

On the efficiency of user identification: a system-based approach

Apostolos Malatras¹ · Dimitris Geneiatakis² · Ioannis Vakalis¹

Published online: 5 July 2016

© The Author(s) 2016. This article is published with open access at Springerlink.com

Abstract In the Internet era, users' fundamental privacy and anonymity rights have received significant research and regulatory attention. This is not only a result of the exponential growth of data that users generate when accomplishing their daily task by means of computing devices with advanced capabilities, but also because of inherent data properties that allow them to be linked with a real or soft identity. Service providers exploit these facts for user monitoring and identification, albeit impacting users' anonymity, based mainly on personal identifiable information or on sensors that generate unique data to provide personalized services. In this paper, we report on the feasibility of user identification using general system features like memory, CPU and network data, as provided by the underlying operating system. We provide a general framework based on supervised machine learning algorithms both for distinguishing users and informing them about their anonymity exposure. We conduct a series of experiments to collect trial datasets for users' engagement on a shared computing platform. We evaluate various well-known classifiers in terms of their effectiveness in distinguishing users, and we perform a sensitivity analysis of their configuration setup to discover optimal settings under diverse conditions. Furthermore, we examine the bounds of

sampling data to eliminate the chances of user identification and thus promote anonymity. Overall results show that under certain configurations users' anonymity can be preserved, while in other cases users' identification can be inferred with high accuracy, without relying on personal identifiable information.

Keywords User identification · Anonymity · Machine learning

1 Introduction

The proliferation of online applications and services available to the end users combined with their continuous, high-frequency usage have led to an explosion in the amount of data generated by the users that are subject to monitoring and recording. This ever-increasing amount of highly granular data that refer to personal and private user activities poses a great risk to user privacy since it can be exploited to potentially reveal their activities and threaten their anonymity.

Various research works [1–3] determine the basic threats against users' privacy and anonymity, when accessing electronic services. However, not only personal identifiable information (PII) such as IP, e-mail addresses and cookies [4] can be used to monitor and identify a user, but also information produced by general system elements and sensors e.g., browser fonts, GPS and accelerometer, as reported in [5–7]. For instance, GPS sensors generate unique data that can be easily exploited for user identification [8]. Even inconspicuous sensors' data such as those coming from accelerometers and magnetometers can be exploited to distinguish users [7, 9, 10] using machine learning techniques. Additionally, other approaches operate on users' microdata to perform user identification [11].

Dimitris Geneiatakis was within EC, JRC while this work was taking place.

✉ Dimitris Geneiatakis
dgeneiat@auth.gr

¹ European Commission, Joint Research Centre (JRC),
Institute for the Protection and Security of the Citizen,
21027 Ispra, VA, Italy

² Electrical and Computer Engineering Department,
Aristotle University of Thessaloniki, 541 24 Thessaloniki,
Greece

Such techniques might not be suitable for user identification on devices lacking specific sensors as the aforementioned ones. Conversely, existing approaches do not distinguish users based on system inherent features, while additionally the identification of users who share computational resources, e.g., PC, phone, with others has hitherto received minimal attention. Analyzing such data to extract useful patterns and accordingly using them to identify distinct users sharing a PC or a mobile phone based on their usage behavior might threaten their anonymity on the one side, while orthogonally on the other side it can be used in the fight against identity theft. In this direction, our goal is dual in providing facilities to distinguish users using computing systems by means of monitored general system properties, as well as in protecting their anonymity.

We postulate that it is feasible to develop frameworks based on machine learning techniques that would allow computing systems of inferring who is using them by making use of general system properties such as utilization of memory, CPU and network stats. This is because all the existing computational systems have these features available as they hold a CPU unit, memory and network connection. We aim to deliver tools indicating to users that during their activity on a computing device, utilization patterns can lead to their identification and consequently exposing their online anonymity. In addition, we examine complementary aspects of using such tools to provide users with a *continuous authentication service* allowing them to be constantly authenticated based on their computational patterns, instead of relying only on other forms of credentials i.e., username and password. Thus we are moving in the direction of enhancing users' identity protection when acting online.

We report here on the feasibility and efficiency of user identification relying on non-PII system features. Accordingly, we employed a real-time monitor to collect memory, CPU and network stats in regard to specific users. By applying well-established data processing and classification techniques, we study the effectiveness of supervised machine learning classifiers to identify and distinguish the monitored users. Relevant results are extremely promising and show that users can be identified with accuracy of up to 98 % depending on the employed algorithm. *To the best of our knowledge, this is the first work focus on the effectiveness of users identification that relies on non-PII.* Moreover, we study through a sensitivity analysis the effect of various parameters on the efficiency of the classification process. In this way, we acquire a set of indicators that can be utilized to protect user anonymity (e.g., by reducing the sampling rate of the monitoring process) or constitute user authentication more stringent (e.g., require higher accuracy rates).

The main contributions of this work can be summarized in the following:

- (a) We introduce an approach for user identification using non-PII. Results show that we can achieve high accuracy, in particular when we rely on memory datasets.
- (b) This is the first work that reports on the effectiveness of users identification that relies on non-PII. The detailed analysis of results also yielded a set of novel indicators as to the configuration settings for users' continuous authentication and anonymity.
- (c) Our solution's software as well as the experimental dataset is freely available.¹ This way, we not only facilitate possible extensions of this work, but also encourage other researchers to validate and contribute to our results.

The rest of the paper is structured as follows. In Sects. 2 and 3, we briefly describe the problem space of our work and the related work, respectively. In Sect. 4, we introduce our proposed framework architecture for user identification. In Sect. 5, we discuss the details of the machine learning process we undertook to conduct our experiments. In Sect. 6, we assess our approach in terms of accuracy, while additionally we report on the sensitivity analysis of all considered parameters. In Sect. 7, we present a discussion on the experimental results and discuss the optimal uses of our approach. Finally, we conclude our work presenting also some pointers for future work in Sect. 8.

2 Problem statement and motivation

The goal of this work is to establish a framework for user identification based on the monitoring of system properties and characteristics e.g., memory and CPU. Moreover, as such, this type of data not considered as being "personal" might be mistakenly assumed that it does not threaten users' anonymity. Note that as anonymity is *the state of being not identifiable within a set of subjects, the anonymity set* [12], it is really important to define also the bounds of system-based features required to be collected in order to guarantee end users' anonymity in a systematic way. To the best of our knowledge, none of the existing works in the related literature have hitherto pursued this area of study.

Identifying users based on non-PII information is a much more challenging task compared to using traditional PII. This is because PII refers to personally identifiable information and is tightly linked to the identity of a person. It can be as simple as the passport number of a person or IP address of a user's device, but also combinations of other information. Non-PII on the other hand is the information that at first glance does not yield any hint on the identity of a person. At this point, we should mention that even though information such as heartbeat can be used in biometric-based authentication systems [13] and user monitoring as well, it requires

¹ <https://github.com/dgeneiatakis/user-identification/>.

the adoption of specific devices and users' explicit consent. Furthermore, such type of information is sensitive from its nature.

General system properties as the ones that we have considered in this paper cannot be easily and in a straightforward manner traced back to a user's identity. First, using non-PII involves the processing of significant amounts of data and is not as straightforward as using PII data. Second, it would require some piece of PII to be able to actually have exact user identification. Conversely, what non-PII information yields is the generic identity of a user, e.g., user A instead of John Doe, and it is this privacy-friendly approach that supports our continuous user identification goal via the proposed framework.

Such a framework can have a wide applicability and promote users' security, and orthogonally, their anonymity in the direction of the following aspects:

- (a) Enforcing user re-authentication when behavior out of the ordinary is detected.
- (b) Inform end users about their anonymity exposure over time.
- (c) Provide a tool to end users to control information that might affect their anonymity.

The motivation behind the use of such system properties lies in the fact that they are always available and they reflect the activity of every person using a computing device, taking also in consideration the always online status of users in Internet era. This constant availability of information can therefore be utilized to support *continuous authentication* in a privacy-friendly manner, and building on this, it can accordingly serve the aforementioned aspects.

The aforementioned aspects are in most of the cases complementary to each other. For example, user authentication might be reinforced when the normal pattern of users' behavior is not observed, thus implying that another identity is interacting with the monitored system. In parallel, users might be informed that their behavior is subject to loss of anonymity and accordingly to propose ways to protect it. Such an approach is followed by the PrivateEye,² software to automatically protect users' privacy by relying on a webcam and attention sensing technology. In this respect, adversaries cannot "eavesdrop" on personal data by furtively looking in one's monitor.

Furthermore, another potential use of such an approach involves the case of multiple users sharing a PC e.g., in an Internet cafe, an airport and a company laptop. Under these conditions, a scenario might require the need to distinguish between users, as illustrated in Fig. 1, who were operating on the common PC in order to facilitate, for example, some criminal investigations, i.e., due to illegal activities. Therefore,

² <http://www.privateeyesoftware.com>, accessed December 2014.

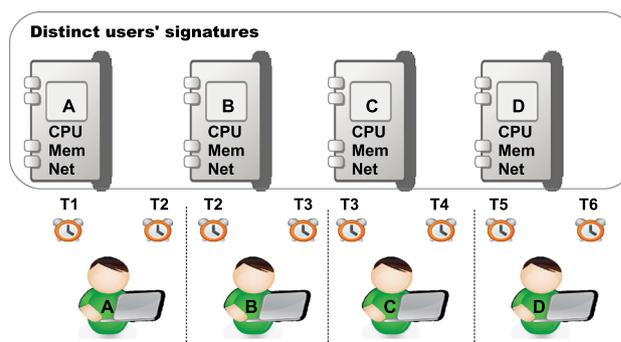


Fig. 1 An example of resource sharing among different users over time

information stemming from system monitoring that could serve to identify distinct users can be construed—subject to necessary processing—as digital forensics evidence in security operations.

3 Related work

The work presented in this paper aims to address whether it is possible to identify or alternatively profile a user by monitoring general-purpose system's elements such as CPU, memory and networking. Whereas we could identify various research areas that are related to our current work such as malware detection through monitoring systems features [14–17], we review here only the approaches focusing on user identification in the direction of developing a set of distinct signatures or signals to identify a user.

3.1 Web browsing

Yang et al. [18] build a profile model for user identification based on users' web browsing behavior patterns. In this respect, users' activities are monitored to extract the strongest indicators that can be used to develop a profile. According to the presented results, the proposed approach performs more efficiently, compared to decision trees and support vector machine classifiers. Herrmann et al. [19] proposed to observe over a period of time web user activities and to rely on Multinomial Naive Bayes classifiers to identify different users. In an alternative approach, Xu et al. [20] monitor users HTTP requests building a model of their mouse clicks based on Hidden Semi-Markov model, hence achieving an accuracy of 93 %.

Melnikov et al. [21] introduce the notion of *cybermetric*—as the sequencing of the performing steps at each new browsing session—for user identification through the analysis of network flows. Kapusta et al. [22] focus on time value estimation that should be computed for user identification approaches that rely on time threshold techniques [23]. In

the same direction, Yang et al. [24] quantify the *amount* of data, in terms of web sessions, required to achieve better results on user identification.

3.2 Computer resource monitoring

CPU, memory and other performance counters are fundamental mechanisms used to monitor the current status of a computing device. Such a monitoring also has the advantage that it can be conducted in real time dynamically, and therefore, it allows for reactive decision making in response to certain actions and in order to accomplish various tasks. In the previous discussion on the use of web browsing techniques, we reported on monitoring users' web activities with the objective to investigate to what extent the collected information can be utilized for identification of the user and his/her activities. Conversely, Abrahao et al. [25] study applications' workload characteristics to enable an efficient resource management to the provided applications. Along the same lines, Gong et al. [26] introduce a framework based on fast Fourier transform (FFT) [27] and dynamic time warping (DTW) [28] to predict resource usage on cloud computing platforms. Moreover, Zhang et al. [29] develop a methodology to improve scheduling and overall performance in heterogeneous computing. This approach uses the principal component analysis (PCA) and the k-nearest neighbor (k-NN) classifier to characterize applications' monitoring characteristics such as CPU-intensive, I/O and paging-intensive and network-intensive. The experimental results, for a set of three benchmark applications, demonstrate that the acquired application class information improves the overall scheduling decisions and system throughput.

3.3 Phone sensors and touch screens

User identification using system properties has found great applicability on smartphones and mobile devices due to the abundance of sensors that they are carrying and the need for innovative continuous authentication techniques. Accordingly, Feng et al. [14] introduce a user identification framework on android devices based on user's gestures that can be used to trigger user's authentication when a high deviation from normal behavior has been detected. A detailed evaluation of this scheme in the case of uncontrolled environments showed that the proposed framework's accuracy reaches levels of up to 90 %. Shi et al. [7] rely on users' gesture patterns using, however, an n-gram [30] model. In this case, an accuracy of 75 % is achieved. In addition, Blaica et al. [31] focus on user identification on multitouch displays with the ability to detect five or more touchpoints. The authors assess well-known classifiers and report results on achieving accuracy between 92 and 96 %. Zahid et al. [32] propose a user identification scheme based

on keystroke information of mobile devices. In this approach, an accuracy of 98 % is reported. Weiss et al. [9] collect data from the mobile phone's accelerometer for user identification using supervised machine learning classifiers. A similar approach is followed by [10] taking into account the fact that user movements as recorded by the accelerometer sensor are inherently unique. Pan et al. [33] introduce a method to utilize a camera sensor to provide a more secure identification for mobile users. Specifically, the proposed solution extracts and analyzes geometrical features from a human hand to identify a user. In the latter solution, authors achieve accuracy up to 94 %. Mock et al. [34] rely on grayscale images retrieved from the touch screen. Authors use multi-class support vector machine learning algorithms and manages to achieve an accuracy of 97 %.

Based on the aforementioned review of related research works, we assert that there is a need to establish efficient, privacy-friendly user identification techniques that do not utilize personal identifiable information. Thorough investigation on the bounds of such techniques should also be conducted in order to effectively identify users or inversely inform them on their anonymity exposure levels. These motivations form the foundation of our proposed work as discussed in the following.

4 System architecture

We designed a functional architecture as illustrated in Fig. 2 based on which we proceeded in its development to facilitate experimental evaluation on the one hand and validate the realistic deployment of our proposed system on the other hand. Users interact with computing devices and conduct their regular activities. With the use of the services provided by the SIGAR API,³ we are able to record the values of certain computing properties, e.g., CPU, memory, network, and monitor their variation as a result of these activities. The architecture is consisted of three main elements, namely the *Feature Monitor*, the *Information Manager* and the *Classification Manager*.

Specifically, the *Feature Monitor* is responsible for collecting the low-level data from the monitored computer platform. Because the collected data have different formats and monitoring requirements, e.g., collection frequency and granularity, we devised dedicated handlers for each feature type, i.e., CPU, network and memory. Each of them has a listener set to the corresponding SIGAR service, thus constantly acquiring data on the computing property it refers to. The *Data Wrapper* is a subcomponent of the *Feature Monitor* that collects data from the dedicated handlers and formats them in a unified way by removing excess information and

³ <http://www.hyperic.com/>.

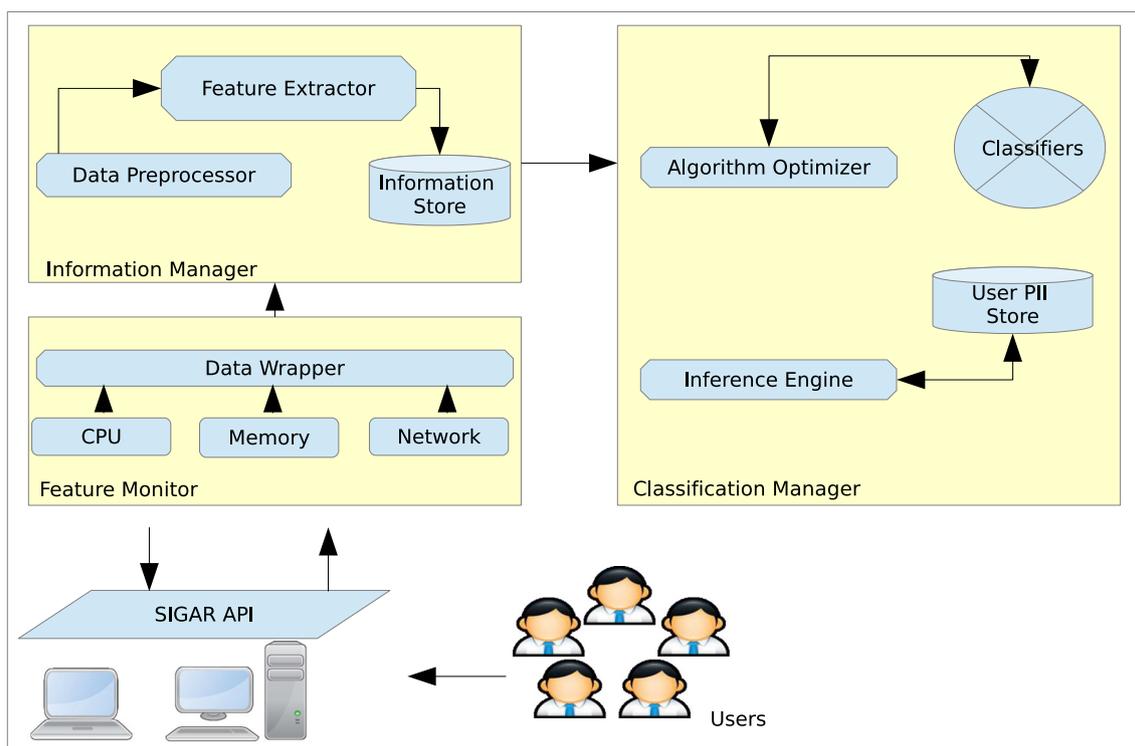


Fig. 2 Functional architecture for user identification system

unnecessary verbosity. It also provides interfaces for other elements of our architecture to query for and access the monitored data.

The functionality of the *Information Manager* mainly reflects the preprocessing step in machine learning classification. The raw data coming from the *Data Handler* cannot be used as such to identify individual users. This is (a) because the raw data represent values that are bound to the local device semantics, and (b) due to the large scale of the collected data. Accordingly, the *Data Preprocessor* preprocesses the collected data prior to feeding it to the *Classification Manager*. The aim of this step is to cleanse the collected dataset, to remove outliers, erroneous entries and prepare the data for the *Feature Extractor* which extracts the corresponding statistical features.

In addition, the *Data Preprocessor* also applies a windowing technique to extract groups of data that could expose repetitive activities or tasks with common characteristics. It is not realistic to assume that machine learning techniques will infer something useful about the data if we are to consider every single collected instance. On one side, the windowing process is used to reduce the size of the collected dataset and on the other side to allow machine learning techniques to be applied on subsets of data that might yield representative information about users' activities. The window instances are passed, as well, on to the *Feature Extractor* that employs time- and frequency-domain processing on the datasets to

extract the appropriate features. The latter constitute possible PII for users and are stored in the *Information Store* repository. It should be noted that this information lies at the core of user identification.

The *Classification Manager* is where the core of our user identification process takes place. It utilizes the preprocessed features stored in the *Information Store* and retrofits the classification process—meaning the machine learning classifiers. Assuming the training phase of the classification process has been completed, the different available *classifiers* will have been subject to a thorough sensitivity analysis to establish their suitability in handling and processing a variety of collected data. It is evident that there cannot be universal parameter settings for each classifier over all the collected data. The *Algorithm Optimizer* selects the most appropriate classifier according to the types of collected data and applies the optimal set of configuration parameter as a result of the aforementioned analysis. The optimized data are forwarded to the *Inference Engine* component that is responsible to distinguish users' entities, at a specific time window, with a certain confidence level—meaning that it infers which user is active on the computing platform at a given time.

The *Algorithm Optimizer* has the additional functionality of continuously updating the classifiers in order for them to adapt to the new data generated by users. As mentioned before, the prototype version of our framework involves a sole training phase. However, this approach does not take

into account potential variations in users' patterns over the course of time. For this reason and in order to fully satisfy our goal for *continuous authentication*, we plan to extend the framework to incorporate a constant training phase, namely to allow the framework to continue learning about users' patterns of behavior while operating. This reactive mechanism will augment the framework in providing an evolving, adaptive user identification process.

5 User identification approach

We describe in the following the methodology that we have adopted to realize our proposed user identification approach, which is based on machine learning techniques. Accordingly, we also discuss feature extraction techniques and the classification algorithms that we have considered, along with the configuration of the experiments that we undertook to validate the performance of our approach.

5.1 Methodology

To identify distinct users of any computation device (e.g., PC, mobile phone) based on their behavior and usage patterns, we analyze monitored data regarding computational properties of the device that is being used. The methodology that we have employed to do so is illustrated in Fig. 3. It involves a typical machine learning three-step iterative process, namely data collection, processing and classification for both training and test phases.

Briefly, during the training phase, the classifier is fed with sample data to train it in order to be able to characterize unclassified (test) data at the test phase. This is an iterative approach in which classifier's accuracy is evaluated against the ground truth, while additional modifications and parameters reconfigurations might be needed based on the irregularities between training and test data. Moreover, the process could be repetitive in order to take into account new data regarding users and utilize them to optimize and adapt the classification process toward *continuous authentication*.

It should be noted that while for our experimental analysis we used the same PC for all users, our approach is not limited by this factor. Our work relies on general system-based properties such as those related to memory, CPU, TCP MIB and network statistics without considering any hardware or operating system-specific feature. Evidently, such system

properties/attributes are not subjected to be dependent on the particular PC used since they do not refer to fixed values, but instead on values that are relative to the system's operation such as the number of TCP connections and used memory.

Data Collection is performed in our system with the use of the services provided by the SIGAR API. There is a great number of system properties that can be monitored, not all of which, however, are useful for pattern recognition since they do not exhibit any characteristic variation. We performed a thorough analysis of all properties that can be observed by SIGAR and pruned the observed dataset to include only those ones that entail noteworthy variability such as CPU, memory and network-related information.

Data Processing tasks involve *data validation*, the removal of outliers from the original dataset, data annotation for classification purposes (this applies only to training data, since the classifier will predict the class of the test data) and the extraction of the features. In terms of data validation, prior to their processing we apply two types of validation, namely data type validation in which each of the monitored attributes is expected to return a value with a particular data type and constraint validation meaning that minimum and maximum values are checked to ensure that only valid values are recorded. Moreover, we apply a data filtering method to ensure that no empty values are recorded, due to error in the monitoring process or for other reasons. In addition, raw data have usually very fine granularity and it is difficult therefore to discern statistical properties on them. Therefore, it needs to be processed in order to extract statistical features over time and frequency domains.

Data Classification refers to the application of classification algorithms on the processed data in order to infer to which class they belong to, namely to which user they refer to. Each classifier has a set of configuration parameters and a sensitivity analysis of each of them, and their influence on the accuracy of the classifier needs to be performed. This is required in order to conclude on the most appropriate classifier for the considered experiments settings. Having decided on the optimal classifier, it needs to be applied on the collected test data (a posteriori or at runtime depending on the experimental settings). The classifier will be used to identify the class to which each of the test data belong (namely which user was actively using the monitored computer), and it decides upon that based on the similarity and pattern matches of the test data features with the training data features.

5.2 Feature extraction

The original training dataset is a high-frequency dataset, and this constitutes a problem when its processing is concerned. Moreover, the sampling frequency is too high for individual samples to exhibit any interesting properties, especially in

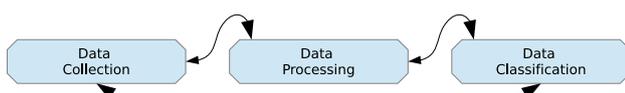


Fig. 3 Methodological approach for user identification

regard to user identification, since one cannot realistically expect that it would be feasible to identify a distinct user working on a machine subject to a mere snapshot of that machine's operation. To alleviate such concerns, we initially apply a sliding window scheme with 50 % overlap on the original dataset to create logical instances. This way we do not only reduce the problem space, but we also assist in grouping similar samples.

Subsequently, the feature extraction process is performed over different data windows sizes to further reduce scalability issues, as well as to provide useful qualitative and quantitative semantics information related to their statistical properties. In this work, we have considered time- and frequency-domain features. The latter necessitated that we first apply a Fast Fourier Transform (FFT) over the windows' data and then calculate the corresponding features. data windows

We calculate the following time-domain features: (a) mean, (b) geometric mean, (c) median, (d) minimum, (e) maximum, (f) variance, (g) standard deviation and (h) root-mean-square, for each of the data windows. We also consider computing pairwise, for all parameters, the values for covariance and Pearson's correlation. For the frequency-domain features, we calculated the (a) mean, (b) the dominant component (DC), (c) median, (d) minimum, (e) maximum and (f) the spectral energy.

5.3 Classifiers

There is a great variety of machine learning classifiers, each one with distinct features and properties that affect its behavior. We do not go into details about machine learning classifiers, and we nonetheless refer the interested reader to relevant state-of-the-art reviews such as [35,36]. In this work, we considered the following well-known classifiers,⁴ namely machine learning classification algorithms, to apply in the extracted dataset:

J48 It is a C4.5 decision tree classifier with a variety of configuration parameters, such as whether to use binary splits on nominal attributes when building the trees, the confidence factor used for pruning and the minimum number of instances per leaf to mention a few [37].

Decision Table (DT) It has a default rule that maps an unidentified instance to the majority class based on an evaluation metric, which is used to assess the performance of attribute combinations used in the decision table, i.e., accuracy, root-mean-square error, area under ROC and mean absolute error [38].

Bayes Networks (BN) It is based on Bayes Network learning and utilizes a variety of quality measures and search algorithms such as genetic search, simulated annealing and hill climber [39].

Support Vector Machine (SVM) It is based on supervised learning techniques and classifies training data instances into two possible output classes; it is therefore a non-probabilistic, binary and linear classifier [40].

Sequential Minimal Optimization (SMO) SMO is used to train an SVM in a simplified manner and address the optimization problem present in training SVMs using traditional techniques [41].

k-NN It is an instance-based learning algorithm, namely it compares new problem instances with instances seen in training. The number k indicates how many "close by" instances in the training should be used, which have been stored in memory [42].

Multilayer Perceptron neural network (MLP) It is based on a neural network that is using supervised learning techniques and a back-propagation algorithm to assign instances to the appropriate classes [43].

5.4 Experimental configuration

To study the efficiency of the aforementioned classifiers in regard to user identification when we rely on general system resources, we experimented with four users sharing the same PC for three sessions of 1 h each, using the schema illustrated in Fig. 1. Moreover, for the purposes of our experiments, we deployed an application based on the SIGAR API, as mentioned previously, to audit specific system properties. In particular, we monitor the following system properties, which are the attributes of the considered datasets:

- (a) CPU total system kernel, user and input/output wait times, as well as total time CPU is servicing softirqs.
- (b) Total system's free and used memory, actual system free and used memory, as well as percentage of total free and used system memory.
- (c) TCP active and current established connections, the total number of data segments received, sent and retransmitted, the number of established connection resets, as provided through the TCP MIB service.
- (d) Additional network statistics such as total outbound traffic, TCP outbound traffic and the number of connections in TCP close-wait, established and listen status.

Accordingly, because of the nature of the collected attributes, it is evident that the datasets are comprised of integer values. Moreover, the majority of the collected attributes in the datasets are cumulative, namely their value grows over time with respect to the monitored system property. Statistical properties of certain attributes exhibit particular interest. For example, the percentage of total used memory for all users has a mean of 32.2413 and a standard deviation of 10.4989, thereby illustrating noteworthy variability. It should be noted that this variability is due to the different way users

⁴ The selection being made to accommodate classifier diversity.

are interacting with the computer, since for a single user in our experiments the standard deviation was as low as 2.9541 with a mean of 19.4678. The same attribute has a low positive skewness value of 0.1534 and a kurtosis value of -0.9208 indicating a rather flat distribution.

While it would have been possible to monitor a greater variety of system properties and after careful examination of data characteristics, we opted to discard certain properties since they do not exhibit any type of variation and can lead to data leakages [44, 45] and therefore influence the results. System properties that were not considered in our work include:

- The total system CPU idle, involuntary wait and nice time, the total system CPU time spent in servicing interrupts.
- The total system RAM.
- The number of connection attempt that led to failure, passively opened TCP connections and the total number of segments received in error in the TCP MIB.
- Network statistics in regard to the number of all inbound connections, TCP bound connections and TCP connections in one of the CLOSE, CLOSING, IDLE or LAST_ACK state.

During each session, users were instructed to conduct activities as per their normal, daily usage. We study the influence that different parameters such as window size, sampling rate and the number of users have on the classifiers' accuracy. For all of our tests, we rely on the same amount of overlap i.e., 50 % between instances. For each of the previously mentioned system properties, we compute a series of statistical features, as discussed in Sect. 5.2 over the instances comprising each data window to be used for classification purposes. We need nonetheless to underline that a sensitivity analysis of all these parameters has been conducted to account for their influence on the performance of the algorithms and relevant results are presented in the following.

For classification and testing purposes, we used the Weka [46] open-source toolkit that supports a rich set of machine learning classifiers and can be easily integrated in other programs. Relevant results from our experimental evaluation are reported in the Sect. 6. Note that missing bars in graphs indicate the inability of certain classifiers to be applied on the corresponding dataset due to the properties and features of the latter.

6 Experimental results

To test the accuracy and efficiency of the classifiers presented in the previous section, we performed a comparative analysis of different configurations and applied them on the diverse monitored datasets. There are two aspects to validate with

regard to our work, namely the validity of (a) the collected datasets, and (b) the classification model and process itself.

As far as the first aspect is concerned, we rely on the well-established and highly used SIGAR API to monitor and collect system properties. Thus, we assume that the SIGAR API works according to its specifications, and therefore, accurate and valid data is being collected. To further promote validity of the collected data, we employed data type and constraint validation check, as already discussed in Sect. 5.1.

To ensure the validity of our experimental classification process and the corresponding results, we used cross-validation with tenfolds for reducing variability as well. Moreover, we executed each experiment for ten iterations to exclude statistical bias, while to evaluate the accuracy of the classifiers we split the monitored data into a training set (66 %) and a test one (33 %), as it is regarded the standard method even for the cases when "limited" data are available [36]. To gain a level of statistical confidence in the presented results, we applied the statistical hypothesis test Students' *T* test with a requested confidence of 0.05. This indicates a statistical difference threshold when performing pairwise comparison between schemes. All figures present standard deviation values for completeness.

6.1 Overall performance

We examine classifiers overall performance considering data coming from CPU, memory, TCP MIB and network statistics individually and collectively, with a window size of 256, overlap of 50 % and a sampling rate of 1ms. In general, as Fig. 4 presents, classifiers can achieve high accuracy almost up to 95 % depending on the classifier and the dataset. Note that similar to any other identification solution, our solution in our approach is prone to false identifications; nevertheless, depending on systems' requirements, there is always a tolerance threshold for such false identification.

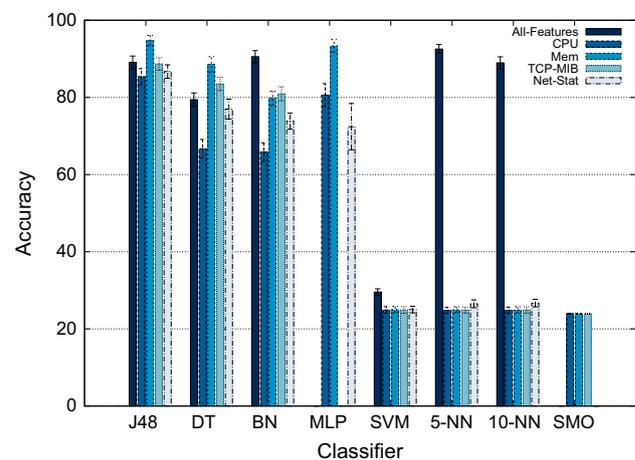


Fig. 4 Accuracy of classifiers using all features collectively

More specifically, we can observe very satisfactory results that reach up to 92.53 % for the 5-NN algorithm when all computational properties are collectively considered. It is interesting to note that when considering computational properties in a standalone manner, k-NN classifiers have a very low accuracy, whereas when all features are combined and used for classification, the accuracy levels climb to values above 90 %. This indicates potential correlations between data referring to diverse computational properties that can act as identifiable patterns for users.

Conversely, when examining classifiers accuracy solely on the different datasets e.g., CPU, memory, TCP MIB and networking statistics, J48 surpasses in terms of accuracy all the others. Particularly, it achieves accuracy of 85.40, 94.74, 88.68 and 86.70 % for the CPU, memory, TCP MIB and network statistics datasets, respectively. Moreover, SVM and SMO that are two computationally intensive algorithms in terms of training exhibit very low accuracy due to their binary classification approach.

However, the evaluation of a classifier should not only be based on the accuracy of the classification process, but it should also include a series of quantitative metrics that expose different aspects of the behavior of the classifier, as well as a qualitative analysis such as the use of confusion matrices that indicate the actual misclassifications. In this respect, in Table 1 we present an indicative example of the statistical properties regarding the quantitative evaluation of the classifiers when applied on the memory dataset. In accordance with the results, it is important to note that despite the fact that SVM, SMO and k-NN have similar accuracy levels, the root-mean-squared error of SVM is higher compared to SMO and k-NN classifiers (root-mean-squared error of 0.6124 for SVM compared to 0.4748 for SMO and 0.4519 for 10-NN), thus highlighting that SVM is less accurate.

Furthermore, in terms of the F -measure that combines both precision and recall and thus also expresses the accu-

racy of the classifier, all classifiers with the exception of SMO exhibit the same behavior as with the traditional accuracy metric. Specifically, SMO has a recall value of 1, indicating that the classifier was fully successful in correctly classifying instances (it does not assign instances to classes at all otherwise), and this fact induced a much higher than expected value in the F -measure metric. Interestingly, the area under ROC metric that has been used extensively to assess the quality of the classification process is equally high for J48, DT, MLP and BN—the latter having exhibited much lower accuracy levels than the other ones. Since this metric is associated with true and false positives, we can assert that the BN classifier has a very low rate of false-positive classifications and that its comparatively lower accuracy is based on true-negative classifications.

We experimented further with different statistical measures to try to expose the behavior of the classification algorithms and understand their operation. Accordingly, the Kappa statistic is used to indicate the agreement of prediction compared to the ground truth, and aside to the latter, it also takes into account the probability that a correct classification was made by chance. The results follow the same trend as the other metrics, although it is interesting to note the much lower than expected performance of the DT and BN classifiers, highlighting increased chances of random correct classifications. Lastly, we calculated the KB mean information value [47], which excludes prior class probabilities. It thus assesses with higher accuracy the performance of classifiers under uncertain conditions. This metric too follows the same pattern as the other ones, with J48 having the highest value of 1.8578.

Training time is also an important factor for the employment of specific classifier for user identification in real-time services. In this respect, classifiers that demonstrate high training times deter us from considering them as viable options in practical scenarios such as user re-authentication.

Table 1 Comparative evaluation of classifiers' performance for memory data

Classifier evaluation metric	J48	DT	BN	SVM	SMO	5-NN	10-NN	MLP
Accuracy	94.7489	88.5537	79.7578	24.9971	23.9304	24.9573	24.8887	93.3733
Incorrect classifications (%)	5.2511	11.4463	20.2422	75.0029	76.0696	75.0427	75.1113	6.6267
Root-mean-squared error	0.155	0.2125	0.3126	0.6124	0.4748	0.4748	0.4519	0.1562
IR precision	0.9531	0.9071	0.7762	0.0574	0.2393	0.0503	0.0718	0.9419
IR recall	0.9368	0.8502	0.7032	0.24	1	0.21	0.3	0.8801
F -measure	0.9444	0.8764	0.7367	0.0927	0.3862	0.0811	0.1159	0.9089
Area under ROC	0.9776	0.9795	0.9468	0.5	0.5	0.5	0.5	0.9747
KB mean information	1.8578	1.5168	1.5082	0.1881	0.2072	0.1736	0.1442	1.8208
Kappa statistic	0.9299	0.8472	0.7297	0	0	0	0	0.9116
Elapsed time training (s)	0.3327	1.1742	0.1867	0.4203	0.0158	0.0003	0.0003	2498.8935
Elapsed time testing (s)	0.0003	0.0009	0.0053	0.0402	0.0003	0.7313	0.7312	0.0105

This is, for instance, the case of the MLP classifier, the training period of it approximates 40 min, though its accuracy is quite high, namely at 93.37 %. The remaining classifiers do not suffer from long training periods; therefore, their applicability in the envisaged scenarios is unhindered.

Overall, the J48 classifier emerges as an excellent candidate for user identification since it combines computational efficiency (low training and testing times) with high accuracy and small classification errors. This is further backed up by the statistical properties presented in Table 1 regarding the comparative evaluation of the classifiers.

6.2 Windows size

The size of the windows that are used in the classification process is considered a parameter of high importance, since it influences classifiers' performance. This is because windows essentially group together data to expose common group characteristics. Evidently, the smaller the size of the window,

the larger the number of instances that will be available to the classifier to operate on, whereas inversely a large window size would reduce the number of available data instances.

We experimented with window sizes, namely 64, 128, 256, 512 and 1024, retaining the same amount of overlap, i.e., 50 % between instances. The parameter settings for the considered classifiers are the same as in the original experiments, and the sampling rate has been set to 1 ms. Windows size influence on the classifiers' accuracy, when applied on different types of data, is illustrated in Fig. 5. Overall, J48 surpasses in terms of performance all the other classifiers under all the configurations. The best (97.51 %) and the worst (85.58 %) identification accuracies were achieved when performing on memory and network statistics data, respectively.

Specifically, Fig. 5a presents the influence of the windows size on the accuracy of the classifiers on CPU data. SVM, SMO and the k-NN classifiers seem unaffected by the different sizes of windows with low performance accuracy, around 25 %.

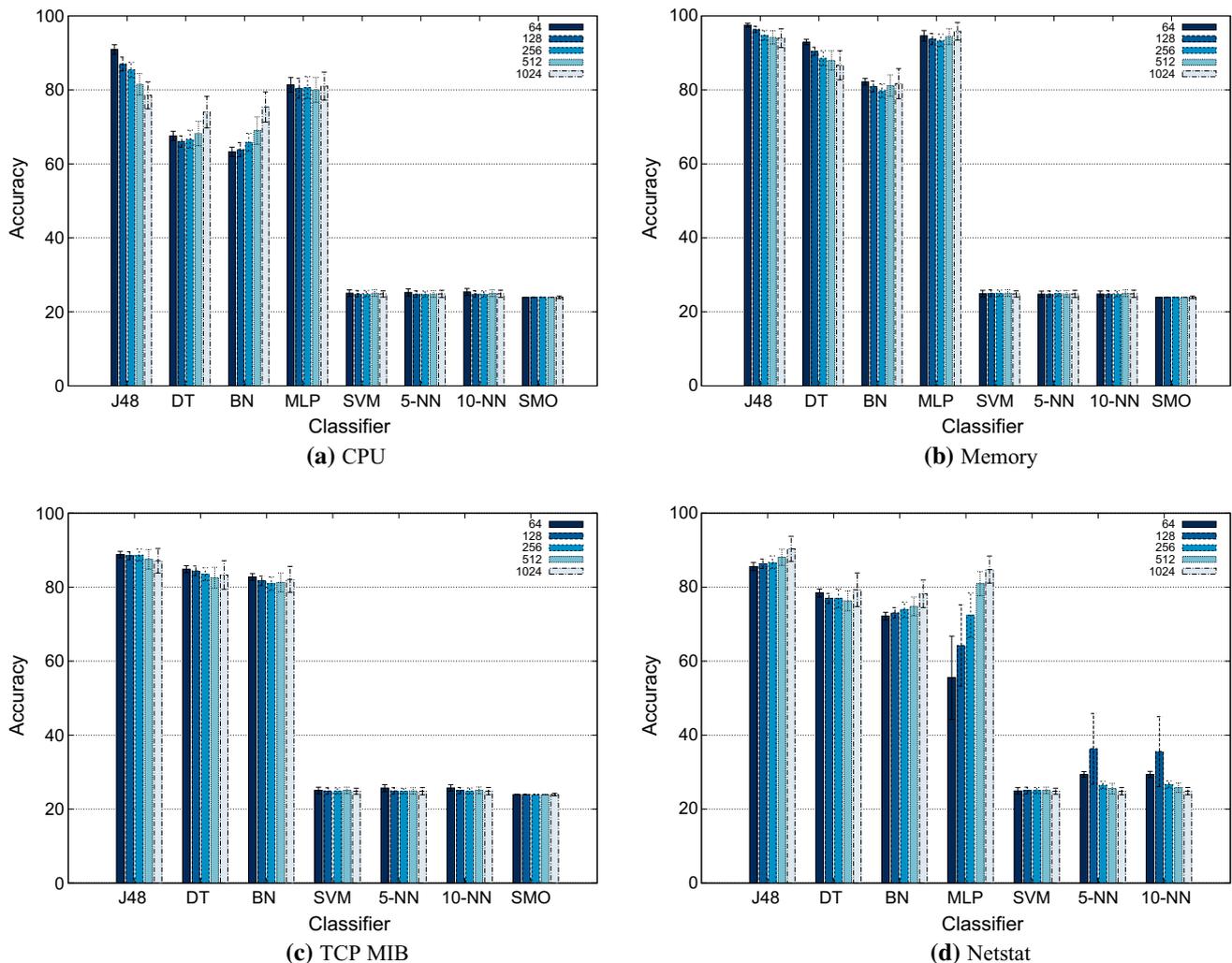


Fig. 5 Effect of size of windows on the accuracy of classifiers. **a** CPU. **b** Memory. **c** TCP MIB. **d** Netstat

Similarly, MLP exhibits minor performance variations subject to the different size of windows, but achieves accuracy of up to 81 % in the case of window of size 64. Conversely, the J48 algorithm that exhibits the best performance overall is highly influenced by the size of windows. For instance, when the window size is set to 64, the accuracy of the classifier reaches 90.92 %, while it drops to 78.53 % in the case of window size of 1024. This performance degradation is attributed to the use of binary splits when building the trees combined with loss of distinguishable patterns in the training dataset when a higher window size is considered. The granularity of the data complicates the pinpointing of patterns when a lot of entries are considered, whereas smaller window sizes make it easier to find distinct patterns. DT and BN accuracy performance is less than 70 %. However, their accuracy is increased as the windows size expanded. This is because these classifiers are based on probabilistic metrics and not on binary ones, and so classification accuracy increases with the window size as the number of available instances decreases.

As far as the memory is concerned, Fig. 5b indicates a slightly different behavior of the classifiers. Consistent to what was observed for CPU data, classifiers SVM, SMO and the k-NN ones perform on the basis of exactly the same behavior. With reference to J48, DT and BN classifiers, their respective accuracy drops when the window size increases. While for J48 this is consistent with the results concerning CPU, DT and BN do not exhibit the same behavior. The reason for this observation lies in the nature of the memory data that are changing less frequently than the ones related to CPU or network. This led to an overall smaller amount of data being collected for memory and hence the accuracy of the classifiers decreases with the window size in this case, since there are fewer distinct patterns that become more dispersed with larger windows.

When using data from the TCP MIB to test the classifiers' performance, the results regarding the effect of the window size illustrate a minimal influence as illustrated in Fig. 5c. More specifically, SVM, SMO and k-NN classifiers have no discernible change in their accuracy because of the modifications of the window size, while they perform on the same levels as before. However, J48, DT and BN have a slight decrease when the window size increases, achieving accuracy of up to 88 %. The justification for this lies as before on the loss of distinct patterns over the data as the size of windows increases.

Evidently, the particular TCP MIB data have fewer such patterns compared to the CPU ones and hence the smaller decrease in classifiers' accuracy. However, it is noteworthy that the effect of the window size on the F -measure metric is more significant, whereby for the three classifiers, namely J48, DT and BN, it starts at 0.95, 0.91 and 0.88, respectively, for a window size of 64, to drop to 0.89, 0.88 and 0.78,

respectively, for a window size of 1024. Since the F -measure reflects the quality of the classification process (it combines both precision and recall), we argue that such an observation should be taken into consideration when considering the optimal window size for TCP MIB data and according to favor small windows.

Lastly, we analyzed the effect of the window size on the classification of the data reflecting network statistics through the netstat service. As it can be seen in Fig. 5d, the results follow a different line compared to the ones regarding CPU, memory and TCP MIB, an observation that can be attributed to the particularities of this type of data: It refers mostly to nominal integer values. Specifically, while once again the SVM classifier remains unaffected by the changes in window size, the 5-NN and 10-NN classifiers outperform SVM and they exhibit significant improvements in accuracy when windows of size 64 and 128 are considered; accuracy reaches 36.30 and 35.55 % for a window size of 128 for 5-NN and 10-NN, respectively, although performance remains very poor. Conversely, for the other classifiers, namely J48, DT, BN and MLP, accuracy increases with the window size, albeit at different rates. Whereas for J48 accuracy increases from 85.58 to 90.34 % for windows of 64 and 1024, respectively, for MLP these values are accordingly 55.50 and 84.77 %. The latter leads us to consider using the network statistics data for user identification with high values in the window size, since on one hand the accuracy levels are quite high (almost 85 %), and on the other hand, this would lead to significantly less processing due to the reduced number of data instances.

It can be broadly seen that for the considered features and datasets, the J48 classifier (i.e., C4.5 decision tree) yields much better classification results in comparison with the other algorithms tested and in particular the SVM one. The SVM classifier is inherently a binary linear classifier, and for this reason, we have experimented with its multi-class variants in order to reduce our multi-class problem into a series of multiple binary classification problems [48,49].

Moreover, the linear nature of the SVM classifier implies the existence of a linear relationship among the features of the dataset, which clearly does not exist in this case as evidenced by the results of our experiments. Conversely, J48 classifier is an extension of decision trees and is thus by design aimed at tackling multi-class problems such as the one studied in this problem. J48 considers the notion of information entropy to classify new instances to a particular class, in contrast to the establishment of a linear relationship used by SVM. The use of information entropy is a much more descriptive method to distinguish between features of the dataset and allows for more general classifications, i.e., not only those characterized by linearity. It becomes therefore evident why under the considered conditions the J48 classifier greatly outperforms the SVM one.

6.3 Sampling rate

Sampling rate is considered to be an important factor when monitoring users' activities in the direction of distinguishing users. It affects the amount of collected data and hence the processing overhead imposed on the classifiers, as well as the time elapsed to accurately distinguish users. In this respect, we experimented with different sampling rates of 1, 10, 50, 100 and 500 ms for information recording retaining the same window size of 256 and an amount of overlap of 50 % between instances. The parameter settings for the considered classifiers are the same as in the original experiments. The influence of sampling rates on the classifiers' accuracy for different types of data is illustrated in Fig. 6. Similarly to the outcomes referring to the window size, J48 performs with higher accuracy under all the configurations, ranging between 84 and 95 %.

Specifically, Fig. 6a overviews classifiers' performance in terms of accuracy when applied on CPU data. SVM, SMO,

and k-NN classifiers are slightly influenced by the sampling rate, while they exhibit accuracies in the range of 25 %. DT and MLP are characterized by a slight change of approximately 2 % between different sampling rates. However, J48 is highly impacted from the sampling rate. For instance, it reaches 85 % at a sampling rate of 1 ms, a value that decreases to 69 % when the sampling rate is tuned to 100 ms. The reason for this behavior lies in the fact that CPU monitored data are extremely time sensitive and therefore patterns from such data are difficult to be observed under high sampling rates. Moreover, the amount of available data instances is relatively smaller for higher sampling rates, which can lead to fewer data being used for training and therefore lower accuracies under high sampling rates. Nevertheless, a completely different behavior is exhibited by the BN classifier, which is inherently probabilistic and thus performs better with smaller data populations, which favor random class assignments.

The classifiers exhibit a very similar behavior to the one corresponding to the CPU data when applied on the memory

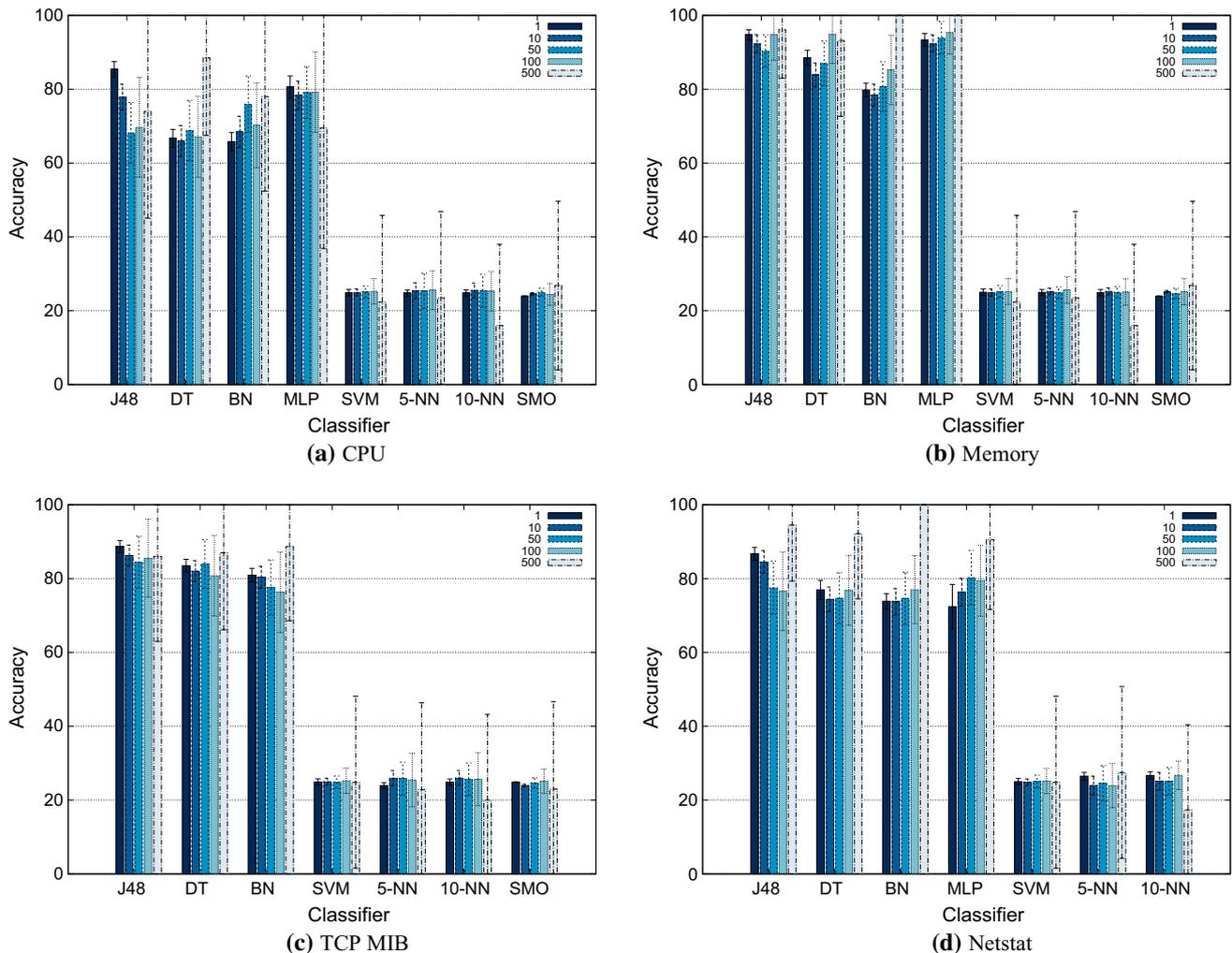


Fig. 6 Effect of data sampling rate on the accuracy of classifiers. **a** CPU. **b** Memory. **c** TCP MIB. **d** Netstat

dataset, albeit at a higher accuracy as illustrated in Fig. 6b. On the one side, previously as SVM, SMO and k-NN classifiers do not incur any impact from different sampling rates, they still achieve poor accuracy as low as 25 %. On the other side, J48, DT, BN, and MLP are influenced by the different sampling rates, i.e., DT accuracy is decreased by 5 % between sampling rates of 1–10 ms, while the remaining classifiers are facing a decrease of 1–2 % accordingly. However, in general J48 and MLP enjoy accuracy levels of around 93 %.

With regard to classifiers' accuracy on TCP MIB data, as Fig. 6c depicts, SVM, SMO, and k-NN exhibit the same behavior as before. J48, DT and BN achieve less accuracy as less information is collected with increased sampling rates. In particular, J48 and BN accuracy is decreased by almost 4 % when the sampling rate is set from 1 to 100 ms, while for DT the respective decrease is at 3 %. This decrease on accuracy is expected, because when the sampling rate is increased, despite the fact that users' interactions with networking services remain identical, smaller variations on network data are nonetheless recorded. However, the results show that relying solely on TCP MIB data, the classifiers' accuracy can still be around a promising 85 % for J48, DT as well as BN.

Furthermore, the performance of the classifiers' accuracy when considering the dataset regarding network statistics is reflected in Fig. 6. In this case, the accuracy of J48 decreases almost by 10 % when examining the results for sampling rates between 1 and 100 ms, while all the other classifiers perform similarly to the output of the experimental results that were exhibited in the case of TCP MIB data. A distinct exception is that of MLP, the accuracy of which is increased when the sampling rate is decreased. Data originating from the netstat service is rather time sensitive, and there exists a strong time dependence, e.g., when a user is connecting to a Web site, there is a distinctive pattern of communication comprising the establishment of the connection, its negotiation, the data transfer and the finalization of the connection. Clearly, when considering a high sampling rate, such patterns cannot be observed and therefore the accuracy of the classifiers is accordingly reduced. As far as the MLP is concerned, the results highlight the increased efficiency of neural network algorithms to compensate such adverse characteristics and therefore ensue encouraging accuracy results.

Overall SVM, SMO, 5-NN and 10-NN perform very poorly, in the range of 25 %, for all the examined datasets and sampling rates. The remaining classifiers, namely J48, DT, BN and MLP, perform between 60 and 95 % in terms of accuracy depending on the dataset and the sampling rate. However, it is significant to underline that as the sampling rate is decreased, classifiers' accuracy drops slightly. For instance, when the sampling rate is set to 1 ms, the accuracy of the DT classifier on the memory dataset is 88 %, while when the rate is set to 10 ms, the accuracy is decreased by as little as 5 %. J48 outperforms the other classifiers, under all

the examined datasets. J48, as the other classifiers, achieves better results when it is applied on the memory feature. Its accuracy in the latter case is more than 90 % even when the sampling rate is decreased.

Another important point to highlight involves the high sampling rates and the fact that much fewer data will be monitored under such conditions. This observation imposes constraints on the evaluation of the classification process, especially for values as high as 500 ms. The amount of the monitored data in this case is not considered adequate to perform a thorough statistical analysis, and therefore, the corresponding results presented in Fig. 6d should be taken with a grain of salt. It is for example clear that probabilistic algorithms such as BN would in this case exhibit unrealistic accuracy levels of up to 100 % since there are too few instances to classify. We plan to address this shortcoming in our future work by expanding the time of the user sessions in order to account for additional rich datasets.

6.4 Users

Another interesting parameter to study involves the number of distinct users that the classifiers need to consider for classification i.e., to distinguish one from another and its influence on the classifiers' accuracy. Since it is highly likely that users might exhibit similar characteristics when using a computational resource, we might expect to encounter difficulties in distinguishing them with high accuracy. Therefore, the number of users is important to study the scalability and potential of the classifiers to be applied to realistic scenarios that include large populations of users.

We examine the accuracy of the classifiers as the number of users is increased from two to four, while the window size is fixed to 256, the overlap is 50 % and the sampling rate is 1 ms. Figure 7 reports on classifiers performance on user identification, using the different monitored features i.e., CPU, memory and network consumption, as the number of users is increased. As expected, classifiers' accuracy is degraded when additional users are introduced in the system. On the one hand, under all the configurations, the SVM, SMO, 5-NN and 10-NN classifiers perform very poorly. On the other hand, the J48, DT, BN and MLP classifiers achieve accuracy levels of up to 98 % depending on the configuration; however, the general trend regarding the effect of the number of users is the same for all the configurations. Nevertheless, we need to underline that the decline in performance, subject to the number of users, is not detrimental and it does not occur at a staggering rate, thus providing encouraging indications regarding large-scale deployment of such classifiers for user identification.

In particular, Fig. 7a presents classifiers performance on CPU data as the number of users is increased. SVM, SMO and k-NN exhibit the same levels of accuracy at approximately

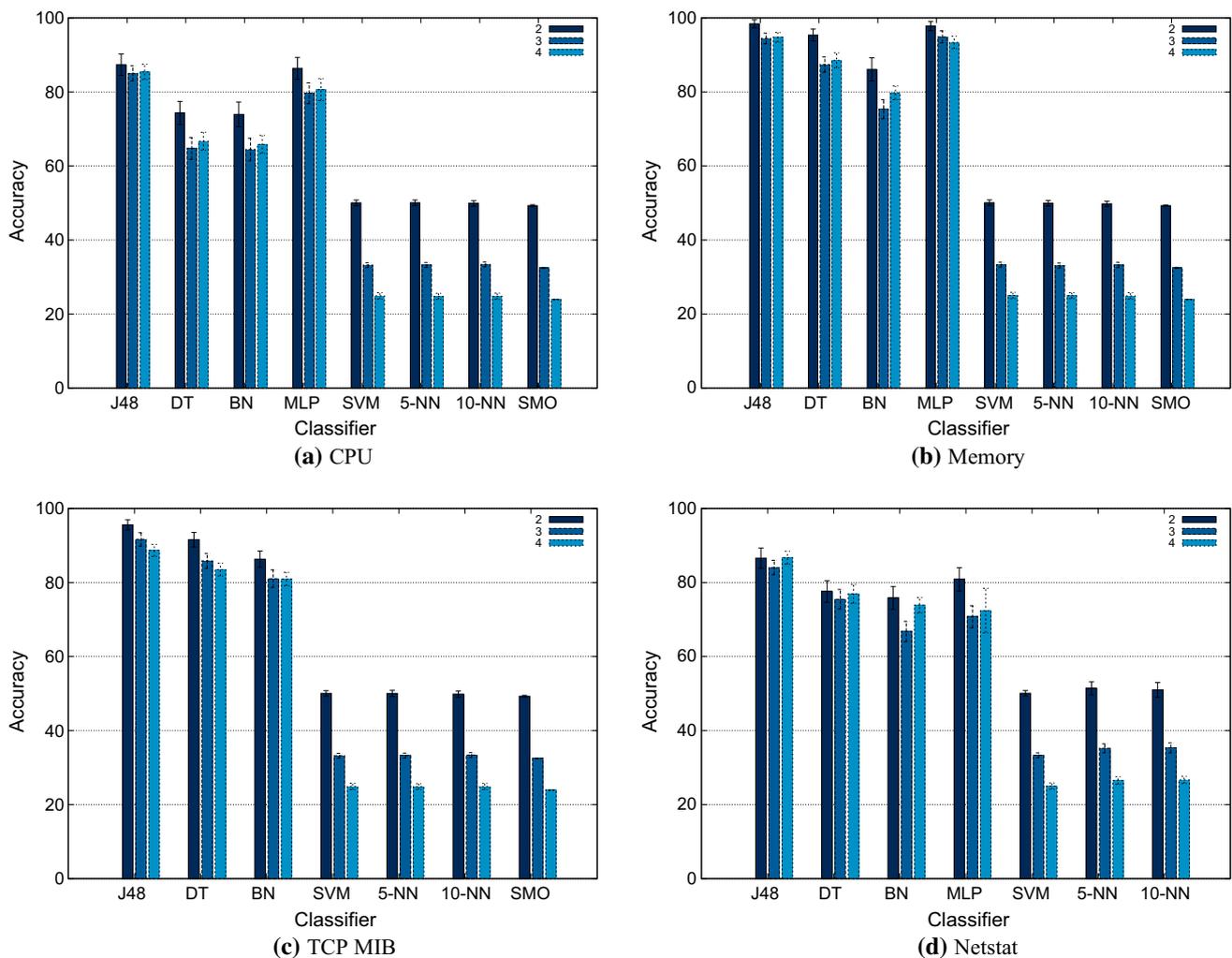


Fig. 7 Effect of number of users on the accuracy of classifiers. **a** CPU. **b** Memory. **c** TCP MIB. **d** Netstat

50, 33 and 24 % for two, three and four users, respectively. DT and BN perform better with accuracy in the range of 66–74 % without imposing high accuracy variations compared to the previous classifiers. MLP and J48 achieve almost the same level of accuracy (87 %) when they are applied to distinguish among two users. MLP accuracy decreases up to 6 % for three and four users, while J48 accuracy is degraded only by 2 %, both of its results are very satisfactory in terms of scalability.

With regard to classifiers performance when applied on the memory dataset, as the number of users is increased they follow a similar trend to that of the CPU dataset, albeit exhibiting an overall higher accuracy, as Fig. 7b illustrates. SVM, SMO, and k-NN perform poorly with accuracy levels as low as 23 % for the configuration of four users. Conversely, MLP and J48 perform almost identically in terms of accuracy for the configuration of two users, approximately 98 %. However, J48 outperforms MLP by about 1 % for the case of four users, while DT and BN achieve smaller levels of accuracy between 80 and 85 % for all cases. Similarly to the other experiments,

the results concerning the memory dataset are the ones that exhibit the highest accuracy out of all considered ones.

As far as TCP MIB information is concerned, classifiers perform with less accuracy compared to the memory dataset, as Fig. 7c presents. SVM, SMO and k-NN performance is highly degraded as the number of users is increased, thus clearly sidelining these algorithms from practical deployment. Their accuracy decreases by almost 25 % when increasing from two to four users. BN and DT contrariwise achieve better accuracy and they incur the same level of decrease by approximately 7 % when considering two and four users. Lastly, as in most experimental configurations, the J48 classifier performs most efficiently and manifests the best accuracy for this setup as well.

Network statistics classifiers manifest a performance in terms of accuracy that is quite similar to that observed for the CPU dataset as Fig. 7d illustrates. J48 has once again the highest accuracy rates, and the interesting thing to note is the fact that accuracy is not influenced by the number of

users (86.58 % for two users and 86.7 % for four users). Similar scalable behavior can be noted for the DT and BN algorithms, whereas MLP has a slight decrease of around 8 % accordingly. However, in the latter case, there is almost 6 % of standard deviation. It is noteworthy that standard deviation increases with the number of users only for MLP, which is attributed to the inherent design of the algorithm: The complexity of the underlying neural network is increased and a greater number of internal weight functions are considered, hence increasing uncertainty. This is also consistent with the decrease in accuracy.

6.5 Learning curve

Another aspect that we examined in our experimental validation of the proposed user identification framework refers to the learning curve of the classifiers. Examining the latter can yield significant information regarding the time it takes for a classifier to start correctly identifying users, whereas in parallel it serves as an indication on the classifier's performance. As in Sect. 6.1, results for the SMO and MLP classifiers are not discussed here since these classifiers cannot be applied to the combined features dataset.

In our configuration, the amount of data collected for each user increases at an extremely high rate, as we collect one instance per 1 ms for the overall period of 3 h per user, so one would expect that the performance of the classification process would drop with the progression of time, due to the need to constantly train the classifier with the new data and the increased difficulty in finding distinguishing patterns over large datasets compared to smaller ones. For this reason, we created four data subsets comprising of 25, 50, 75 and 100 % of the original dataset we collected through our experiments. Using these datasets, we performed a series of experiments concerning all features combined (CPU, memory, TCP MIB and, netstat) with the standard configuration settings, as discussed in Sect. 5.4. Such an approach allowed us to analyze the learning curve of the classifiers in a progressive manner, when more and more data are being incorporated to our framework.

Figure 8 illustrates the results with regard to classifiers accuracy for the four considered datasets. We can note that the results are quite encouraging in that they show very small variation, and satisfactory accuracy levels can be reached within the first 45 min of monitoring user behavior through general system properties (25 %). The Bayes network classifier reaches accuracy levels of over 95 %, whereas the 5-NN one reaches 93.0955 %. These high levels of accuracy for the bayes network classifier and the k-NN ones are reduced as more data are being considered. This seems counterintuitive since one would expect that more data would allow for better classifications. This is indeed the case with the J48, DT and SVM classifiers that exhibit a more normalized behav-

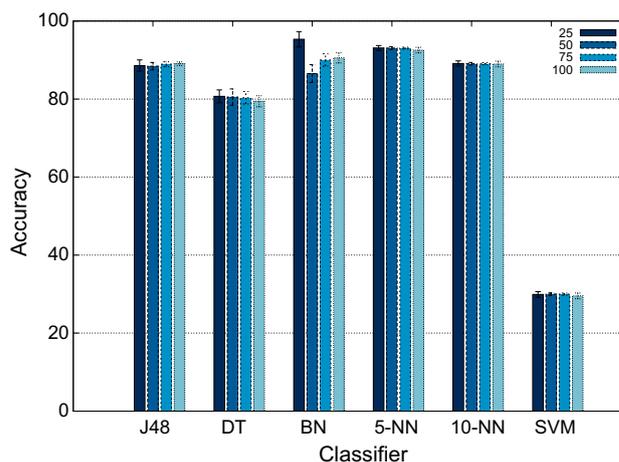


Fig. 8 Data size effect on classifiers accuracy

ior. The explanation lies in the fact that the 25 % dataset and in general the smaller datasets contain fewer number of entries upon which classification and identification are tested, especially considering the 66 % testing to 33 % training split that we perform over the datasets. Therefore, identification accuracy is implicitly boosted since it is a probabilistic process.

We wished to further explore the behavior of the classifiers when considering the learning curve and thus better understand the quality of the received results. Accordingly, we examined the decision trees (for the J48 classifiers) for the various classifications in order to be able to pinpoint the specific features of the dataset that contribute to the user identification process. Indicatively, the J48 decision tree for the user identification process referring to the entire dataset (100 %) can be seen in Fig. 9. It is interesting to observe that the features that mainly contribute to the identification process are the spectral energy distribution of FFT and the mean for the memory. This serves to underline that not only time-domain features, but also frequency-domain features can be considered as good indicators for user identification.

7 Discussion

The presented experimental results have proved the feasibility of our approach in using general system properties for user identification with accuracy of prediction reaching up to 98 % for certain configurations. This fact does not validate only our assertion that such an approach is viable for the direction of a continuous authentication framework, but also shows that even non- directly PII can be used to violate users' anonymity. To this end, orthogonally if the appropriate parameters are taking into account, users anonymity could be preserved as well.

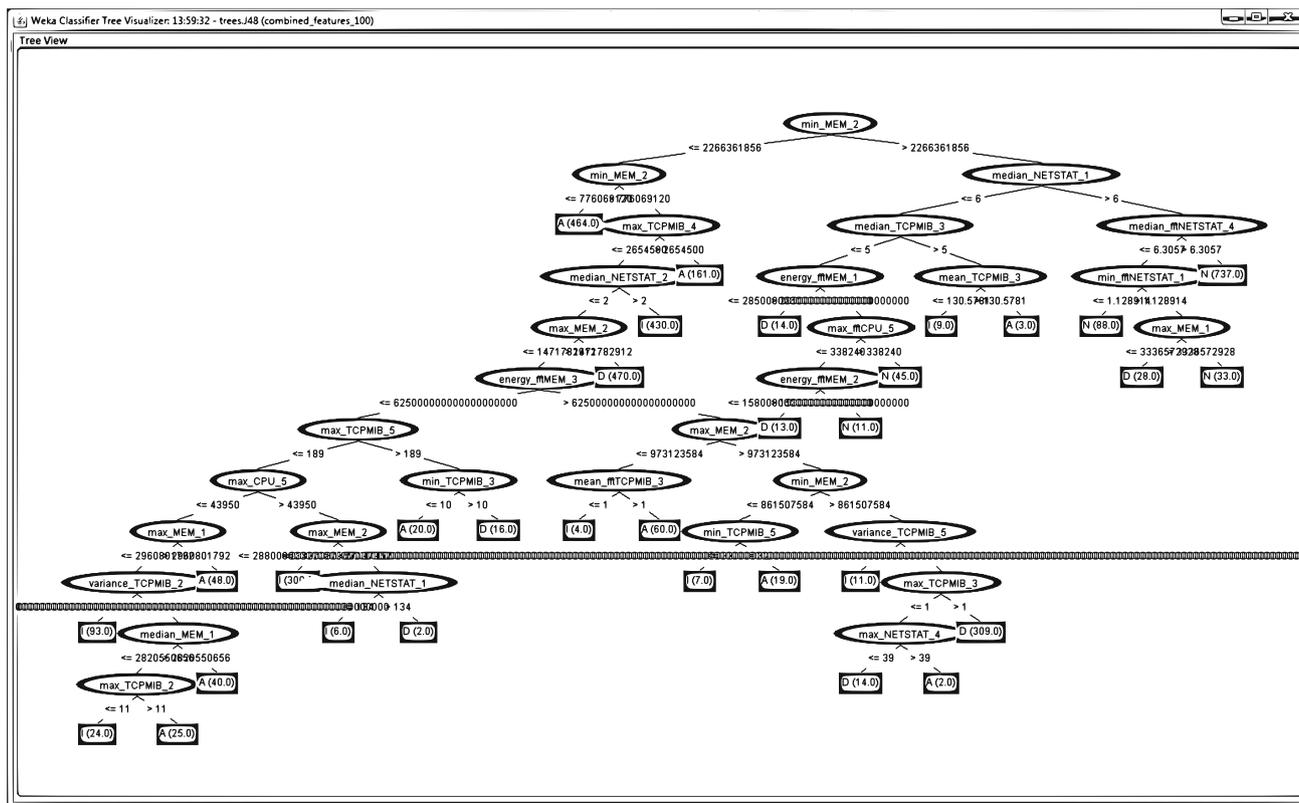


Fig. 9 J48 Decision tree (100 %)

However, similar to any other identification solution, one cannot expect to accomplish an absolute accuracy of user identification. This is more the case when users do not follow specific patterns of behaviors but instead operate their PCs in continuously dynamic manner. In this respect, we are considering to introduce a procedure to update users patterns as their behavior changes during time. In this way, we can eliminate false user identifications toward a continuous authentication solution, even if they operate the system unusually. To be more explicit, as it stands our system has a sole training phase and then based on the training data it can perform classification tasks. Our ongoing work aims to have a continuous training phase; thereby, the system will be able to continuously learn from the behavior of the user and adapt prospective classifications accordingly.

Overall, when all the system properties are considered collectively, it is evident that classifiers such as SVM, SMO, and k-NN cannot be considered to be effective. On the contrary, J48 and MLP emerge as the optimal candidates to be used for user identification. As far as the latter classifiers are concerned, smaller windows sizes (i.e., 64 and 128 instances) exhibited highest accuracy; however, even windows of 256 reach accuracies of up to 94.74 % for J48. Similarly, high-frequency sampling, i.e., 1 ms, results in highest accuracies. However, in this point one might argue that the number of users in our experiments is quite small. Nonetheless, the

promising results concerning four users can serve as a guideline of how the system would work with more users included. Note also that the performance of the system does not significantly degrade subjected to the increase in the number of users and any deterioration does not take place at a high rate. This encouraging observation will be further tested and validated in future experiments with additional number of users.

If we were to consider only data related to CPU utilization, once again J48 and MLP appear to be the most suitable classifiers to use. Optimal performance in that case is incurred configuring the window size to include 64 instances and the sampling rate to be 1 ms reaching accuracy level around 90 % for J48 and 81 % for MLP. J48 seems to be largely unaffected by increasing number of users, whereas the performance of MLP is slightly adversely influenced. Therefore, J48 with the aforementioned configuration should be considered when only CPU data are available.

Memory datasets are the most interesting ones, since they yield the best results in terms of user identification with accuracy values reaching approximately 98 % in the case of J48 and MLP. Whether this is due to particular behavior of the specific applications executed by the users is an aspect that requires further research. Overall, J48 and MLP classifiers should be preferred for such datasets, with sampling rates of 1–10 ms and window sizes of 256. The fact

that classifiers perform very well even with large windows and higher sampling rates significantly reduces processing and data overhead and thus promotes the attractiveness of memory datasets for user identification.

TCP MIB data are satisfactory for user identification when J48, DT and BN classifiers are employed, with accuracies consistently above 80 %. Compared to the other system properties, TCP MIB data perform the worst, but they are broadly unaffected by window sizes and sampling rates; hence, they could be used under certain conditions that necessitate small overhead and average performance. Lastly, network information from the netstat service should be considered as viable for user identification only when applying the J48 classifier. Relevant accuracy results are up to 90 % in this case, whereas they are well below 80 % for the remaining classifiers, including the MLP one that performed exceptionally for the other system properties. In general, large window sizes of 1024 proved to be the most suitable ones to achieve optimal accuracy and this leads to significantly less overhead due to the reduced number of entries for processing. Conversely, continuous and high granularity monitoring is required since the optimal sampling rate was that of 1 ms.

8 Conclusions

Modern computational devices, e.g., PC and mobile phones, are capable of accomplishing a series of activities. In order to support users' tasks, they bear various sensors generating different types of data. Some of these data have inherent unique properties, e.g., GPS location, considering them as personal data, while others are not considered as personal as such since they cannot be directly linked with a specific entity. Under the latter category, we can categorize general system data produced by main computational components such as CPU, memory and network units. Whereas such data have been utilized for system management, to the best of our knowledge there is no previous work that has studied the effectiveness of user identification by means of such data or alternatively whether users could be accordingly monitored.

We reported on the feasibility of user identification relying on general system management data. We introduced a modular, extensible framework for user identification in the direction of building a continuous authentication service on the one side and informing users on their anonymity exposure on the other side. In this respect, this approach can be also used in order to process such data in a privacy-friendly manner if the appropriate parameters are taking into account e.g., type of data and classifier. We experimented with different classifiers, data and number of users to examine whether such an approach can be construed as being beneficial to the end users. Our findings show that even not directly PII, such as CPU, memory and network information, could be used to

identify users with high accuracy subject to the appropriate parameters having been employed.

Our ongoing work involves experimenting with additional statistical features, i.e., pairwise Pearson's correlation and normalized entropy values that could augment classification and assist in reaching even higher levels of accuracy. Additional experiments will take place to further study the effect a large number of users will have on the performance and efficiency of our framework, as well as other conditions that might affect the classification methods. Furthermore, we are considering to introduce a procedure to update users' patterns as their behavior change during time as discussed, in order to fully fulfill our goal to support *continuous user authentication*.

Overall, we believe that this work constitutes a promising advancement in the field of user monitoring considering the two different directions that we are aiming to address: (a) enhancing users' anonymity and (b) introducing a continuous authentication framework to compensate for the shortcomings of current authentication approaches.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Kambourakis, G.: Anonymity and closely related terms in the cyberspace: an analysis by example. *J. Inf. Secur. Appl.* **19**(1), 2–17 (2014)
2. Akkus, I.E., Chen, R., Hardt, M., Francis, P., Gehrke, J.: Non-tracking web analytics. In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, pp. 687–698 (2012)
3. Yang, W., Li, N., Qi, Y., Qardaji, W., McLaughlin, S., McDaniel, P.: Minimizing private data disclosures in the smart grid. In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, pp. 415–427 (2012)
4. Tappenden, A.F., Miller, J.: Cookies: a deployment study and the testing implications. *ACM Trans. Web* **3**(3), 9:1–9:49 (2009)
5. Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., Vigna, G.: Cookieless monster: exploring the ecosystem of web-based device fingerprinting. In: *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, ser. SP '13. Washington, DC, USA: IEEE Computer Society, pp. 541–555 (2013)
6. Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gürses, S., Piessens, F., Preneel, B.: Fpdetective: dusting the web for fingerprinters. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '13. New York, NY, USA: ACM, pp. 1129–1140 (2013)
7. Shi, W., Yang, J., Jiang, Y., Yang, F., Xiong, Y.: Senguard: passive user identification on smartphones using multiple sensors. In: *Proceedings of the 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communica-*

- tions, ser. WIMOB '11. Washington, DC, USA: IEEE Computer Society, pp. 141–148 (2011)
8. Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.-Y.: Understanding mobility based on gps data. In: Proceedings of the 10th International Conference on Ubiquitous Computing, ser. UbiComp '08. New York, NY, USA: ACM, pp. 312–321 (2008)
 9. Weiss, G.M., Lockhart, J.W.: Identifying user traits by mining smart phone accelerometer data. In: Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data, ser. SensorKDD '11. New York, NY, USA: ACM, pp. 61–69 (2011)
 10. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Cell phone-based biometric identification. In: Biometrics: 2010 Fourth IEEE International Conference on Theory Applications and Systems (BTAS). IEEE, pp. 1–7 (2010)
 11. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **10**(5), 571–588 (2002)
 12. Pfitzmann, A., Köhntopp, M.: Anonymity, unobservability, and pseudonymity: a proposal for terminology. In: International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability, pp. 1–9. Springer-Verlag New York Inc, New York, NY, USA (2001)
 13. Islam, M.S.: Heartbeat biometrics for remote authentication using sensor embedded computing devices. *Int. J. Distrib. Sens. Netw.* **2015**, 3:3–3:3 (2015). doi:[10.1155/2015/549134](https://doi.org/10.1155/2015/549134)
 14. Feng, T., Yang, J., Yan, Z., Tapia, E.M., Shi, W.: Tips: Context-aware implicit user identification using touch screen in uncontrolled environments. In: Proceedings of the 15th Workshop on Mobile Computing Systems and Applications, ser. HotMobile '14. New York, NY, USA: ACM, pp. 9:1–9:6 (2014)
 15. Demme, J., Maycock, M., Schmitz, J., Tang, A., Waksman, A., Sethumadhavan, S., Stolfo, S.: On the feasibility of online malware detection with performance counters. In: Proceedings of the 40th Annual International Symposium on Computer Architecture, ser. ISCA '13. New York, NY, USA: ACM, pp. 559–570 (2013)
 16. Tang, A., Sethumadhavan, S., Stolfo, S.: Unsupervised anomaly-based malware detection using hardware features. In: Stavrou, A., Bos, H., Portokalidis, G. (eds.) *Research in Attacks, Intrusions and Defenses*, ser. Lecture Notes in Computer Science, vol. 8688, pp. 109–129. Springer, Berlin (2014)
 17. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. *J. Comput. Secur.* **6**(3), 151–180 (1998)
 18. Yang, Y.C.: Web user behavioral profiling for user identification. *Decis. Support Syst.* **49**(3), 261–271 (2010)
 19. Herrmann, D., Gerber, C., Banse, C., Federrath, H.: Analyzing characteristic host access patterns for re-identification of web user sessions. In: Proceedings of the 15th Nordic Conference on Information Security Technology for Applications, ser. NordSec'10. Berlin, Heidelberg: Springer, pp. 136–154 (2012)
 20. Xu, C., Du, C., Zhao, G.F., Yu, S.: A novel model for user clicks identification based on hidden semi-Markov. *J. Netw. Comput. Appl.* **36**(2), 791–798 (2013)
 21. Melnikov, N., Schönwälder, J.: Cybermetrics: user identification through network flow analysis. In: *Proceedings of the Mechanisms for Autonomous Management of Networks and Services, and 4th International Conference on Autonomous Infrastructure, Management and Security*, ser. AIMS'10. Berlin: Springer, pp. 167–170 (2010)
 22. Kapusta, J., Munk, M., Svec, P., Pilkova, A.: Determining the time window threshold to identify user sessions of stakeholders of a commercial bank portal. *Proc. Comput. Sci.* **29**(0), 1779–1790 (2014). (2014 International Conference on Computational Science)
 23. Berendt, B., Mobasher, B., Spiliopoulou, M., Wiltshire, J.: Measuring the accuracy of sessionizers for web usage analysis. In: *Int. SIAM Workshop on Web Mining (Apr 2001)*
 24. Yang, Y.C., Padmanabhan, B.: Toward user patterns for online security: observation time and online user identification. *Decis. Support Syst.* **48**(4), 548–558 (2010)
 25. Jain, A.K., Hong, L., Pankanti, S.: Characterizing applications workloads on cpu utilization for utility computing. Hewlett Packard, Tech. Rep. HPL-2004-157
 26. Gong, Z., Gu, X.: Pac: Pattern-driven application consolidation for efficient cloud computing. In: Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, ser. MASCOTS '10. Washington, DC, USA: IEEE Computer Society, pp. 24–33 (2010)
 27. Singleton, R.C.: On computing the fast Fourier transform. *Commun. ACM* **10**(10), 647–654 (1967)
 28. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.* **11**(5), 561–580 (2007)
 29. Zhang, J., Figueiredo, R.J.: Application classification through monitoring and learning of resource consumption patterns. In: Proceedings of the 20th International Conference on Parallel and Distributed Processing, ser. IPDPS'06. Washington, DC, USA: IEEE Computer Society, pp. 144–144 (2006)
 30. Brown, P.F., deSouza, P.V., Mercer, R.L., Pietra, V.J.D., Lai, J.C.: Class-based n-gram models of natural language. *Comput. Linguist.* **18**(4), 467–479 (1992)
 31. Blaica, B., Vladušić, D., Mladenčić, D.: MTi: a method for user identification for multitouch displays. *Int. J. Hum. Comput. Stud.* **71**(6), 691–702 (2013)
 32. Zahid, S., Shahzad, M., Khayam, S.A., Farooq, M.: Keystroke-based user identification on smart phones. In: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, ser. RAID '09. Berlin: Springer, pp. 224–243 (2009)
 33. Pan, S., Chen, A., Zhang, P.: Securitas: User identification through rgb-nir camera pair on mobile devices. In: Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, ser. SPSM '13. New York, NY, USA: ACM, pp. 99–104 (2013)
 34. Mock, P., Edelmann, J., Schilling, A., Rosenstiel, W.: User identification using raw sensor data from typing on interactive displays. In: Proceedings of the 19th International Conference on Intelligent User Interfaces, ser. IUI '14. New York, NY, USA: ACM, pp. 67–72 (2014)
 35. Kotsiantis, S.B.: Supervised machine learning: A review of classification techniques. In: Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies Amsterdam, The Netherlands, The Netherlands: IOS Press, pp. 3–24 (2007). <http://dl.acm.org/citation.cfm?id=1566770.1566773>
 36. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition* (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann Publishers Inc., San Francisco (2005)
 37. Quinlan, J.R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc, San Francisco (1993)
 38. Kohavi, R.: The power of decision tables. In: Proceedings of the 8th European Conference on Machine Learning, ser. ECML '95. Springer, London, UK, pp. 174–189 (1995)
 39. Heckerman, D., Geiger, D., Chickering, D.M.: Learning bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.* **20**(3), 197–243 (1995)
 40. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers Inc., San Francisco (2000)
 41. Platt, J.C.: Using analytic qp and sparseness to speed training of support vector machines. In: Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II. MIT Press, Cambridge, MA, USA, pp. 557–563 (1999)

42. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. *Mach. Learn.* **6**(1), 37–66 (1991)
43. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. *Neural Netw.* **2**(5), 359–366 (1989)
44. Kaufman, S., Rosset, S., Perlich, C.: Leakage in data mining: Formulation, detection, and avoidance. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. New York, NY, USA: ACM, 2011, pp. 556–563. doi:[10.1145/2020408.2020496](https://doi.org/10.1145/2020408.2020496)
45. Perlich, C., Melville, P., Liu, Y., Świrszcz, G., Lawrence, R., Rosset, S.: Breast cancer identification: Kdd cup winner's report. *SIGKDD Explor. Newsl.* **10**(2), 39–42 (2008). doi:[10.1145/1540276.1540289](https://doi.org/10.1145/1540276.1540289)
46. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. *SIGKDD Explor. Newsl.* **11**(1), 10–18 (2009)
47. Kononenko, I., Bratko, I.: Information-based evaluation criterion for classifier's performance. *Mach. Learn.* **6**(1), 67–80 (1991)
48. Hsu, C.-W., Lin, C.-J.: A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Netw.* **13**(2), 415–425 (2002)
49. Duan, K.-B., Keerthi, S.S.: Which is the best multiclass svm method? An empirical study. In: *Proceedings of the 6th International Conference on Multiple Classifier Systems*, ser. MCS'05. Springer, Berlin (2005), pp. 278–285. doi:[10.1007/11494683_28](https://doi.org/10.1007/11494683_28)