.70% | 57.45% | 48.84% | 35.56% | 53.85% |

[1]The concept drifts occurrences are calculated using the ADWIN [3] strategy.

## Sampling Techniques for Class Imbalance

Imbalanced data are characterized by an unequal distribution between the classes. Since the minority class(es) instances rarely occur, the models only focus on patterns for correctly classifying the majority class(es) samples, so avoiding the ones for predicting the minority class(es) ones. The resulting problem is that the model will tend to predict all the samples as majority class ones, without really examining any of their features.

In the literature, there are four approaches for handling class imbalance [2]: sampling techniques, cost-sensitive learning, kernel-based methods, and active learning methods. In particular, sampling techniques allow changing the data distribution before the training phase. As a consequence, the algorithms can focus on cases that are more relevant to the user. For this reason, this work focuses on them: they allow a sort of meta-strategy development to prepend to any SML- model chosen by the user.

The most popular balancing technique is SMOTE [5]. For each minority class sample $s$, it synthetically generates new minority class points lying on the segments that join $s$ to any/all of its *k* minority class nearest neighbours. In general SMOTE has been shown to improve classification, but it may also show drawbacks related to the way it creates synthetic samples. Specifically, SMOTE introduces new samples without considering how the majority class points are distributed into the space region, and so risking to place the new points in a majority class region and to increase the overlapping between classes. To this end, more than a hundred SMOTE variants have been proposed [6] to overcome the overlapping between classes. The most well-known are Borderline-SMOTE [7], ADASYN [8], DBSMOTE [9], MDO [10], SWIM [11], and G-SMOTE [12].

However, SMOTE it is still considered the ”de-facto” balancing technique [6]. This is the reason why C-SMOTE is based on it. Moreover, as a sampling technique, SMOTE caches the entire dataset in memory. This approach is against the basic principles of the data stream paradigm that states that a sample can be inspected only once, as fast as possible, and then discarded. In the next section we explain how to overcome this problem.

## C-SMOTE

This section recalls the C-SMOTE description, inspired by SMOTE, originally presented in [4]. C-SMOTE is designed to rebalance an imbalanced data stream and, as Fig. 2 shows, it can be pipelined with any SML- technique. C-SMOTE stands for Continuous-SMOTE, meaning that the new SMOTE version is applied continuously. Its pseudocode is presented in Algorithm 1, while its implementation is available in the MOA GitHub repository[2].

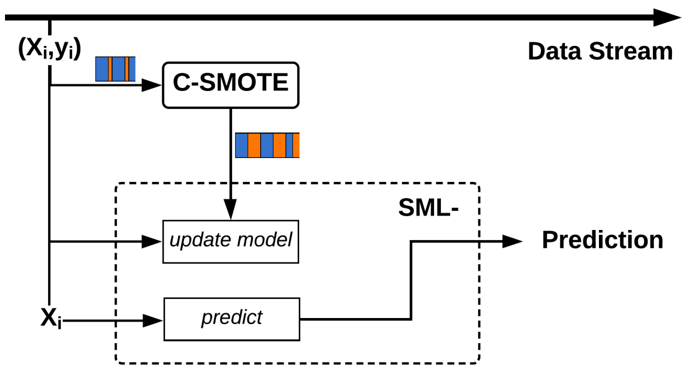


**Figure 2.** Architecture of SML pipelines using C-SMOTE meta-strategy.

[2]https://github.com/Waikato/moa

**Algorithm 1:**

```
1  Function C-SMOTE ($minSizeMinority, l, rebalanceThreshold, S$):
2      $W, W_{label} \leftarrow \varnothing$
3      $S_0, S_1, S_{\underline{0}}, S_{\underline{1}} \leftarrow 0$
4      $S_{\underline{gen}} \leftarrow \varnothing$
5      $imbalanceRatio \leftarrow 0$
6      $adwin \leftarrow \varnothing$
7      while $hasNext(S)$ do
8          $X, y \leftarrow next(S)$
9          $prequentialEvaluation(X, l)$
10         $train(X, y, l)$
11         $W \leftarrow add(X, y)$
12         $updateWindows(X, y, W_{label})$
13         $updateCounters(y, S_0, S_1)$
14         $adwin \leftarrow add(y)$
15         $checkConceptDrift(adwin, W, W_{label}, S_0, S_1, S_{\underline{0}}, S_{\underline{1}}, S_{\underline{gen}})$
16         $W_{min}, S_{min}, S_{\underline{min}} \leftarrow selectMinorityClass(W, W_{label}, S_0, S_1, S_{\underline{0}}, S_{\underline{1}})$
17         $W_{maj}, S_{maj}, S_{\underline{maj}} \leftarrow selectMajorityClass(W, W_{label}, S_0, S_1, S_{\underline{0}}, S_{\underline{1}})$
18         if $checkMinSize(minSizeMinority, S_{min})$ then
19             $imbalanceRatio \leftarrow ratio(S_{min}, S_{maj}, S_{\underline{min}}, S_{\underline{maj}})$
20             while $rebalanceThreshold > imbalanceRatio$ do
21                 $\hat{X}, \hat{y} \leftarrow newSample(W_{min}, S_{\underline{gen}})$
22                 $S_{\underline{min}} \leftarrow S_{\underline{min}} + 1$
23                 $train(\hat{X}, \hat{y}, l)$
24                 $imbalanceRatio \leftarrow ratio(S_{min}, S_{maj}, S_{\underline{min}}, S_{\underline{maj}})$
25             end
26         end
27     end
28 End Function
```

As said before, the real problem is the lack of the entire data during the rebalance phase. Moreover, it is impossible to store every new sample in memory until the data stream ends for two reasons: 1) the data streams are assumed infinite, and 2) this would be against the stream paradigm approach. The solution is to save the instances into a sliding window $W$ (Line 11) and use ADWIN [3] to keep in $W$ only the i) necessary recently-seen and ii) consistent to the current concept samples (Lines 14-15). ADWIN keeps a variable-length window of recently seen items, with the property that the window has the maximal length statistically consistent with the hypothesis "there has been no change in the average value inside the window" [3]. Then, after having found the real minority and majority classes (Lines 17-18) and the actual $imbalanceRatio$ between the number of minority, minority generated and majority instances stored in $W$ (Line 20), C-SMOTE is ready to rebalance the stream introducing new synthetically generated samples (Line 22). It uses the minority class samples stored in $W_{min}$ to apply an online version of SMOTE. In this way, this one is always applied to data that are consistent with the current concept, and the newly generated samples will be consistent with it, too. SMOTE, before starting to generate new instances, calculates the number of instances to introduce for each minority sample in the batch and then it starts to execute. Instead, C-SMOTE randomly chooses one sample $(X, y)$ from $W_{min}$ and uses it to apply SMOTE. Then, the function, in this rebalance phase, does not use any more $(X, y)$ to generate other synthetic instances. So, in our continuous version, not all the minority class instances are used to generate new instances. After that, at Line 24, the new instance $(\hat{X}, \hat{y})$ is used to train the learner $l$ and the new $imbalanceRatio$ is calculated.

## Related Work

To the best of our knowledge, in the literature exist only a few computing models to be adopted to predict the COVID-19 pandemic spreading [22]. Moreover, only a couple of them addresses the problem as a continuously evolving task [23], [24] without however referring to any concept drift detection or class imbalance. Instead, the methods we propose in this paper (i.e. SML+),

**Table 2.** The principal characteristics of the related works compared to C-SMOTE.

| Method | Classifier type | Approach type | Approach for CD | Approach for Class Imbalance |
|---|---|---|---|---|
| RLSACP [13] | Single | Passive | Forgetting factor | Cost weight |
| ONN [14] | Ensemble | Passive + Active module | Forgetting factor | Cost weight |
| ESOS-ELM [15] | Ensemble | Passive | Weighted ensemble | Cost weight |
| NN [16] | Ensemble | Passive | Weighted ensemble | Cost weight |
| OnlineUnderOverBagging [17] | Ensemble | Passive | Weighted ensemble | Undersampling + Oversampling |
| OnlineSMOTEBagging [17] | Ensemble | Passive | Weighted ensemble | SMOTE |
| OnlineAdaC2 [17] | Ensemble | Passive | Weighted ensemble | Cost weight |
| OnlineCSB2 [17] | Ensemble | Passive | Weighted ensemble | Cost weight |
| OnlineRUSBoost [17] | Ensemble | Passive | Weighted ensemble | Undersampling |
| OnlineSMOTEBoost [17] | Ensemble | Passive | Weighted ensemble | SMOTE |
| ARE$_{RE}$ [18] | Ensemble | Active | ADWIN | Cost weight |
| RB [19] | Multiple (4) | Active | ADWIN | SMOTE |
| OOB [20] | Ensemble | Left to pipelined algorithm | Left to pipelined algorithm | Oversampling + Cost weight |
| UOB [20] | Ensemble | Left to pipelined algorithm | Left to pipelined algorithm | Undersampling + Cost weight |
| WEOB1/WEOB2 [21] | Ensemble | Left to pipelined algorithm | Left to pipelined algorithm | Cost weight |
| **C-SMOTE [4]** | **Meta-strategy** | **Left to pipelined algorithm** | **Left to pipelined algorithm** | **SMOTE + ADWIN** |

wrapped up in Table 2, are able to learn from imbalanced data stream and dealing with concept drift changes that perfectly suit this task. They are commonly categorized into two major groups: passive versus active approaches, depending on whether an explicit drift detection mechanism is employed. Passive approaches train a model continuously without an explicit trigger reporting the drift, while active approaches determine whether a drift has occurred before taking any actions. Examples of passive approaches are RLSACP [13], ONN [14], ESOS-ELM [15], an ensemble of neural network [16], OnlineUnderOverBagging, OnlineSMOTEBagging, OnlineAdaC2, OnlineCSB2, OnlineRUSBoost and OnlineSMOTEBoost [17], while ARF$_{RE}$ [18], RebalanceStream [19] are considered active approaches. Then, there are other models like OOB and UOB [20], WEOB1 and WEOB2 [21] that, similarly to C-SMOTE, propose only an online rebalance strategy, leaving to the pipelined algorithm the concept drift management (approach type and the concept drift approach).

From Table 2, we can notice that the major part of the SML+ methods combine more learners together (ensemble strategy). Only RLSACP uses a single learner and RB uses four learners in parallel. About the approach to manage the concept drift occurrence, we can see that the most used one is to assign a weight to each learner of the ensemble and to discard the one having the lowest weight (weighted ensemble). In this way, the ensemble is composed only of the learners that perform well in the underlying concept. Only a couple of methods use forgetting factor strategies, giving, gradually, less importance to the past data and more importance to the new data in input. There are also two methods that adopt the ADWIN [3] strategy, while the last four leave this task to the pipelined algorithm. Instead, about the class imbalance management, there are different option used. Some algorithms use the cost weight strategy in which the minority class sample importance is increased in comparison to a sample from the majority class, others use SMOTE [5], while others combine together different strategies such as undersampling and oversampling or undersampling/oversampling and cost weight. The last important thing to notice is the difference with C-SMOTE. Being a meta-strategy to be pipelined with any other SML- technique, the approach type and the concept drift approach used by C-SMOTE are the ones used by the model to which is prepended by (SML-). Instead, as class imbalance approach, C-SMOTE uses the combination of ADWIN and SMOTE. In particular, in addition to the different rebalance strategy proposed, OOB, UOB, WEOB1 and WEOB2 differ from C-SMOTE in the classifier type. They all use an ensemble strategy, while C-SMOTE, being a meta-strategy, can be prepended to any type of classifier (single, multiple or ensemble).

## Experimental Settings

This section has five parts. The first four ones discuss i) the dataset, ii) the algorithms, iii) the metrics, and iv) the various experimental settings used to test C-SMOTE, while the last part introduces the hypotheses to test.

**Table 3.** Concept drift and imbalance level summary for several countries.

| Country | Concept Drift | % Increment (1) | % Decrement (0) |
|---|---|---|---|
| ITA | 8 | 51.08% | 48.92% |
| FRA | 12 | 46.97% | 53.03% |
| ESP | 10 | 44.47% | 55.53% |
| GBR | 9 | 47.85% | 52.15% |
| USA | 9 | 53.81% | 46.19% |
| BRA | 8 | 38.44% | 61.56% |
| CHN | 7 | 32.02% | 67.98% |
| IND | 5 | 45.31% | 54.69% |

**Table 4.** Worldwide summary of concept drift, increment and decrement.

| Measure | Min | Max | Avg | Median |
|---|---|---|---|---|
| Concept drift | 0 | 12 | 5.42 | 5 |
| % Increment (1) | 0.00% | 58.27% | 40.96% | 44.92% |
| % Decrement (0) | 41.73% | 100.00% | 59.04% | 55.08% |

## Dataset

To empirically evaluate the C-SMOTE meta-strategy in forecasting the COVID-19 pandemic trend, we used a COVID-19 dataset gathered monitoring the worldwide spreading of the pandemic [25]. The original dataset, updated with the number of new cases every day, had *59 attributes* (54 numerical and 5 nominal) indicating the country, the number of new and total cases and deaths both as absolute numbers and per million of people, the number of tests done, the number of patients in intensive care units and vaccinated people both as absolute numbers and per million of people and some population markers i.e. population number, population density, median age, hospital beds number. We replaced the original *date* attribute with the related *day, month, year, day of week, week of year and is-holiday* attributes. The last attribute states if that day is a holiday or not in that country. We also removed the *tests-units* attribute since it is only a unit of measurement. So, the final version of the dataset has *63 attributes* (54 numerical and 9 nominal). Moreover, we added a label that states if the number of new COVID-19 cases in a day are more or less than the ones that occurred in the previous day (*0* if they are less or equal, *1* if they are greater). For convenience, the *minority* class is always the class *1*, while the *majority* one is the class *0*. The overall imbalance ratio calculated at 05/02/2021 was *41.45%*. Table 3 shows, for the most common country, the number of concept drifts and the imbalance level, while Table 4 shows some statistics of all the countries.

## Algorithms

As SML- models, we tested the Adaptive Random Forest (ARF) [26], Naive Bayes (NB), Hoeffding Adaptive Tree (HAT) [27], K-Nearest Neighbor (KNN) and Temporally Augmented Classifier (SWT) [28] with ARF as base learner techniques. We pipelined these algorithms with the C-SMOTE meta-strategy and compared them against the stand-alone versions. Instead, as SML+ models, we tested the $\text{ARF}_{\text{RE}}$ [18], RB [19], OOB and UOB [20] techniques. Unfortunately, the implementations of the other algorithms cited in the Related Work section were unavailable or did not work.

## Metrics

We evaluated the predictive performances using the prequential evaluation approach [29]. Following He and Garcia [2], we used the most adopted metrics in the literature to address class imbalanced learning problem. They are the *Recall* (R), *F1-Measure* (F1), and *G-mean* (GM). In particular, we computed the first two metrics separately for the minority (R[1], F1[1]) and the majority (R[0], F1[0]) class, while the latter metric is across classes and measures the balance between the minority and majority classes performances [30]. We avoided using the *Accuracy* metric because it is not reliable in a scenario where the interest is to accurately predict the minority class occurrences. This metric would always score a high value due to the majority class samples high occurrences, deceiving the users about the goodness of the result achieved. In fact, *Accuracy* fails to reflect that all the minority class samples were misclassified. Last, we did not show the *Precision* metric results because they can be derived from the comparison of *Recall* and *F1-Measure* metrics results. We performed $10$ runs, so the results proposed are the average.

### Settings

All the experiments were made using the MOA framework[3] with default hyperparameters values for all the techniques involved. The only parameter that we set in C-SMOTE was $minSizeMinority$, the minimum number of minority class samples stored into the window to allow the rebalancing procedure. We used C-SMOTE with $minSizeMinority = 10$ as it was the top performer among (10,100,500,1000) in our hyper-parameter analysis. In particular, RB used four SWT models, while OOB and UOB, as a pipelined algorithm, used a Hoeffding Tree (HT) [31].

All the tests were run in a machine with 2 virtual CPUs Intel Skylake P-8175 at 2.5 GHz and 8 GiB of RAM.

### Research Hypotheses

We formulate our hypotheses as follows:

- *Hp. 1*: We assume that the minority class results of binary classifiers (SML-) pipelined with C-SMOTE are statistically better than those without it.
- *Hp. 2*: We assume that the minority class results of at least one binary classifier (SML-) pipelined with C-SMOTE are statistically better than those achieved by the state of the art techniques (SML+).

## Results and Discussion

In the first part of this section, we discuss the results achieved by the comparison between the pipelines of C-SMOTE and some SML- algorithms and the SML- models alone, while in the second part we compare the C-SMOTE pipelines with some SML+ techniques.

### SML- Comparison

In this section, we discuss the comparison among the ARF, HAT, NV, KNN and SWT algorithms pipelined with and without the C-SMOTE meta-strategy both in term of performances comparison over time and statistical tests, checking the *Hp. 1* validity.

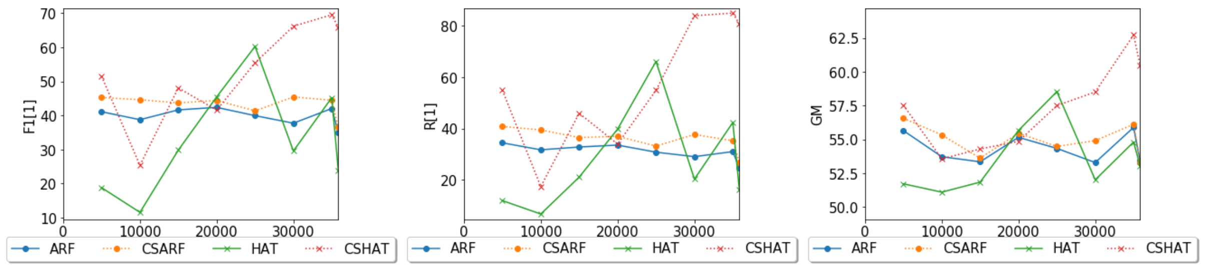
#### Performances Comparison

Fig. 3 shows some minority class performances comparisons over time among the ARF and HAT techniques pipelined with and without C-SMOTE. In all the minority class metrics, we can notice that both SML- models pipelined with C-SMOTE outperform their respective baselines, already verifying the *Hp. 1* hypothesis. We can also point out that there are multiple concepts drifts occurrences as noticed by the continuous performances ups and downs. However, to validate the performances of all the SML- models pipelined with and without C-SMOTE all over the metrics, we decided to perform a statistical test analysis.

#### Statistical Tests

To statistically prove that prepending C-SMOTE to one of the SML- methods presented in Section Algorithms improves the performances of each method, we used the one-tailed *T-Student*

[3] `https://moa.cms.waikato.ac.nz/`



**(a)** F1 minority class results. **(b)** Recall minority class results. **(c)** G-mean results.

**Figure 3.** Comparison between ARF and HAT pipelined with and without C-SMOTE.

| | F1[1] | | | | | F1[0] | | | | | R [1] | | | | | R [0] | | | | | GM | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STREAM | ARF | HAT | KNN | NV | SWT | ARF | HAT | KNN | NV | SWT | ARF | HAT | KNN | NV | SWT | ARF | HAT | KNN | NV | SWT | ARF | HAT | KNN | NV | SWT |
| Covid-19 | 🟩 | 🟩 | 🟥 | 🟩 | 🟩 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 | 🟩 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |

🟩 $Ha_1$ 🟥 $Ha_2$ 🟩 $H_0$

**Figure 4.** One-tail T-Student test comparing C-SMOTE pipelines with baselines.

test with the significance level $\alpha = 0.05$. We checked the *p-value*, in order to determine if there were significant differences between the results obtained with and without C-SMOTE. The null hypothesis $H_0$ is that, for each dataset and algorithm, the means of R[0], F1[0], R[1], F1[1] and GM are equal to the means of C-SMOTE R[0], C-SMOTE F1[0], C-SMOTE R[1], C-SMOTE F1[1] and C-SMOTE GM. We define two alternative hypotheses: $Ha_1$ is that the C-SMOTE means are greater than baseline ones, while $Ha_2$ is that the baseline means are greater than the C-SMOTE ones. The $H_0$ hypothesis is rejected in favor of the $Ha_1$ one if $\frac{\textit{p-value}}{2} < \alpha$ and the *t-statistic* $< 0$, while $H_0$ is rejected in favor of $Ha_2$ if $\frac{\textit{p-value}}{2} < \alpha$ and *t-statistic* $> 0$. Otherwise, both $Ha_1$ and $Ha_2$ are rejected in favor of $H_0$.

Fig. 4 shows the T-Students test results. The columns show the five metrics used and for each of them, there are five more sub-columns, indicating the comparison between the base version and the C-SMOTE version of that algorithm. The single-cell shows which hypothesis is accepted. Green cells tell that we rejected the $H_0$ hypothesis in favour of the $Ha_1$ one, red ones tell that we rejected $H_0$ in favour of $Ha_2$, while light green ones tell that both $Ha_1$ and $Ha_2$ are rejected in favour of $H_0$.

Definitively, we can say that *Hp. 1* is almost always verified. The use of the C-SMOTE meta-strategy improves both R[1] and GM results, meaning that the R[1] gain is bigger than the R[0] loss. Looking at the F1[1] results, in more than half of the cases, the C-SMOTE F[1] results are better than the base algorithms ones. This means that the P[1] increased too or that the R[1] gain is bigger than the P[1] loss.

## SML+ Comparison

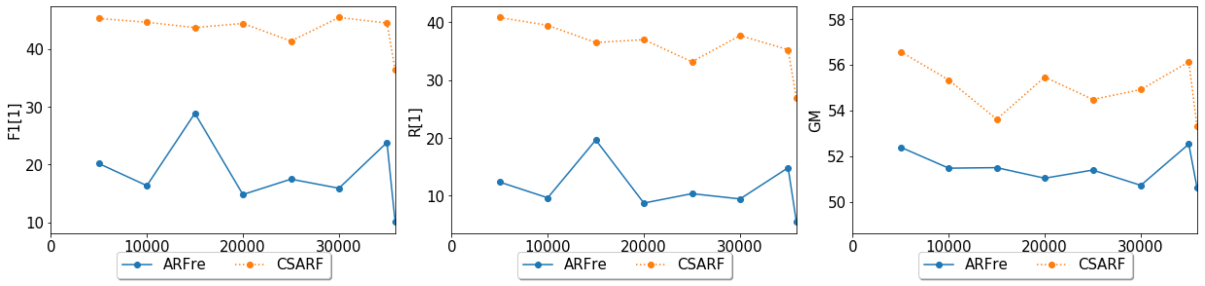
In this section, we discuss the comparison among the $\text{ARF}_{\text{RE}}$, RB, OOB and UOB SML+ algorithms and the ARF, HAT, NV, KNN and SWT algorithms pipelined with the C-SMOTE meta-strategy (from now on called C-SMOTE*) both in term of performances comparison over time and statistical tests, checking the *Hp. 2* validity.

### Performances Comparison

Fig. 5, Fig. 6, and Fig. 7 show some minority class performances comparisons over time among, respectively, the $\text{ARF}_{\text{RE}}$ and ARF pipelined with C-SMOTE techniques, the OOB, UOB and HAT pipelined with C-SMOTE techniques, and the RB and SWT pipelined with C-SMOTE techniques. The reason why we decided to show the results in this way is the similarity among the algorithms. $ARF_{RE}$ is a new ARF version properly introduced to deal with class imbalance; OOB and UOB used the HT, the first version of HAT, as their baseline; and RB used four SWT models. Both in Fig. 5 and Fig. 6, in all the minority class metrics, we can notice that the SML- models pipelined with C-SMOTE outperform, respectively, the $ARF_{RE}$ and the OOB and UOB techniques. Only the RB technique is better than the SWT pipelined with C-SMOTE one (Fig. 7). Also, in this case, to validate the performances of all the SML- models pipelined with C-SMOTE w.r.t. the SML+ techniques all over the metrics, we decided to perform a statistical test analysis.

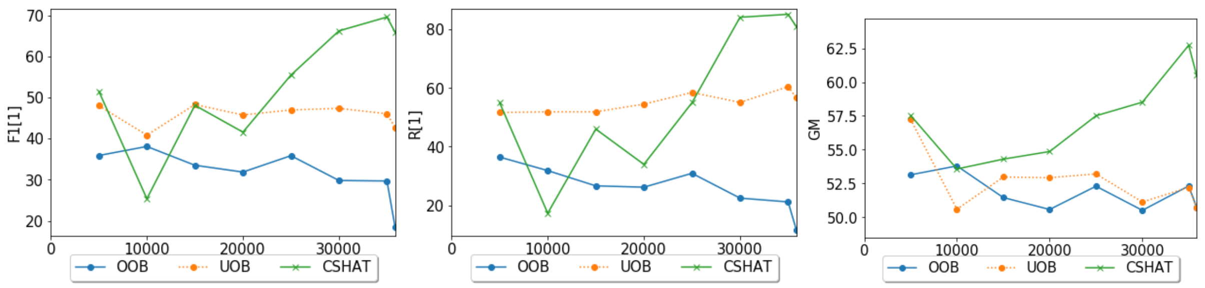
## Statistical Tests

To statistically compare the performances of the SML+ algorithms with the performances achieved by C-SMOTE*, we used the same T-Student tests with the same hypotheses used before. As Fig. 8 shows, comparing C-SMOTE* with RB algorithm, HAT pipelined with C-SMOTE is the only algorithm to outperform RB in R[1], F1[1] and GM. We can notice a similar behaviour comparing C-SMOTE* with UOB algorithm, with the considerable difference that, here, C-SMOTE* always improves the GM results. Finally, comparing C-SMOTE* with $ARF_{RE}$ and OOB algorithms, in general, the C-SMOTE* results are better. So, we can conclude that there is at least one algorithm pipelined with C-SMOTE that outperforms all the SML+ models, so *Hp. 2* is validated.



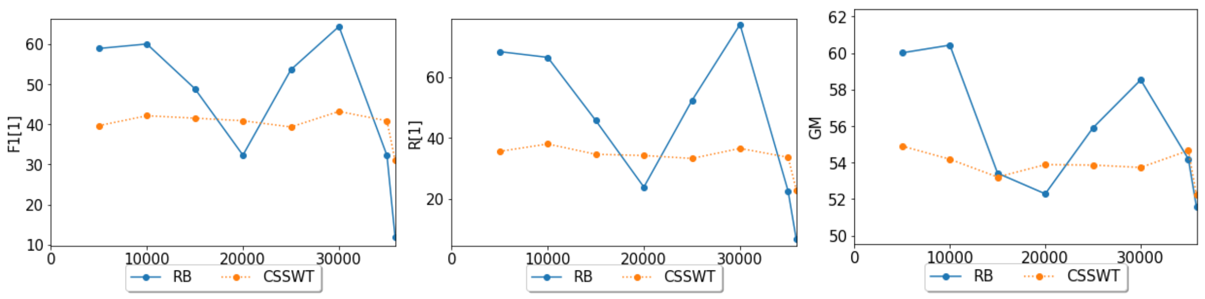
**(a)** F1 minority class results. **(b)** Recall minority class results. **(c)** G-mean results.

**Figure 5.** Comparison between $ARF_{RE}$ and ARF pipelined with C-SMOTE (CSARF).



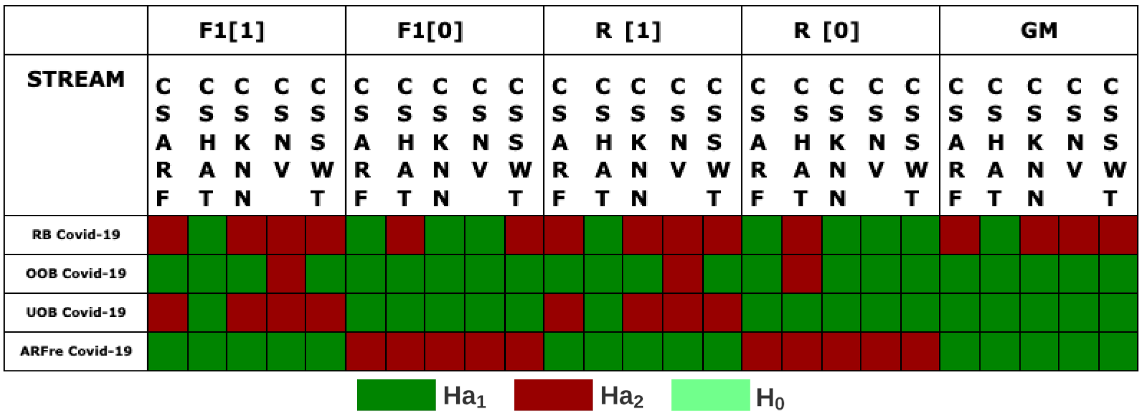
**(a)** F1 minority class results. **(b)** Recall minority class results. **(c)** G-mean results.

**Figure 6.** Comparison among OOB, UOB and HAT pipelined with C-SMOTE (CSHAT).



**(a)** F1 minority class results. **(b)** Recall minority class results. **(c)** G-mean results.

**Figure 7.** Comparison between RB and SWT pipelined with C-SMOTE (CSSWT).



| | F1[1] | | | | | F1[0] | | | | | R [1] | | | | | R [0] | | | | | GM | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STREAM | CSARF | CSHAT | CSKNN | CSNV | CSSWT | CSARF | CSHAT | CSKNN | CSNV | CSSWT | CSARF | CSHAT | CSKNN | CSNV | CSSWT | CSARF | CSHAT | CSKNN | CSNV | CSSWT | CSARF | CSHAT | CSKNN | CSNV | CSSWT |
| RB Covid-19 | 🟥 | 🟩 | 🟥 | 🟥 | 🟥 | 🟩 | 🟥 | 🟩 | 🟩 | 🟥 | 🟥 | 🟩 | 🟥 | 🟥 | 🟥 | 🟩 | 🟥 | 🟩 | 🟩 | 🟩 | 🟥 | 🟩 | 🟥 | 🟥 | 🟥 |
| OOB Covid-19 | 🟩 | 🟩 | 🟩 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟩 | 🟩 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| UOB Covid-19 | 🟥 | 🟩 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟩 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| ARFre Covid-19 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |

🟩 $Ha_1$ 🟥 $Ha_2$ 🟩 $H_0$

**Figure 8.** One-tail T-Student test comparing C-SMOTE* with RB, OOB, UOB and $ARF_{RE}$ models.

## Conclusions

In this work, we presented the application of the meta-strategy called C-SMOTE, based on the popular SMOTE technique that allows balancing an evolving data stream one sample at a time and it can be used as a data filter with all the streaming techniques, to forecasting the COVID-19 pandemic trend. Concerning SMOTE, C-SMOTE does not need a static batch during the pre-processing phase, but it saves every time the new sample in a window and, accordingly to ADWIN, it uses the minority class samples contained in it to introduce new synthetic samples.

We tested the meta-strategy pipelined with both SML- and SML+ models on a COVID-19 dataset, to demonstrate that C-SMOTE can be useful also in this emergency. Accurately forecasting if the number of infections will increase or decrease has a high impact on the economy and society overall.

The results summarized in Table 5 demonstrate that the C-SMOTE pipelines minority class results are, in most cases, better than both the ones of the SML- models alone (Q1) and the SML+ algorithms (Q2) i.e., C-SMOTE can magnify signs of decrement during the spreading periods and signs of increment in the contraction periods. We also proved that, in general, the recall of the minority class gain is bigger than the recall of the majority class loss i.e., the improvement in the ability to correctly forecast decrements (increments) when the infection is spreading (contracting) is larger than the error introduced. In particular, we can notice that the KNN, ARF, and SWT models, in this case study, are the best SML- all over the metrics to be pipelined with C-SMOTE. Thus, in light of the results achieved, we can affirm that in real-world scenarios presenting multiple concepts drifts occurrences and class imbalance, like the COVID-19 case study, C-SMOTE can enhance the performances of some SML- algorithms better than other SML+ models to the point that they can make an important statistical impact.

For future works, our principal goal is to perform memory- and time-consuming analysis. We also want to improve even more the C-SMOTE performance, reducing as much as possible the trade-off between improving the minority class performances and decreasing the majority class ones. The solution can be using rebalance techniques that consider the overlapping between classes (i.e. Borderline-SMOTE, ADASYN, DBSMOTE). Other improvements can be to adapt C-SMOTE to multiclass and regression tasks. In the long term, the aim is to investigate other meta-strategies based on different rebalance techniques and to compare them with C-SMOTE.

**Table 5.** Times that C-SMOTE prepended to the SML- models outperformed both the SML- models alone and the SML+ models tested. **Bold** means that it performed better in more than half of the total occurrences (5). The *Rank* column is the average of each model's rankings.

| **SML-** | **F1[1]** | **F1[0]** | **R[1]** | **R[0]** | **GM** | **Rank** |
|---|---|---|---|---|---|---|
| ARF | **3 (2)** | **3 (3)** | **3 (2)** | **3 (2)** | **4 (2)** | 2.2 |
| HAT | **5 (1)** | 2 (5) | **5 (1)** | 1 (5) | **5 (1)** | 2.6 |
| KNN | 2 (4) | **4 (1)** | **3 (2)** | **4 (1)** | **4 (2)** | **2** |
| NV | 2 (4) | **4 (1)** | 2 (5) | **3 (2)** | **4 (2)** | 2.8 |
| SWT | **3 (2)** | **3 (3)** | **3 (2)** | **3 (2)** | **4 (2)** | 2.2 |

## References

[1] A. Tsymbal, "The problem of concept drift: Definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[2] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[3] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *SDM*, SIAM, 2007, pp. 443–448.

[4] A. Bernardo, H. M. Gomes, J. Montiel, B. Pfahringer, A. Bifet, and E. Della Valle, "C-smote: Continuous synthetic minority oversampling for evolving data streams," in *BigData*, In press, IEEE, 2020.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[6] A. Fernández, S. García, F. Herrera, and N. V. Chawla, "SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary," *J. Artif. Intell. Res.*, vol. 61, pp. 863–905, 2018.

[7] H. Han, W. Wang, and B. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *ICIC (1)*, ser. LNCS, vol. 3644, Springer, 2005, pp. 878–887.

[8] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: adaptive synthetic sampling approach for imbalanced learning," in *IJCNN*, IEEE, 2008, pp. 1322–1328.

[9] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "DBSMOTE: density-based synthetic minority over-sampling technique," *Appl. Intell.*, vol. 36, no. 3, pp. 664–684, 2012.

[10] L. Abdi and S. Hashemi, "To combat multi-class imbalanced problems by means of over-sampling and boosting techniques," *Soft Comput.*, vol. 19, no. 12, pp. 3369–3385, 2015.

[11] C. Bellinger, S. Sharma, N. Japkowicz, and O. R. Zaïane, "Framework for extreme imbalance classification: SWIM - sampling with the majority class," *Knowl. Inf. Syst.*, vol. 62, no. 3, pp. 841–866, 2020.

[12] G. Douzas and F. Bação, "Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE," *Inf. Sci.*, vol. 501, pp. 118–135, 2019.

[13] A. Ghazikhani, R. Monsefi, and H. S. Yazdi, “Recursive least square perceptron model for non-stationary and imbalanced data stream classification,” *Evol. Syst.*, vol. 4, no. 2, pp. 119–131, 2013.

[14] ——, “Online neural network model for non-stationary and imbalanced data stream classification,” *Int. J. Machine Learning & Cybernetics*, vol. 5, no. 1, pp. 51–62, 2014.

[15] B. Mirza, Z. Lin, and N. Liu, “Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift,” *Neurocomputing*, vol. 149, pp. 316–329, 2015.

[16] A. Ghazikhani, R. Monsefi, and H. S. Yazdi, “Ensemble of online neural networks for non-stationary and imbalanced data streams,” *Neurocomputing*, vol. 122, pp. 535–544, 2013.

[17] B. Wang and J. Pineau, “Online bagging and boosting for imbalanced data streams,” *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3353–3366, 2016.

[18] L. E. B. Ferreira, H. M. Gomes, A. Bifet, and L. S. Oliveira, “Adaptive random forests with resampling for imbalanced data streams,” in *IJCNN*, IEEE, 2019, pp. 1–6.

[19] A. Bernardo, E. Della Valle, and A. Bifet, “Incremental rebalancing learning on evolving data streams,” in *ICDM (Workshops)*, IEEE, 2020, pp. 844–850.

[20] S. Wang, L. L. Minku, and X. Yao, “A learning framework for online class imbalance learning,” in *CIEL*, IEEE, 2013, pp. 36–45.

[21] ——, “Resampling-based ensemble methods for online class imbalance learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1356–1368, 2015.

[22] I. E. Agbehadji, B. O. Awuzie, A. B. Ngowi, and R. C. Millham, “Review of big data analytics, artificial intelligence and nature-inspired computing models towards accurate detection of covid-19 pandemic cases and contact tracing,” *International journal of environmental research and public health*, vol. 17, no. 15, p. 5330, 2020.

[23] I. Arpaci, S. Alshehabi, M. Al-Emran, M. Khasawneh, I. Mahariq, T. Abdeljawad, and A. E. Hassanien, “Analysis of twitter data using evolutionary clustering during the covid-19 pandemic,” *Computers, Materials & Continua*, vol. 65, no. 1, pp. 193–204, 2020.

[24] J. Farooq and M. A. Bazaz, “A novel adaptive deep learning model of covid-19 with focus on mortality reduction strategies,” *Chaos, Solitons & Fractals*, vol. 138, p. 110 148, 2020.

[25] J. Hasell, E. Mathieu, D. Beltekian, B. Macdonald, C. Giattino, E. Ortiz-Ospina, M. Roser, and H. Ritchie, “A cross-country database of covid-19 testing,” *Scientific data*, vol. 7, no. 1, pp. 1–7, 2020.

[26] 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,” *Mach. Learn.*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[27] A. Bifet and R. Gavaldà, “Adaptive learning from evolving data streams,” in *IDA*, ser. Lecture Notes in Computer Science, vol. 5772, Springer, 2009, pp. 249–260.

[28] A. Bifet, J. Read, I. Zliobaite, B. Pfahringer, and G. Holmes, “Pitfalls in benchmarking data stream classification and how to avoid them,” in *ECML/PKDD (1)*, ser. Lecture Notes in Computer Science, vol. 8188, Springer, 2013, pp. 465–479.

[29] J. Gama, R. Sebastião, and P. P. Rodrigues, “Issues in evaluation of stream learning algorithms,” in *KDD*, ACM, 2009, pp. 329–338.

[30] J. Akosa, “Predictive accuracy: A misleading performance measure for highly imbalanced data,” in *Proceedings of the SAS Global Forum*, vol. 12, 2017.

[31] P. M. Domingos and G. Hulten, “Mining high-speed data streams,” in *KDD*, ACM, 2000, pp. 71–80.