Journal of Computational Science 61 (2022) 101653

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

A new adaptive sampling algorithm for big data classification

Kheyreddine Djouzi *, Kadda Beghdad-Bey, Abdenour Amamra

École Militaire Polytechnique, BP 17, Bordj El-Bahri 16111, Algiers, Algeria

ARTICLE INFO

Keywords:
Big data
Data classification
Sampling methods
Subsampled Double Bootstrap
Naïve Bayes classifier

ABSTRACT

The exponential growth of the quantity of data that circulates on the web led to the emergence of the big data phenomenon. This fact is a natural consequence of the proliferation of social media, mobile devices, the abundance of free online storage, and new technologies like the internet of things. Subsequently, big data has created several challenges to the computer science community, among which the large size of data is the most challenging. Traditional machine learning algorithms used mostly for insight extraction find themselves inadequate, even on high-performance computer architectures. For instance, big data analytics algorithms can overcome the size issue by either: (1) adapting the existing machine learning techniques to the scale of the big data; or, (2) by sampling big datasets, choosing randomly much smaller subsets of the data population, to meet what current algorithms can handle. In the present work, we aim to proceed through the second alternative to address the size challenge in the big data context. We propose intelligent sampling techniques based on Scalable Simple Random Sampling (ScaSRS) and Subsampled Double Bootstrap (SDB). Test results carried out on public generic datasets show that our proposal is able to address the size dimension efficiently. The proposed algorithms were evaluative on a classification task where the obtained results provided significant improvement compared to the state-of-the-art.

1. Introduction

The world is witnessing an unprecedented data outburst due to the amount of digital content being generated at a high speed by various online platforms such as social media, smartphones, e-commerce, etc. The emergence of new technologies and paradigms, such as the Internet of Things (IoT), Cloud Computing, Interactive Web Applications, and Surveillance Systems have multiplied the quantity of data output to be stored and processed in order to predict upcoming events, optimize costs and profits, among other applications. These dynamics around information technologies have given rise to new concepts known as Big Data. As with most new concepts, we have to ensure sustainability over time, since some concepts do not succeed due to the lack of maturity, or supporting technology or market interest. Thus, Big Data presents many challenges as a field of research that mobilizes the data science community to address its problems and provide solutions in response to the large data that is being produced [1]. Data analysis is an obvious bottleneck in many applications, due to the lack of scalability of the underlying algorithms and the value of the data itself. Overcoming when comes to being linked to other data, thus data integration is a producer of major value [2].

Nowadays, most data is generated directly in digital format. Hence, we are faced with the opportunity and the challenge of adapting the creation of the raw data to facilitate subsequent processing, analysis, management, retrieval, and insight extraction [3]. Big in big data refers to the sheer size of the handled data, aside from being complex, and constantly changing in time [4]. Mostly, the rapid development of the internet has resulted in more freely available storage. More social media services and big data have become a phenomenon not only for computer science but also in most scientific and technical fields such as physics, biology, medicine, and engineering. When we properly handle this immense volume of correlated and heterogeneous data we could potentially maximize the knowledge and understanding of the underlying structure. However, this opportunity comes at a cost and several challenges. Initially, the current big data challenges emerged from data mining, which had been later translated through a range of algorithms and techniques to big data problems [5].

Among all challenges, the reduction of the size and parallel processing scheduling stand as the most daunting components for all big data systems [6]. Such a problem can be either addressed through revisiting the classical sampling and scheduling algorithms and adapting them to big data architectures or by wholly carrying out new research for customized solutions independently of what has been done for small data systems.

One feasible approach to optimizing the existing techniques and algorithms is through the exploitation of parallel computer architectures.

* Corresponding author.
E-mail address: djouzi.kheyreddine@gmail.com (K. Djouzi).

https://doi.org/10.1016/j.jocs.2022.101653
Received 16 February 2021; Received in revised form 22 February 2022; Accepted 20 March 2022
Available online 1 April 2022

To this end, we need to design and implement tools for large-scale data processing and management. This has accompanied the emergence of massive data processing platforms like the Apache Mahout, which were implemented with the map-reduce paradigm on top of Hadoop and Apache Spark. An alternative approach consists of carrying out the preprocessing of large volumes of data in order to reduce them so that they can be easily processed with analysis tools [7]. Here, data storage and reduction can be carried out in two different ways. If the reduction precedes storage, we are in the case of algorithms that apply a well-defined logic to select adequate data candidates for processing. Alternatively, when the storage precedes reduction, sampling methods are used to extract a sample that summarizes a larger population. This latter methodology was adapted in the present work.

The main objective of the present work is to improve the storage and computation performance of big data systems. Among all possible applications, we are particularly interested in big data classification tasks where we try to obtain optimal classification rates on samples instead of the whole data.

The remainder of this paper is organized as follows. In Section 2, we discuss previous work on sampling methods for big data. We present our algorithm in full detail in Section 3. The experimental results for the validation of the scalability and efficiency of SDBGDAS are given in Section 4. In Section 5, we conclude the paper and discuss future directions.

2. Related works

Sampling is the process in which a fraction of the population is selected to represent the characteristics of a larger group. In most cases, it is impossible to collect or analyze all the data at our disposal due to restrictions of time, money, and sometimes even access, this is what makes us resort to the sampling technique. Based on the existing sampling techniques in the literature, we found that they mainly belong to two main classes which are general-purpose sampling methods and adaptive sampling methods.

The first sampling methods that emerged are based on probabilistic techniques. A probability sampling method is a method of selecting a sample based on chance and a random draw. The method is said to be probabilistic because the representativeness of the random sample obtained is a priori ensured by the statistical laws of probability. The most common probability sampling methods in literature are: Simple Random Sampling (SRS) [8], Stratified Random Sampling (StRS) [9], Systematic Sampling (SyS) [10] and Scalable Random Sampling (ScaSRS) [11]. The most intuitive and common method of sample selection is Simple Random Sampling (or SRS) in which the sample is selected unit by unit and with an equal probability of selection for each unit and at each draw. Despite its simplicity, this method requires random access to data in addition to the number of sample elements.

Another general-purpose known sampling method is called Stratified random sampling where, the set of elements S is heterogeneous, which is common for large-scale datasets. It may be possible to partition it into multiple homogeneous subsets called strata and noted $s_1, \ldots, s_n$. By "homogeneous", we mean the elements of a stratum are similar to each other. For example, a learning set can be divided into positive and negative activities, or web users' activities can be partitioned according to the days of the week. The first step is to apply a sampling plan (eg. SRS) to obtain a sample of strata, then in each stratum, we apply sampling to have a representative subset of this stratum. Although this method provides better coverage, the stratification process requires complicated operations. Moreover, this method cannot be applied when the population is not divisible into adjustable subgroups; therefore, it is not useful.

Systematic sampling, on the other hand, is a type of probability sampling where each element of the population has a known and equal probability of being selected. Hence, the sample can be spread more evenly over the population. The probabilistic framework is maintained by selecting one or more random starting points [10]. Often, a systematic sample is obtained by randomly selecting 1 unit from the first k units of the population and each kth element thereafter. This approach is called a systematic sample 1 in k with a random start. Unfortunately, this method requires both the total and sample size which makes it less efficient. Besides, the sample might not be representative in the case of periodic datasets.

Xiangrui Meng proposed in [11] a Scalable SRS algorithm that has the same principle as the random sorting algorithm mentioned above (with the same X_i sequence). In addition, it uses probabilistic thresholds to accept, reject or list a pending element in order to reduce the number of incoming elements in the sorting step. Furthermore, it can be used in a parallel environment, allowing a very high performance to be achieved. Nevertheless, this method is complicated to implement.

On the other hand, adaptive sampling methods are more suited for big data. In literature, progressive methods, also known as static adaptive methods, were the first adaptive methods that have emerged. John and Langley [12] introduced the arithmetic sampling method, which is based on the following schedule $S_a = \langle n_0, n_0 + n_\delta, n_0 + 2n_\delta, \ldots N\rangle$, where n_0 is the starting sample size, and n_δ represents the fixed difference between successive terms. Arithmetic sampling has the main inconvenience which is if n_{min} (the size of the smallest sufficient training set at which convergence occurs) is a large multiple of n_0, then in order to reach convergence, many runs of the underlying algorithm are required.

To overcome the shortcomings of arithmetic sampling, progressive sampling with a geometric schedule is used. The schedule is used as follows, $S_g = \langle n_0, an_0, a^2n_0, a^3n_0, \ldots, N\rangle$, where a is the common ratio. However, one of the constraints of this method is overshooting. In large datasets, the instances number to be added in each iteration increases exponentially, resulting in the sample size greatly exceeding the optimal number n_{min}. The work presented in [13] proposes a dynamic adaptive sampling algorithm known as DASA, based on the definition of the learning curve phenomenon [14] that uses Chernoff inequality [15] to find how many instances are needed to achieve convergence. Although, this approach did not take into consideration sample bias or variance of the sample as the data characteristics, and it was not useful for incremental learners.

In order to overcome the limitations of DASA, authors of [16] proposed a novel algorithm dubbed Generalized Dynamic Adaptive Sampling (GDAS). GDAS uses the Chebyshev inequality [17] to calculate the size of the sample and take advantage of the information on sample variance to obtain a more accurate sample by using the Bootstrap method [18]. However, the Bootstrap technique to estimate variance is very expensive in terms of execution time, extracting several samples and calculating the accuracy for each can make the algorithm cumbersome. One should notice that both algorithms do not specify how the samples are selected. The selection algorithm is unknown, which is a real problem because this step can radically affect the execution time. However, these two limitations make the GDAS method low in terms of scalability, which is an essential feature for algorithms processing data at the Big Data scale. In Table 1 we summarize these techniques by excluding non-probability methods and specifying their strengths and weaknesses.

3. Proposed approach

The representation of a sample is the idea, so it must produce results that will be extrapolated to the population studied. To formalize this idea, it is necessary to define what is meant by a representative sample. However, a representative sample is not an end in itself since we want the statistically obtained results (classification accuracy) to be able to remain generalizable over the whole population. Therefore, what interests us more is the representativeness of these results. The latter must be derived from unbiased estimates, which not all sampling strategies do. Concerning the adaptive methods, we estimate a quantity

Table 1
Comparative study of the different criteria, advantages and disadvantages of each method.

Method	Description	Replacement	Adaptability	Multisteps	Pros	Cons
SRS	Chooses a random subset in a larger dataset.	Possible	No	No	+ Simple to implement.	- Needs of sample elements number - Needs a random access to data.
ScaSRS	An implementation of the SRS.	No	No	No	+ Gives better performance compared to the simple random sorting + Parallelizable.	- The total number of items and the number of items in the sample are always required - A more complicated implementation.
StRS	Involves dividing a population into strata	Possible	No	Possible	+ Get better coverage by exploiting additional information.	- Not useful when the population cannot be divided into disjoint subgroups - Stratification is an expensive operation - Needs whether the total size and the sample size.
Systematic	Selects a random element of departure then an element every k clusters.	Possible	No	Possible	+ Spreads the sample more evenly over the population.	- The sample may not be representative if the data set is periodic - Needs the total size and sample size.
Arithmetic	Selects the instances to be included in the sample in a progressive arithmetic manner.	No	Yes	No	+ Finds a sample size that maximize the interest variable.	- The running time depends on the number of elements to add to each step.
Geometric	Selects the instances to be included in the sample in a geometric progressive manner.	No	Yes	No	+ Finds a sample size that maximize the interest variable + A better running time	- May far exceed the size of convergence.
DASA	Selects the instances to be included in the sample based on the characteristics of the data in dynamic way.	No	Yes	No	+ Takes advantages of the data to obtain the sample size estimates	- The Chernoff bound is somewhat low and the estimated sample may be large.
GDAS	Selects the instances to be included in the sample based on the characteristics of the data in dynamic way.	Yes	Yes	No	+ Uses the sample information to estimate the sample size. + Uses variances measures for more accuracy.	- The running time is greater because of the variance estimation.

for which, whatever the sample is, we do not have unbiased estimators. The conclusion would be that there are no representative samples and our work would be useless.

We must therefore think in terms of utility. More formally, if we consider our entire population as a very large dataset n ($n \gg 1$) and $u(D)$ the utility function that corresponds to D (i.e. in our context, the utility function represents the classification accuracy). We then seek a subset $D' \subset D$ such that $\hat{u}(D') \approx u(D)$. In other words, it exists $\epsilon > 0$ very small, such as:

$$|\hat{u}(D') - u(D)| < \epsilon \tag{1}$$

with $\hat{u}(D')$ denoting the utility estimated by the sample D'.

The proposed sampling method for mass data classification takes the GDAS method as a backbone and employs other techniques for improving sample performance and quality. We will therefore use it as a foundation for a combined technique using the ScaSRS algorithm for the selection phase and a scalable Bootstrap technique (SDB) to efficiently estimate the variance. Also, we will use an active approach to select instances that can improve the accuracy of the classification. Roughly speaking, the process of our solution is summarized in Fig. 1.

3.1. Data selection using ScaSRS

ScaSRS is a simple scalable random selection algorithm designed by Xiangrui Meng of *LinkedIn*, which applied the method to a data set of 60 million to 1.5 billion units in 2013 [11]. This method uses probabilistic thresholds, by which the execution algorithm decides whether to accept or reject the population in the sample or whether to include it in the waiting list, thereby reducing the number of instances in the sorting phase. In the initial analysis, we assume that we know k (sample size) and n (population size) and we can simply infer the probability of selection as $p = k/n$. Probabilistic thresholds are then defined as follows:

- **Rejecting Items on the Fly:** if we reject items whose associated random keys are greater than:

$$q_1 = min(1, p + \gamma_1 + \sqrt{\gamma_1^2 + 2\gamma_1 p}) \tag{2}$$

where $\gamma_1 = -\frac{log\delta}{n}$ and n is the number of all elements of the population and the probability of sampling $p = \frac{k}{n}$ with k is the desired sample size, for some $\delta > 0$. The resulting algorithm is still correct with a probability of at least $1 - \delta$.

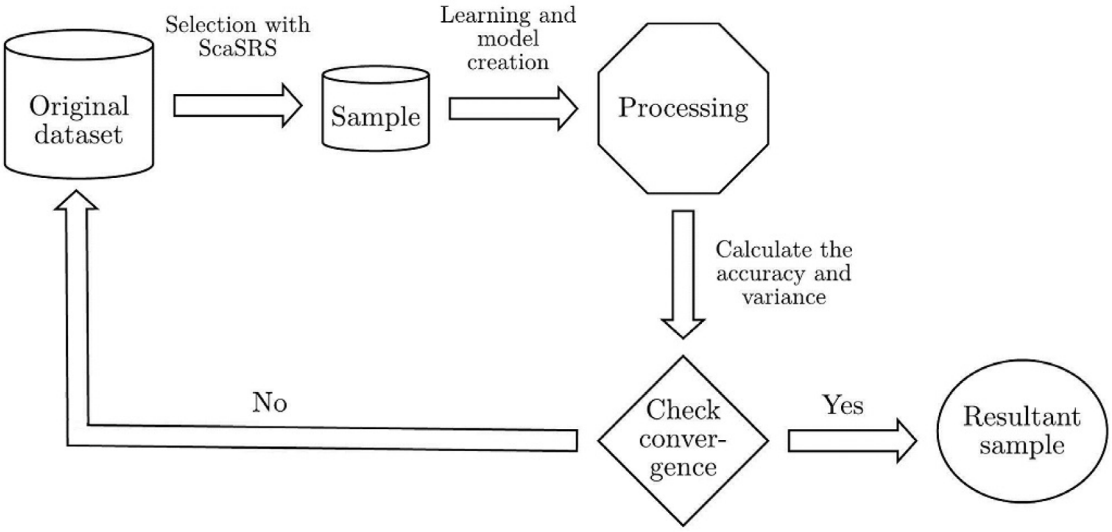

Fig. 1. Illustration of the proposed sampling approach.

- **Accepting Items on the Fly:** if we accept items whose associated random keys are less than:

$$q_2 = max(0, p + \gamma_2 - \sqrt{\gamma_2^2 + 3\gamma_2 p}) \quad (3)$$

where $\gamma_2 = -\frac{2log\delta}{3n}$, for some $\delta > 0$. The resulting algorithm is still correct with a probability of at least $1 - \delta$.

Algorithm 1: ScaSRS algorithm

Input: The confidence parameter δ, the size of sample k and the dataset S of size n
Result: The list of selected elements

```
Compute q1 and q2 based on Eqs. (2) and (3);
Let l = 0 and W = ∅ be the waiting list;
for each item s_j ∈ S do
    Draw a key X_j independently from U(0; 1);
    if X_j < q2 then
        Select s_j and let l = l + 1;
    else
        if X_j < q1 then
            Associate s_j with X_j and add it into W;
        end
    end
end
Sort W's items in the ascending order of the key;
Select the smallest k − l items from W.
```

ScaSRS (defined in Algorithm 1) is a randomized algorithm with a certain chance of failure. In practice, we set $\delta = 5 \times 10^{-5}$, hence the failure rate is controlled at 0.01%, which makes ScaSRS a reliable algorithm.

3.2. Learning and model creation

In this step, there is a multitude of available techniques in order to classify data and create learning model. Among which we mention: Naïve Bayes Classifier [19], Bayesian Networks [20], Decision Trees [21], Rule-Based Classification [22], Deep Convolutional Neural Networks [23], Logistic Regression [24], Support Vector Machines (SVM) [25].

3.3. Calculating the variance

We mentioned earlier that variance estimation using Bootstrap in GDAS is one of the main disadvantages, especially for relatively large sample sizes. The Bootstrap provides a simple yet powerful way to evaluate the quality of estimators. However, in an environment involving large amounts of data, the calculation of Bootstrap-based parameters induces a high computational load.

To cope with this weakness we thought of the use of a scalable and efficient technique designed for big data called Subsampled Double Bootstrap (SDB). We present this new technique which could help to overcome the aforementioned drawback.

The Subsampled Double Bootstrap method (SDB) is a combination of subsampling and technical acceleration of the Double Bootstrap [26] method proposed by Sengupta et al. [27] to construct a massive data sample in order to evaluate the accuracy of statistical inference or to apply DM algorithms available for simple data. It retains the automatic nature and computing power of Bag of Little Bootstraps (BLB) [28]. However, the authors argue that SDB outperforms BLB in some scenarios with a time-limited cost, especially when it is only possible to run small bootstraps. The SDB method pseudocode is shown as in Algorithm 2.

Algorithm 2: SDB Algorithm

Input: data $\mathcal{X}_n = \{X_1, ..., X_n\}$, θ : parameter interest, $\hat{\theta}_n$: estimator, $\xi(.)$: accuracy measure, b: subset size, S: number of subsets, $T_n(\hat{\theta}_n, \theta)$: root function
Result: $\xi(Q_n)$ estimates of ξ

```
for j ← 1 to S do
    Choose random subset X*j_{n,b} from X_n ;
    Compute θ̂(X*j_{n,b}) from X*j_{n,b} ;
    Generate resample (X*j_{n,b}, W*(j,1)_{n,b}) from X*j_{n,b} ;
    Compute resample estimate θ̂(X*j_{n,b}, W*(j,1)_{n,b}) from
      (X*j_{n,b}, W*(j,1)_{n,b}) ;
    Compute resample root: T*j_n = T_n((θ̂(X*j_{n,b}), W*(j,1)_{n,b}), (θ̂(X*j_{n,b})) ;
end
Compute empirical distribution of roots:
  Q̂_{n,b,S} = ecdf{T*1_n, ..., T*S_n} ;
Calculate the plug-in estimator ξ(Q̂_{n,b,S}).
```

3.4. Improved sample accuracy using active learning

In order to better select the instances that will be used in the classification, more information on the instances must be introduced for the choice decision. Active learning, therefore, uses this principle

in order to achieve greater precision by using a minimum of instances. The principle is to give an efficiency score to each instance and estimate whose instances will give the maximum knowledge. These instances will be part of the sample which has the effect of improving the sample quality. A probabilistic classifier is generally used to calculate efficiency scores.

Introducing a probabilistic classifier is not time-consuming for learning. We will use the active learning approach with the ScalableSRS algorithm (the pseudocode is given in Algorithm 3). We will proceed with the following steps:

1. First, we choose an initial sample D_i from D.
2. We will make learning to this sample using a probabilistic algorithm.
3. Subsequently, we proceed as in the ScaSRS algorithm using the same thresholds, except that the items that will be included in the waiting list will have priorities according to their classification (well or misclassified) and the probabilities of classification.

Algorithm 3: Active ScaSRS

Input: The confidence parameter δ, the size of sample k, the dataset S of size n, and a probabilistic classifier C generated a priori by an initial sample D_1

Result: The list of selected elements

Compute q_1 and q_2 based on Eqs. (2) and (3);
Let $l = 0$ and $W = \emptyset$ be the waiting list;
for *each item* $s_j \in S$ **do**
  Draw a key X_j independently from $U(0; 1)$;
  if $X_j < q_2$ **then**
    Select s_j and let $l = l + 1$;
  else
    if $X_j < q_1$ **then**
      if s_j *is misclassified by* C *with a probability* p **then**
        Associate s_j with $q1 + X_j \times p$ and add it into W;
      else
        Associate s_j with $q2 + X_j \times (1 - p)$ and add it into W;
      end
    end
  end
end
Sort W's items in the ascending order of the key;
Select the smallest $k - l$ items from W.

3.5. Utility confidence interval (convergence)

Definition 1. Let u be our utility function, $u(D)$ the real utility when using all data (population), and $\hat{u}(D_i)$ the estimate of its utility on the basis of a sample $D_i \in D$ of size m. Then θ is a utility confidence bound to u if and only if $\forall \delta \in]0, 1]$:

$$Pr[|u(D) - \hat{u}(D_i)| \geq \theta] \leq \delta \tag{4}$$

The Eq. (4) translates the fact that θ provides a bilateral confidence interval of $\hat{u}(D_i)$ with a confidence level δ. In other words, the probability of drawing a D_i sample, so that the difference between the actual and estimated utility of a disagreeable assumption of θ or more (in both directions) is below δ. If more than $\forall \delta \in]0, 1]$ and $\forall \epsilon > 0$, there is a number m such as $\theta < \epsilon$, the confidence interval can be reduced (at any confidence level δ) at arbitrarily low nonzero values using a sufficiently large sample.

In this context, the utility function we consider is the average over all instances, of an instance function $f(x_i)$, where $x_i \in D$. This function determines whether an instance is properly classified or not. The utility is then defined as: $u(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} f(x_i)$ the precision on the whole dataset and $\hat{u}(D_i) = \frac{1}{|D_i|} \sum_{i=1}^{|D_i|} f(x_i)$ the precision on the sample D_i.

3.6. Chebyshev inequality

In probability theory, Chebyshev inequality is a concentration inequality making it possible to show that a random variable will take with a high probability a value relatively close to its expectation. This result applies in very diverse cases, requiring the knowledge of few properties (only the expectation and the variance must be known), and makes it possible to demonstrate the weak law of large numbers.

Definition 2. For any probability distribution, the probability that the average estimate of p' is greater than ϵ near the true value of p of the average after m independent observations is increased by:

$$Pr[|p - p'| \geq \epsilon] \leq (\frac{\sigma^2}{p'^2 \epsilon^2}) \frac{1}{m} \tag{5}$$

We will therefore have:

$$\lim_{n \to \infty} Pr[|p - p'| \geq \epsilon] = 0 \tag{6}$$

because $\lim_{n \to \infty} \sigma^2 = 0$ according to the weak law of large numbers in combination with the properties of the learning curve.

Two reasons why Chebyshev inequality is used for our generalized sampling algorithm:

- Chebyshev gives a better limit than the Markov [29] and Chernoff inequalities because it uses more information on the distribution. In particular, the standard deviation of the random variable and therefore its variance;
- Chebyshev's limit applies to a class of random variables and gives an exponential drop in the probability.

When stopping this process is a sequential decision problem? Indeed, we do not know the actual advantage of p as this would require the use of all data, which goes against the purpose of sampling. To meet this challenge, sample pairs are considered and one step at a time is observed.

Let D_{opt} be the optimal sample size of n_{min} (the smallest sample size sufficient to reach the plateau region). Ideally, we would stop when $|u(D_{opt}) - u(D_i)| < \epsilon$ (for some D_i), that is when we are at a distance ϵ from the optimal utility measure $u(D_{opt})$ (in both directions). Since we do not know D_{opt} for a given population, we use a «myopic »strategy consisting of looking at one step at a time. To use this myopic strategy, we need to define p the population and p' the estimate as follows:

Definition 3. We consider our population consisting of all separate pairs of training set sizes (α, β) with $\alpha > \beta > 1$ from two (02) successive sampling steps. Reaching the plateau region [30] translates into convergence when $u(D_\alpha) \approx u(D_\beta)$.

Definition 4. Let $u(D_\alpha)$ and $u(D_\beta)$ be utility measures of respective sample sizes $|D_\alpha|$ and $|(D_\beta)|$. We define the true value p and its estimate p' as follows:

$$p = u(D_\alpha) - u(D_\beta), |D_\alpha| > |D_\beta| > n_{min} \tag{7}$$

$$p' = u(D_\alpha) - u(D_\beta), n_{min} > |D_\alpha| > |D_\beta| > 1 \tag{8}$$

In practice, it is assumed that once the plateau area is reached, the value of p is zero, which agrees with Definition 3 above. The definition of p' is similar to that of p except that the constraint $|D_\beta| > n_{min}$ may not be satisfied in this case (i.e. we may not have reached the plateau region). Afterward, we can use p and p' to determine the number of observations needed for each iteration of our algorithm using Chebyshev's inequality. An estimate of the total number of individuals m is provided to ensure that the difference will be at the ϵ distance from an optimal utility. For two successive steps and, by substituting p and

p' values starting from the above definitions while using the Bootstrap variance estimate (σ^2_{BOOT}), Eq. (5) is summarized in the following:

$$Pr[|0-(u(D_{i+1})-u(D_i))| \geq \epsilon] \leq (\frac{\sigma^2_{BOOT}}{(u(D_{i+1})-u(D_i))^2\epsilon^2})\frac{1}{m} \leq \delta \quad (9)$$

We then solve for m of Eq. (9) to get:

$$m \geq (\frac{\sigma^2_{BOOT}}{(u(D_{i+1})-u(D_i))^2\epsilon^2})\frac{1}{\delta} \quad (10)$$

3.7. Termination criterion

An important aspect of sampling algorithms, like any other algorithm, is to determine when to stop at a particular sample. With each iteration of our algorithm, convergence is verified by performing the following tasks:

- Compare $|u(D_{i+1})-u(D_i)|$ to ϵ to evaluate the difference between the utilities of two successive iterations;
- Calculate, with Chebyshev inequality, the probability of occurrence of an event and compare it to δ, the confidence parameter.

There are four possible cases:

- **Case 1:** $|u(D_{i+1})-u(D_i)| \geq \epsilon$ and $(\frac{\sigma^2_{BOOT}}{(u(D_{i+1})-u(D_i))^2\epsilon^2})\frac{1}{m} > \delta$, we calculate the number of instances m using the previously derived Eq. (10) and add this number of instances to form the next sample D_{i+2}.
- **Case 2:** $|u(D_{i+1})-u(D_i)| \geq \epsilon$ and $(\frac{\sigma^2_{BOOT}}{(u(D_{i+1})-u(D_i))^2\epsilon^2})\frac{1}{m} < \delta$, here we have reached convergence because the probability of exceeding ϵ is less than δ.
- **Case 3:** $|u(D_{i+1})-u(D_i)| < \epsilon$ and $(\frac{\sigma^2_{BOOT}}{(u(D_{i+1})-u(D_i))^2\epsilon^2})\frac{1}{m} > \delta$, in this case, it seems that convergence is achieved from $|u(D_{i+1})-u(D_i)| < \epsilon$, but it is not, as the other constraint shows. We then discard this sample and start with a new sample.
- **Case 4:** $|u(D_{i+1})-u(D_i)| < \epsilon$ and $(\frac{\sigma^2_{BOOT}}{(u(D_{i+1})-u(D_i))^2\epsilon^2})\frac{1}{m} < \delta$, convergence is achieved with a probability of success greater than or equal to 1-δ and a stabilization of the error is noted.

3.8. SDBGDAS algorithm

The proposed SDBGDAS sampling method (described in Algorithm 4) is an improvement of the GDAS method. It is based on the SDB method for variance estimation instead of standard Bootstrap. This is in order to take advantage of the advantages of SDB namely better scalability and temporal complexity. SDBGDAS also uses the Active Scalable SRS method to improve the quality of the selected instances that will be included in the learning which will improve the quality of the resulting classification model.

As the proposed SDBGDAS method is based on SDB which performs better than BLB and Bootstrap in terms of temporal complexity, parallelization, and scalability. Then SDBGDAS is more powerful compared to the previous methods GDAS and DASA.

The use of the SDB method instead of Bootstrap where BLB influences the behavior of the proposed algorithm by increasing the computation time and decreasing the number of instances selected for processing. Using the Scalable SRS method improves sample quality by prioritizing well-ranked instances, and further reducing the number of instances included in the learning.

Algorithm 4: SDBGDAS algorithm

Input: D, ϵ, δ,
Result: n_{tot}, $t_{average}$
$\hat{u}(D_0) = 0$;
Randomly select $D_1 = ScaSRS((1/10) * |D|, |D|, \delta' = 0.00005)$ instances;
Make the learning, generate a probabilistic classifier C and determine $\hat{u}(D_1)$;
$n_{tot} = 0$;
foreach $i \geq 1$ **do**
  $n_{tot} = n_{tot} + |D_i|$;
  Calculate the variance σ^2_{SDB} using the SDB method;
  Check the convergence using the criterion in Eq. (9) ;
  if $\left(\frac{\sigma^2_{SDB}}{\epsilon^2(u(D_i)-u(D_{i-1}))^2}\right)\frac{1}{m} \leq \delta$ **then**
    return n_{tot} et $t_{average}$
  else
    if $|u(D_i)-u(D_{i-1})| \leq \epsilon$ **then**
      Ignore the sample and repeat
    else
      Add instances according to chebyshev's inequality
    end
  end
  Draw randomly and actively m instances to form the new sample $Di+1$ using Eq. (10) and put $D_{i+1} = ActiveScaSRS(|D_i|+m, |D|, \delta' = 0.00005, C)$;
  Make the learning and determine $\hat{u}(D_{i+1})$;
end

4. Experimental results and discussion

4.1. Test datasets

In this subsection, we will present the datasets we used to evaluate our method. Indeed, we have performed a series of tests with both real and synthetic data.

4.1.1. Real datasets

In this part, we will present the various dataset extracted from the UCI repository[1] and their setting in ARFF (Attribute-Relation File Format) which is a format that can be used by Weka. The advantage of this format is that it offers the possibility of processing a large number of instances with reasonable processing capacities and computation time. The ultimate purpose is to see the behavior of the implemented algorithms, which will lead to choosing candidate instances for the training and inference. Therefore, the real issue is how to choose these instances in a way that minimizes resource and processing time and maximizes the classification accuracy. The choice of the weka library and its format is appropriate to simplify things in the same way that other libraries and formats can be used in other contexts.

The four (04) real datasets we will use concern the recognition of human activity from the data indicated by the accelerometers and gyroscopes of a SmartPhone and a SmartWatch. These data are the coordinates x, y, z at the moment of registration. The activities are stand, sit, walk, bike, stairs up, stairs down and the datasets are as follows:

- *SmartPhone Accelerometer:* contains all the data captured by a smartphone's accelerometer at any given moment, and the corresponding human activities. This dataset contains $11,762,265$ instances after removing the unclassified instances in the preprocessing phase, in order to improve learning.
- *SmartPhone Gyroscope:* is the x, y, z coordinates captured by the smartphone's gyroscope as well as human activities. It has $12,063,000$ elements after eliminating unclassified instances.
- *SmartWatch Accelerometer:* through a smartwatch, the data sent by the accelerometer at the time of activity is recorded, along with human activities. After the necessary preprocessing, this dataset contains $2,535,000$ instances.

[1] **https://archive.ics.uci.edu/**

- *SmartWatch Gyroscope:* these are the recorded data from the smartwatch gyroscope, as well as human activities. This one contains $2,735,000$ instances.

4.1.2. Synthetic dataset

For generating our synthetic/artificial dataset, we use *make_blobs* method from sklearn package [31]. This function generates isotropic gaussian blobs for clustering by drawing samples from a special Gaussian mixture model.

A general gaussian mixture model [32] with k clusters has a density of the form:

$$p(x) = \sum_{i=1}^{k} \pi_i \mathcal{N}(\mu_i, \Sigma_i) \tag{11}$$

where $\pi_i \geq 0$ are the weights of each cluster with $\sum_{i=1}^{k} \pi_i = 1$, μ_i are the cluster centers, and Σ_i are the cluster covariances. Here $\mathcal{N}(\mu_i, \Sigma_i)$ refers to the normal gaussian density with mean μ_i and covariance Σ_i.

In particular, for the *make_blobs*() function, each cluster or component has equal probability of being sampled $\pi_i = 1/k$. Isotropic refers to the fact that the covariance matrices will all be diagonal $\Sigma_i = \begin{bmatrix} \sigma_i^2 & 0 \\ 0 & \sigma_i^2 \end{bmatrix}$ with σ_i being the standard deviation that is passed in. By default, all clusters will have the same standard deviation. Its prototype is as follows:

sklearn.datasets.make_blobs(**n_samples**=100, **n_features**=2, **centers**=3, **cluster_std**=1.0, **center_box**=(−10.0, 10.0), **shuffle**=*True*, **random_state**=*None*)

where ***n_samples*** is total number of points equally divided among clusters; ***n_features*** defines the number of features for each sample; ***centers*** parameter takes the number of centers to generate, or the fixed center locations; ***cluster_std*** specifies the standard deviation of the clusters; ***center_box*** allows to delimit the bounding box for each cluster center when centers are generated at random; ***shuffle*** decide whether to perform a mixture of samples and ***random_state*** determines random number generation for dataset creation. In our case we produced a dataset that contains 10.000.000 samples grouped in 10 clusters with a standard deviation equals to 0.2, it composed of 08 features.

4.2. Experimentation and performance testing

We implemented the different sampling algorithms as well as the variance estimation methods namely Bootstrap, BLB, and SDB in python programming language. We will compare the different selection methods in terms of execution time and error rate, then we will test the performance of the methods that will be used for variance estimation. We then use all previous methods to evaluate the proposed SDBGDAS method. The tests and evaluations were carried out on a Windows 10 with an Intel *i*5 2.7 GHz processor and 6 GB RAM.

4.2.1. Evaluation of the ScaSRS algorithm

In this section, we will compare the SelectionRejet (SR), RandomSort(RS), and ScalableSRS selection algorithms mentioned earlier. We will therefore perform six (06) tests $T_1, T_2, \ldots, T_6$, with sample sizes k and total numbers of instances n different, as illustrated in Table 2. Each time, we select the samples that are smaller than the total number of instances n, and we will see the execution time that corresponds to each algorithm based on the sample size. To control the failure rate at 0.01%, we use $\Delta = 5 \times 10^{-5}$, obtained results are given in Table 3

The execution time of the Random Sort (RS) algorithm increases rapidly when the sample size k increases. For the ScalableSRS algorithm, it takes its strength from the fact that it is highly parallelizable, and despite being implemented on a four-core machine. It gives good results compared with the results given by Selection-Rejection (SR), and so if in a distributed context the results will be even better seen its great scalability.

Table 2
The test samples of the selection algorithms.

	T_1	T_2	T_3	T_4	T_5	T_6
n	10^6	2.5×10^6	5×10^6	7.5×10^6	10^7	10^7
$p = k/n$	0.1	0.1	0.1	0.1	0.01	0.1
k	10^5	2.5×10^5	5×10^5	7.5×10^5	10^5	10^6

Table 3
The computation time of each test for each algorithm (in seconds).

	T_1	T_2	T_3	T_4	T_5	T_6
RS	22.53	34.12	42.35	> 500	> 500	> 500
SR	3.5	4.2	6	7.4	13.7	24.1
ScalableSRS	3.8	4.7	6.2	7.3	13.5	27.5

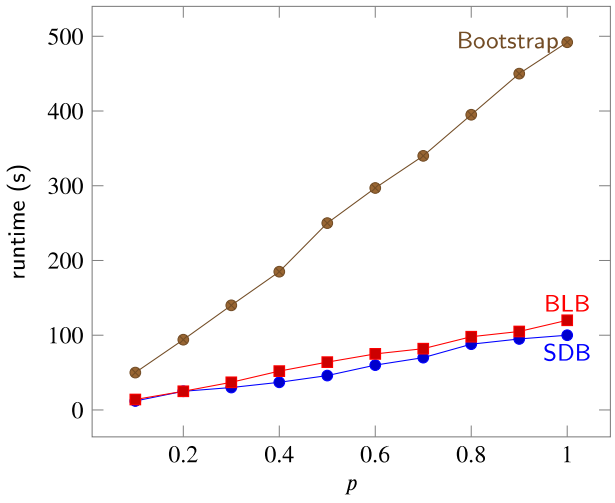

Fig. 2. The evolution of the execution time according to the sample size.

4.2.2. Evaluation of the SDB algorithm

For the classification, we will use Weka's *Naïve Bayes* algorithm, since it guarantees fast learning and is better suited to our situation where several tests have to be carried out. The dataset used in this part is Watch Accelerometer which contains several instances in the order of 10^6 instances.

For the BLB method, the values of the parameters recommended by [28] are: $b = n^{\gamma} = (10^6)^{0.6} \approx 44420$, $s = 20$ and $r = 100$. And for the SDB method we will use for the b and s parameters the same values that BLB namely $b = n^{\gamma} = (10^6)^{0.6} \approx 44420$ and $s = 20$. We will take for the method Bootstrap $r = 100$.

Given the k size of a sample and the total number of instances n, we will take several values of the parameter $p = \frac{k}{n}$ which takes its values between 0.1 and 1.0. It calculates the corresponding Bootstrap, BLB, and SDB variances, and observes the evolution of the execution times and the error rates obtained in each case. The graphs illustrated in Figs. 2 and 3 respectively give the variations in run times and the error rate about the sample sizes taken in each test.

We observe that there is a linear variation in the execution times of the different algorithms according to the sample sizes. The Bootstrap method increases faster than BLB and SDB, but what is interesting is that for BLB. The execution time increases as much faster than SDB and this is a result justified by the fact that SDB has a better temporal complexity than BLB and Bootstrap. BLB and SDB show almost similar variations in error rates depending on sample size. For better visibility of the results obtained, we draw the curves of the variations of the error rates according to the execution times, this is given by Fig. 4.

The curves obtained show the efficiency of the SDB method compared to Bootstrap and BLB, since it allows to reach low error rates more quickly, that is to say in shorter execution times. Knowing that

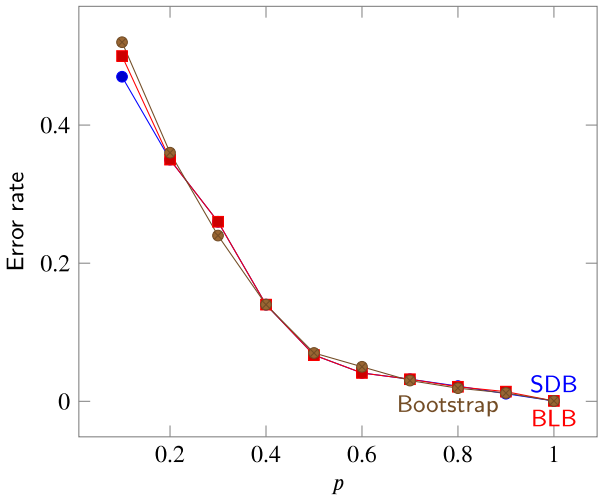

Fig. 3. The evolution of the error rate as a function of the sample size.

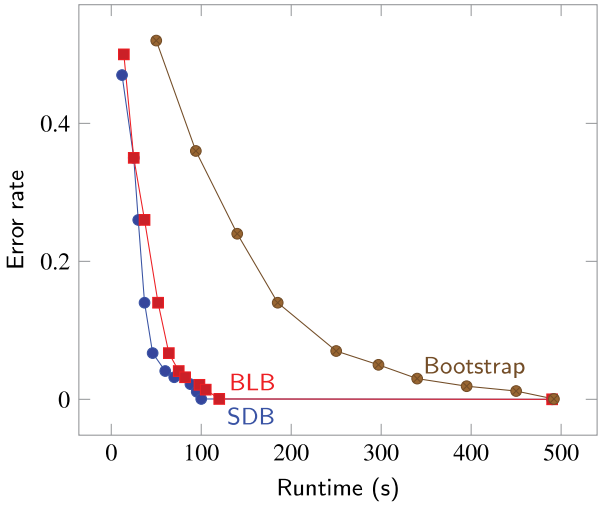

Fig. 4. The evolution of the error rate as a function of the execution time.

SDB takes its strength from the fact that it is highly parallelizable and is better suited with distributed environments. The results are expected to be even better if previous conditions are checked.

4.2.3. SDBGDAS with other methods comparison

We will make a comparative study of the different already existing methods compared to our proposed SDBGDAS method and we will give the results obtained in comparative tables accompanied by curves for visualization and to facilitate interpretation. For each experiment, the execution is repeated 10 times using 70% of the dataset for learning and 30% for testing.

We test our method and compare it with the following methods:

- **Full:** it is the most trivial method! That is to say we use all the instances to make the learning: $S_{full} = \langle N \rangle$;
- **Arith:** it is the arithmetic method [12] that generates samples as follows: $S_a = \langle n_0, n_0 + n_\delta, n_0 + 2n_\delta, \dots, N \rangle$, we use $n_0 = n_\delta = 10\%$ of N;
- **Geo:** the geometric method [33] generates samples according to the following plane: $S_g = \langle n_0, an_0, a^2 n_0, a^3 n_0, \dots, N \rangle$, we use $n_0 = 10\%$ of N and $a = 2$;

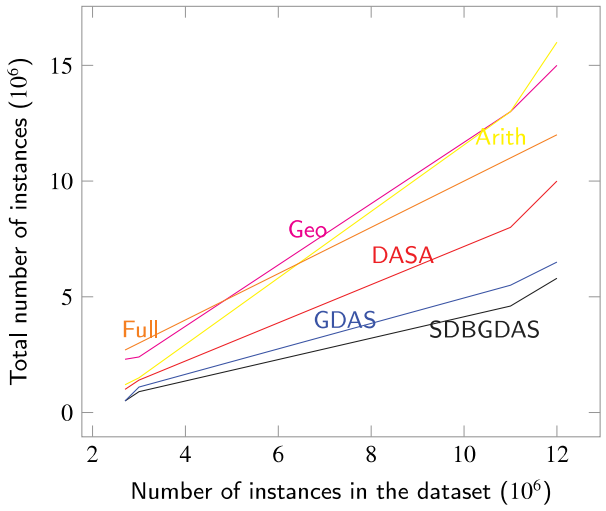

Fig. 5. The variation in the instances number used to achieve convergence according to dataset size.

- **DASA:** the adaptive method using Chernoff inequality [13] with $\epsilon = 0.01$ and $\delta = 0.05$;
- **GDAS:** adaptive method using Chebyshev inequality [16] with $\epsilon = 10^{-4}$ and $\delta = 10^{-3}$;
- **SDBGDAS:** the proposed method that is based on GDAS and uses the SDB and ScaSRS algorithms, with $\epsilon = 10^{-4}$ and $\delta = 10^{-3}$.

The comparative measures of the different approaches are:

1. **The mean computational time:** the runtime averaged over five (05) runs of each of the three data sets.
2. **The total number of instances needed to converge:** for a set of samples such as $D_1, D_2, \dots, D_k$ and the total instances number would be $|D_1| + |D_2| + \cdots + |D_k|$.
3. **The classification accuracy:** most important metric in our work is the classification accuracy achieved in the sample. For instance, this is the main objective of sampling from the whole population. It would be worthless sampling if the obtained instances do not allow the learning of reliable features that classify reliably the data. To this end, we will compute the precision of the models obtained by learning the final samples of each method. The accuracy will be computed from an independent test set.

For each sampling method, we calculated the number of instances to achieve the convergence of the learning curve, and this for each dataset, by using Naïve Bayes Classifier. The results are given in the Table 4.

According to the graph illustrated in Fig. 5, the SDBGDAS and GDAS methods give a relatively small number of instances compared with the DASA method and other static methods. For the various datasets, we noted the different running times of each process by using Naïve Bayes Classifier in Table 5.

According to the graph obtained in Fig. 6, we note that the SDBGDAS method takes a shorter processing time compared to the other methods. Also, we notice that the more instances used during learning the greater the difference are remarkable which confirms the effectiveness of the SDBGDAS method on large scales. The different details of the classification models represent a crucial point of this technique because good accuracy is a positive sign of the effectiveness of the proposed method, as shown in the Table 6 and Fig. 7.

The use of all instances during learning eventually gives better accuracy, and therefore logically the use of a set of instances among the total set of instances will decrease this accuracy. The SDBGDAS method gives the different datasets a better precision than the GDAS

Table 4
Instances number required for the different methods to reach convergence.

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
WatchGyr	2.7×10^6	1.2×10^6	2.3×10^6	10^6	5.1×10^5	5×10^5
WatchAccel	3×10^6	1.5×10^6	2.4×10^6	1.4×10^6	1.1×10^6	9×10^5
PhoneAccel	1.1×10^7	1.3×10^7	1.3×10^7	8×10^6	5.5×10^6	4.6×10^6
PhoneGyr	1.2×10^7	1.6×10^7	1.5×10^7	10^7	6.5×10^6	5.8×10^6
SyntheticData	1.0×10^7	9.4×10^6	9.0×10^6	6.2×10^6	4.9×10^6	4.2×10^6

Table 5
Average processing time for convergence.

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
WatchGyr	187	151	160	152	148	131
WatchAccel	196	162	171	160	154	142
PhoneAccel	867	689	797	603	511	362
PhoneGyr	973	793	882	714	602	454
SyntheticData	733	583	674	589	482	347

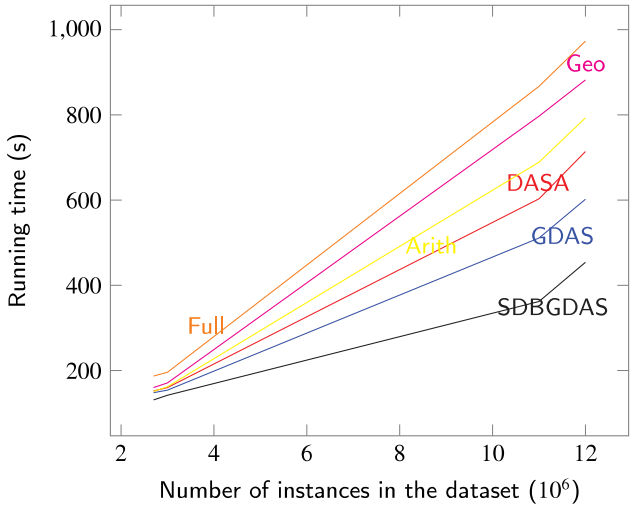

Fig. 6. The variation in running time according to dataset size.

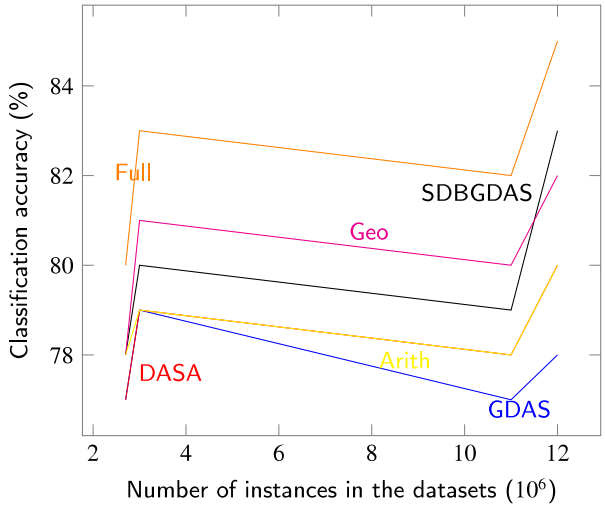

Fig. 7. The variation in classification accuracy.

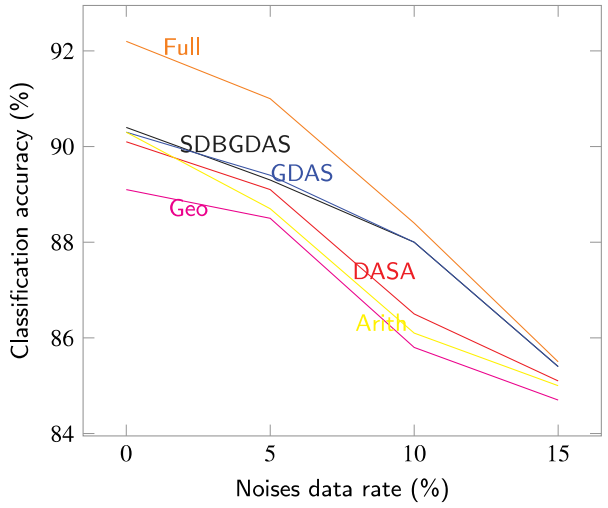

Fig. 8. The variation in classification accuracy according to noises data rates.

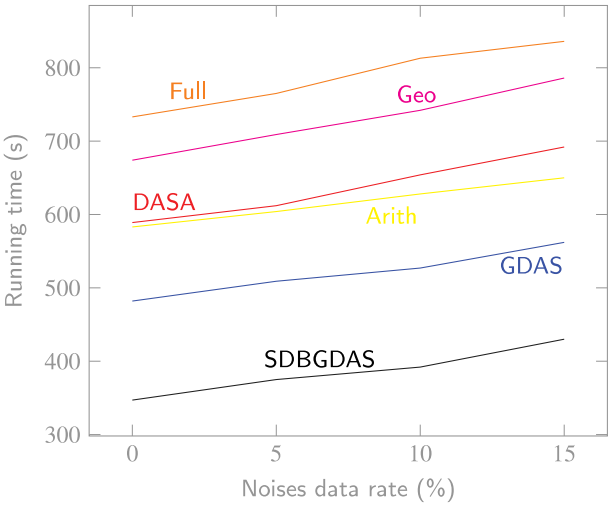

Fig. 9. The variation in running time according to noises data rates.

method and is closer to the precision of the full method which uses all the instances. Tables 7 and 8 show the behavior of the different sampling methods for the classifiers by testing on a synthetic dataset.

We evaluated the influence of noisy data rates on the different sampling methods by using the Naïve Bayes Classifier with synthetic data, the results are given in Tables 9–11. Figs. 8–10 show respectively the variations in accuracy, computation time and instances number to reach the convergence according to noises data rates.

The obtained results from tests carried on the influence of outliers data are given in Tables 12–14. Figs. 11–13 show respectively the variations in accuracy, computation time and instances number to

Table 6
Accuracies achieved for each dataset using Naïve Bayes Classifier (%).

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
WatchGyr	80.4 ± 2.6	78.8 ± 2.7	78.5 ± 2.5	77.2 ± 3.8	77.3 ± 2.3	78.7 ± 2.3
WatchAccel	83.2 ± 2.4	79.0 ± 2.3	81.3 ± 3.6	79.5 ± 3.9	79.1 ± 2.3	80.4 ± 2.1
PhoneAccel	82.2 ± 4.1	78.4 ± 4.3	80.1 ± 3.7	78.2 ± 3.2	77.6 ± 3.5	79.2 ± 3.4
PhoneGyr	85.0 ± 3.8	80.3 ± 3.5	82.1 ± 3.5	80.2 ± 2.7	78.4 ± 2.9	83.5 ± 3.5
SyntheticData	92.2 ± 2.9	90.3 ± 2.5	90.5 ± 2.1	90.1 ± 1.8	90.3 ± 2.1	90.4 ± 1.9

Table 7
Accuracies achieved for synthetic dataset by using various classifiers (%).

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
Naïve Bayes	92.2 ± 2.9	90.3 ± 2.5	90.5 ± 2.1	90.1 ± 1.8	90.3 ± 2.1	90.4 ± 1.9
Decision Trees	92.1 ± 2.7	90.5 ± 3.4	90.2 ± 3.2	90.7 ± 2.5	90.9 ± 1.8	91.0 ± 2.1
SVM	94.3 ± 2.5	92.2 ± 3.5	92.3 ± 2.8	92.0 ± 1.8	92.1 ± 2.4	92.2 ± 2.2
Logistic Regression	82.8 ± 3.5	80.5 ± 3.8	80.6 ± 3.9	80.9 ± 2.5	81.0 ± 1.6	81.0 ± 1.0

Table 8
Average processing time for convergence for synthetic dataset by using various classifiers (s)

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
Naïve Bayes	733	583	674	589	482	347
Decision Trees	1022	998	886	802	691	685
SVM	1352	1243	1181	969	920	895
Logistic Regression	682	503	505	485	441	390

Table 9
Accuracies achieved according to noises data rates (%).

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
Dataset	92.2 ± 2.9	90.3 ± 2.5	90.5 ± 2.1	90.1 ± 1.8	90.3 ± 2.1	90.4 ± 1.9
Dataset + 05%	91.0 ± 3.0	88.7 ± 3.4	88.5 ± 2.5	89.1 ± 2.0	89.4 ± 2.0	89.3 ± 1.9
Dataset + 10%	88.4 ± 3.2	86.1 ± 3.4	85.8 ± 3.5	86.5 ± 3.1	88.0 ± 2.5	88.0 ± 2.9
Dataset + 15%	85.5 ± 3.0	85.0 ± 4.2	84.7 ± 3.8	85.1 ± 2.9	85.4 ± 2.8	85.4 ± 3.2

Table 10
Average processing time for convergence according to noises data rates.

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
Dataset	733	583	674	589	482	347
Dataset + 05%	765	604	709	612	509	375
Dataset + 10%	813	628	742	654	527	392
Dataset + 15%	836	650	786	692	562	430

Table 11
Instances number required to reach convergence according to noises data rates.

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
Dataset	1.0×10^7	9.4×10^6	9.0×10^6	6.2×10^6	4.9×10^6	4.2×10^6
Dataset + 05%	1.05×10^7	9.7×10^7	9.5×10^6	7.4×10^6	5.3×10^6	4.8×10^6
Dataset + 10%	1.1×10^7	1.0×10^7	9.9×10^6	7.9×10^6	5.7×10^6	5.2×10^6
Dataset + 15%	1.15×10^7	1.1×10^7	1.1×10^7	8.7×10^6	6.0×10^6	5.9×10^6

reach the convergence according to outliers data rates (respectively 05%, 10% and 15%).

For each sampling method, we study its evolution in relation to the dimensionality of dataset. In fact, synthetic dataset that has the same instances number (10.000.000 instances) are used with different dimensions (8, 20, 50 and 100 features). Results are given in Tables 15–17. Figs. 14 15 and 16 illustrate respectively the variations in accuracy, computation time and instances number to reach the convergence according to different dimensions.

4.3. Discussion

Compared to the instances number included in the learning and creation of the classification model, we notice that the methods that use Chebyshev's inequality i.e. the GDAS and SDBGDAS methods require fewer instances than the methods that use Chernoff's inequality like the DASA method and also fewer instances than the progressive methods i.e. the arithmetic and geometric methods, this is because the latter requires several iterations to achieve convergence. The SDBGDAS

Table 12
Accuracies achieved according to outliers data rates (%).

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
Dataset	92.2 ± 2.9	90.3 ± 2.5	90.5 ± 2.1	90.1 ± 1.8	90.3 ± 2.1	90.4 ± 1.9
Dataset + 05%	91.2 ± 3.0	88.4 ± 2.7	88.7 ± 2.3	89.5 ± 2.5	90.0 ± 2.4	90.3 ± 2.0
Dataset + 10%	90.4 ± 3.2	87.0 ± 2.9	87.2 ± 2.5	88.7 ± 2.7	89.6 ± 3.0	89.9 ± 2.2
Dataset + 15%	89.7 ± 3.1	85.2 ± 3.0	85.5 ± 2.9	87.9 ± 2.8	89.1 ± 3.0	89.6 ± 2.3

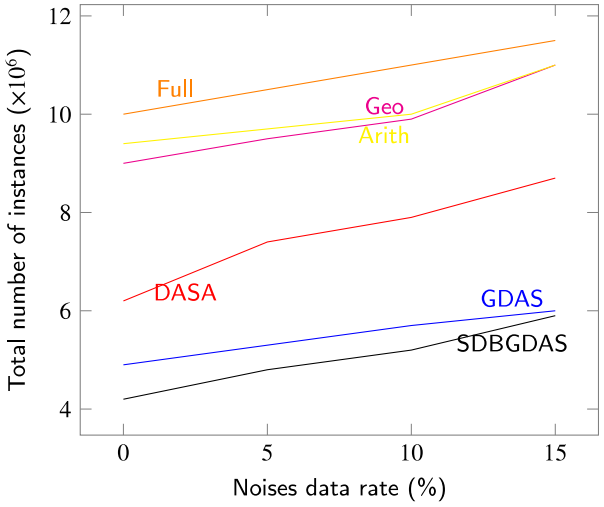

Fig. 10. The variation in the number of instances used to achieve convergence according to noises data rates.

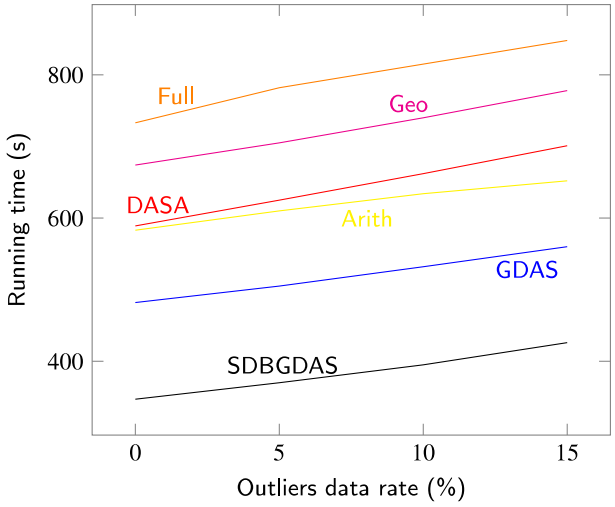

Fig. 12. The variation in running time according to outliers data rates.

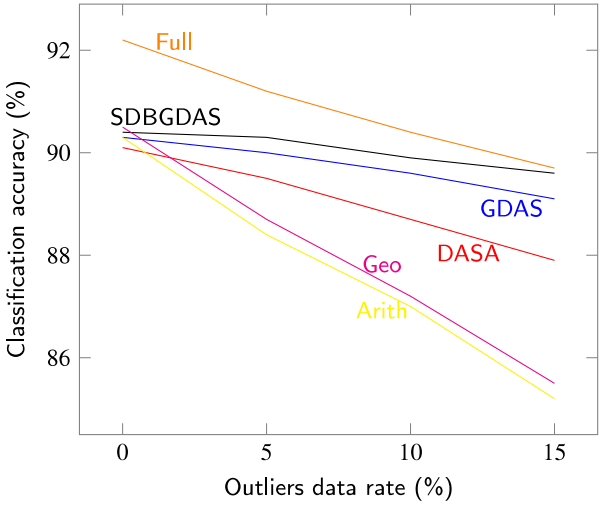

Fig. 11. The variation in classification accuracy according to outliers data rates.

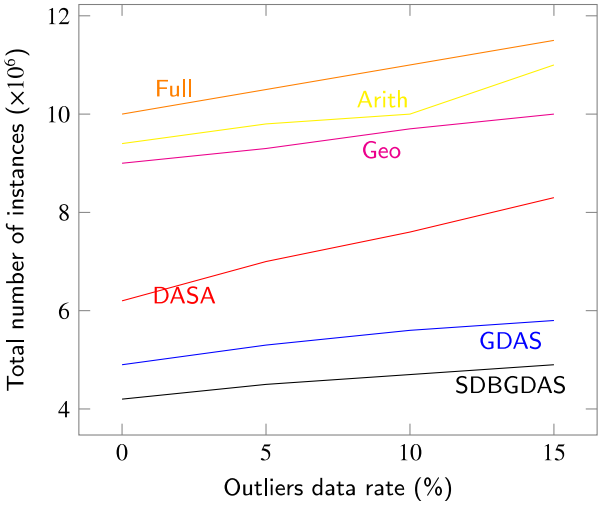

Fig. 13. The variation in the number of instances used to achieve convergence according to outliers data rates.

Table 13
Average processing time for convergence according to outliers data rates.

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
Dataset	733	583	674	589	482	347
Dataset + 05%	782	610	705	625	505	370
Dataset + 10%	815	634	740	662	532	395
Dataset + 15%	848	652	778	701	560	426

method is slightly better compared to GDAS in terms of the learning instances number, this is justified by the use of the SDB method that best estimates the variance of the samples taken against Bootstrap and BLB.

In terms of processing time, the full method as well as the arithmetic, geometric, and DASA methods took longer than the SDBGDAS methods, This comes back to the fact that progressive methods use more instances, hence more time, to achieve the learning. And although our implementations were done on a machine, the proposed SDBGDAS method gave better execution times compared to GDAS because it is based on ScalableSRS which is highly parallelizable, as well as SDB which presents better temporal complexity compared to BLB and bootstrap.

For the classification accuracy, it is clear that to achieve greater precision a maximum number of instances must be used, thus explaining why the full method using all instances gave better classification

Table 14
Instances number required to reach convergence according to outliers data rates.

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
Dataset	1.0×10^7	9.4×10^6	9.0×10^6	6.2×10^6	4.9×10^6	4.2×10^6
Dataset + 05%	1.05×10^7	9.8×10^6	9.3×10^6	7.0×10^6	5.3×10^6	4.5×10^6
Dataset + 10%	1.1×10^7	1.0×10^7	9.7×10^6	7.6×10^6	5.6×10^6	4.7×10^6
Dataset + 15%	1.15×10^7	1.1×10^7	1.0×10^7	8.3×10^6	5.8×10^6	4.9×10^6

Table 15
Accuracies achieved according to different dimensions (%).

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
08 Features	92.2 ± 2.9	90.3 ± 2.5	90.5 ± 2.1	90.1 ± 1.8	90.3 ± 2.1	90.4 ± 1.9
20 Features	92.5 ± 2.8	90.5 ± 2.4	90.6 ± 1.8	90.3 ± 2.1	90.5 ± 2.4	90.6 ± 1.5
50 Features	93.2 ± 2.1	90.8 ± 2.1	91.1 ± 2.3	90.7 ± 2.0	90.9 ± 1.9	91.1 ± 1.8
100 Features	94.3 ± 2.3	91.2 ± 2.2	91.4 ± 2.0	91.0 ± 2.4	91.4 ± 2.6	91.4 ± 2.0

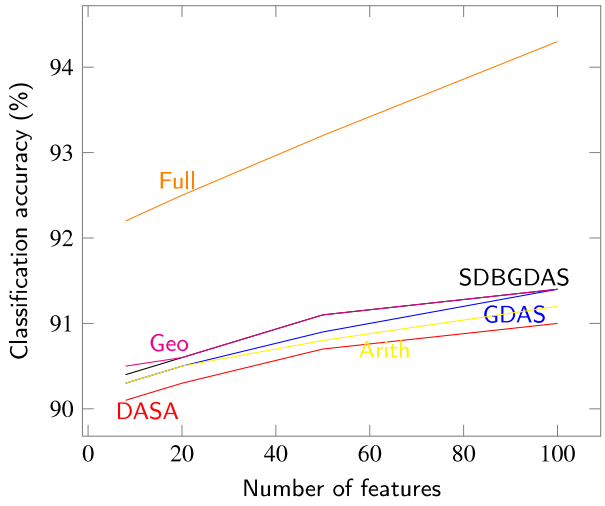

Fig. 14. The variation in classification accuracy according to different dimensions.

Table 16
Average processing time for convergence according to different dimensions.

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
08 Features	733	583	674	589	482	347
20 Features	781	625	689	607	520	381
50 Features	846	653	712	626	575	435
100 Features	902	689	736	645	651	482

results. However, the objective of the sampling methods is to reduce the number of instances used during learning while guaranteeing a minimum deviation inaccuracy. Arithmetic and geometric methods also have good accuracy since they use more instances, but SDBGDAS is better than GDAS since it uses the SDB method that provides better coverage of the selected instances. It thus uses active learning during selection which improves the quality of the chosen instances.

The results obtained from the sensitivity tests to noisy data show that the different sampling methods decrease in performance. However, GDAS and SDBGDAS perform well in terms of classification accuracy thanks to their variance estimation methods (Bootstrap and SDB respectively). Obtained results from the sensitivity tests to extreme outliers data prove that recent algorithms (i.e. DASA, GDAS, SDGDAS) are less sensitive to outliers data.

However, our method outperforms others in terms of running time and instances number required to achieve convergence. All methods gave almost the same results in the dimensionality tests, this reflects in this case that the increase of characteristics includes the increase of accuracy. On the other hand, SDBGDAS outperforms the state-of-art solutions in other evaluation measures.

The test of different sampling methods on the synthetic dataset with various classifiers (Naïve Bayes, Decision Trees, SVM, Logistic Regression) in terms of classification accuracy and execution time needed to reach convergence is carried out, the results obtained are illustrated in Tables 7 and 8. These prove that our algorithm is faster and more accurate than other state-of-art solutions regardless of the classifier used.

5. Conclusion

In the present work, a big data sampling algorithm is used for classification based on selecting an instances number from the totality of instances. This algorithm is introduced to make learning and classification model creation. The major challenge is to reduce the instances number by trying to optimize the sampling algorithms so that they guarantee a better quality of the selected instances and to try to use a minimum of instances that give better accuracy.

Our contributions in this paper include an intelligent sampling technique for data classification that is more suited for big data scales. This method uses the SDB technique to estimate the variance, the ScaSRS algorithm which allows more efficient scaling up, and the Active ScaSRS for better accuracy. We showed empirically that our algorithm outperforms other algorithms concerning the number of instances required and computational time needed.

Although our method gave good results for the numerical type of data. It presents a limit for our work which is intended only for big data classification tasks. As future work, we aim to extend the proposed approach for supporting any type of data including symbolic data. Moreover, our algorithm can be adapted to other machine learning tasks such as regression and clustering.

CRediT authorship contribution statement

Kheyreddine Djouzi: Acquisition of data, Analysis and/or interpretation of data, Original drafting of the manuscript, Reviewing and editing. **Kadda Beghdad-Bey:** Conception and design of study, Revising the manuscript critically for important intellectual content, Reviewing and editing. **Abdenour Amamra:** Supervision, Revising the manuscript critically for important intellectual content, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

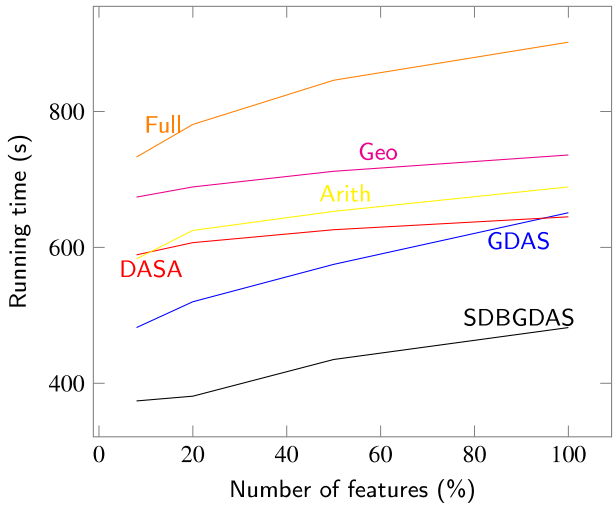

Fig. 15. The variation in running time according to different dimensions.

Table 17
Instances number required to reach convergence according different dimensions.

	Full	Arith	Geo	DASA	GDAS	SDBGDAS
08 Features	1.0×10^7	9.4×10^6	9.0×10^6	6.2×10^6	4.9×10^6	4.2×10^6
20 Features	1.0×10^7	9.4×10^7	8.9×10^6	6.4×10^6	4.9×10^6	4.3×10^6
50 Features	1.0×10^7	9.6×10^7	9.1×10^6	6.4×10^6	5.0×10^6	4.4×10^6
100 Features	1.0×10^7	9.7×10^7	9.3×10^7	6.5×10^6	5.0×10^6	4.4×10^6

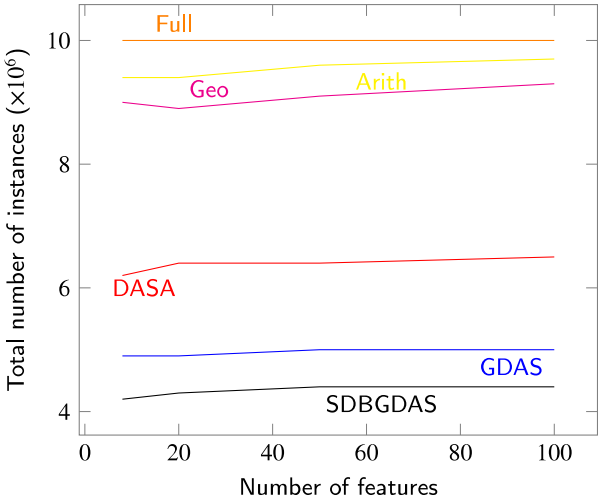

Fig. 16. The variation in the number of instances used to achieve convergence according to different dimensions.

References

[1] H.S. Munawar, S. Qayyum, F. Ullah, S. Sepasgozar, Big data and its applications in smart real estate and the disaster management life cycle: A systematic analysis, Big Data Cogn. Comput. 4 (2) (2020) 4.

[2] M.S. Gal, D.L. Rubinfeld, Data standardization, NYUL Rev. 94 (2019) 737.

[3] M.K. Saggi, S. Jain, A survey towards an integration of big data analytics to big insights for value-creation, Inf. Process. Manage. 54 (5) (2018) 758–790.

[4] B.N. Silva, M. Diyan, K. Han, Big data analytics, in: Deep Learning: Convergence to Big Data Analytics, Springer, 2019, pp. 13–30.

[5] Z.S. Ageed, S.R. Zeebaree, M.M. Sadeeq, S.F. Kak, H.S. Yahia, M.R. Mahmood, I.M. Ibrahim, Comprehensive survey of big data mining approaches in cloud systems, Qubahan Acad. J. 1 (2) (2021) 29–38.

[6] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, K. Stefanidis, An overview of end-to-end entity resolution for big data, ACM Comput. Surv. 53 (6) (2020) 1–42.

[7] J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García, F. Herrera, Big Data Preprocessing: Enabling Smart Data, Springer Nature, 2020.

[8] S. Singh, Simple random sampling, in: Advanced Sampling Theory with Applications, Springer, 2003, pp. 71–136.

[9] G. Sharma, Pros and cons of different sampling techniques, Int. J. Appl. Res. 3 (7) (2017) 749–752.

[10] W.A. Fuller, Sampling Statistics, vol. 560, John Wiley & Sons, 2011.

[11] X. Meng, Scalable simple random sampling and stratified sampling, in: International Conference on Machine Learning, 2013, pp. 531–539.

[12] G.H. John, P. Langley, Static versus dynamic sampling for data mining, in: KDD, vol. 96, 1996, pp. 367–370.

[13] A. Satyanarayana, I. Davidson, A dynamic adaptive sampling algorithm (dasa) for real world applications: Finger print recognition and face recognition, in: International Symposium on Methodologies for Intelligent Systems, Springer, 2005, pp. 631–640.

[14] G. SaravanaPrabhu, R. Vidjeapriya, Comparative analysis of learning curve models on construction productivity of diaphragm wall and pile, IOP Conf. Ser. Mater. Sci. Eng. 1197 (1) (2021) 012004.

[15] M. Huber, Halving the bounds for the Markov, Chebyshev, and Chernoff inequalities using smoothing, Amer. Math. Monthly 126 (10) (2019) 915–927.

[16] A. Satyanarayana, Intelligent sampling for big data using bootstrap sampling and chebyshev inequality, in: 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering, CCECE, IEEE, 2014, pp. 1–6.

[17] J. Navarro, A very simple proof of the multivariate Chebyshev's inequality, Comm. Statist. Theory Methods 45 (12) (2016) 3458–3463.

[18] Z. Mashreghi, D. Haziza, C. Léger, et al., A survey of bootstrap methods in finite population sampling, Stat. Surv. 10 (2016) 1–52.

[19] S. Xu, BayesIan naïve Bayes classifiers to text classification, J. Inf. Sci. 44 (1) (2018) 48–59.

[20] M. Scanagatta, A. Salmerón, F. Stella, A survey on Bayesian network structure learning from data, Prog. Artif. Intell. 8 (4) (2019) 425–439.

[21] Y.-Y. Song, L. Ying, Decision tree methods: applications for classification and prediction, Shanghai Arch. Psychiatry 27 (2) (2015) 130.

[22] D.M. Farid, M.A. Al-Mamun, B. Manderick, A. Nowe, An adaptive rule-based classifier for mining big biological data, Expert Syst. Appl. 64 (2016) 305–316.

[23] A.G. Howard, Some improvements on deep convolutional neural network based image classification, 2013, arXiv preprint arXiv:1312.5402.

[24] L. Connelly, Logistic regression, Medsurg Nurs. 29 (5) (2020) 353–354.

[25] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, A. Lopez, A comprehensive survey on support vector machine classification: Applications, challenges and trends, Neurocomputing 408 (2020) 189–215.

[26] P.J. Bickel, F. Götze, W.R. van Zwet, Resampling fewer than n observations: gains, losses, and remedies for losses, in: Selected Works of Willem Van Zwet, Springer, 2012, pp. 267–297.

[27] S. Sengupta, S. Volgushev, X. Shao, A subsampled double bootstrap for massive data, J. Amer. Statist. Assoc. 111 (515) (2016) 1222–1232.

[28] A. Kleiner, A. Talwalkar, P. Sarkar, M.I. Jordan, A scalable bootstrap for massive data, J. R. Stat. Soc. Ser. B Stat. Methodol. 76 (4) (2014) 795–816.
[29] G. Nikolov, A. Shadrin, Markov L_2-inequality with the laguerre weight, 2017, arXiv preprint arXiv:1705.03824.
[30] F. van Workum, M.H. Stenstra, G.H. Berkelmans, A.E. Slaman, M.I. van Berge Henegouwen, S.S. Gisbertz, F.J. van den Wildenberg, F. Polat, T. Irino, M. Nilsson, et al., Learning curve and associated morbidity of minimally invasive esophagectomy: a retrospective multicenter study, Ann. Surg. 269 (1) (2019) 88–94.
[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[32] V. Vashishth, A. Chhabra, D.K. Sharma, Gmmr: A Gaussian mixture model based unsupervised machine learning approach for optimal routing in opportunistic IoT networks, Comput. Commun. 134 (2019) 138–148.
[33] F. Provost, D. Jensen, T. Oates, Efficient progressive sampling, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 1999, pp. 23–32.

Kheyreddine Djouzi received the master's degrees from Mouloud MAMMERI Tizi-Ouzou University, Algeria, in 2017, in computer science. Since 2018, he has been a Ph.D student at the Ecole Militaire Polytechnique (EMP), Algeria. His research interests include big data classification. Now, he is working on efficient sampling techniques for big data classification.

Kadda Beghdad-Bey has done his post-graduation in Industrial Computer Science from Ecole Militaire Polytechnique (EMP) in 2003. He received his Ph.D. in Computer Science from USTHB University in 2010. He is currently a professor and the Head, Unit of research computer science at EMP. His research interests are in the areas of Grid computing, Biometrics identification systems, Parallel and distributed computing, and resources allocation for task scheduling in Cloud Computing.

Abdenour Amamra received his computer engineering degree from Ecole Militaire Polytechnique, Algeria, in 2011. He was a fellow researcher and Ph.D. student at Cranfield University, UK, from 2012 to 2015. He received a Ph.D. in computer science for a thesis on "Robust 3d Registration and Tracking with RGBD Sensors". He is currently a lecturer and researcher at Ecole Militaire Polytechnique, Algeria. His current research interests include computer vision, deep learning, and computer graphics.