J R C T E C H N I C A L R E P O R T S

Utilizing CPU, Memory and other features signals to control processes and related data in computing devices with potential to identify user

An Application Risk Assessment Approach

Dimitris GENEIATAKIS

Apostolos MALATRAS

Ioannis VAKALIS

2015

Report EUR 27054 EN

Joint Research Centre

European Commission

Joint Research Centre

Institute for the Protection and Security of the Citizen

Contact information

Dimitrios Geneiatakis

Address: Joint Research Centre, Via Enrico Fermi 2749, TP 361, 21027 Ispra (VA), Italy

E-mail: dimitrios.geneiatakis@jrc.ec.europa.eu

Tel.: +39 0332 783973

https://ec.europa.eu/jrc

JRC93611

EUR 27054 EN

ISBN 978-92-79-45027-3 (pdf)
ISBN 978-92-79-45025-9 (print)
ISBN 978-92-79-45026-6 (CD-ROM)

ISSN 1831-9424 (online)
ISSN 1018-5593 (print)
ISSN 1831-9424 (CD-ROM)

doi:10.2788/259848

Luxembourg: Publications Office of the European Union, 2015

Table of Contents

1 Introduction

Much attention has been dedicated to privacy and anonymity of end users' in Internet era. Various research works [39] [40] [41] determine the basic threats against users' privacy and anonymity when accessing electronic services. In Internet era users' rely more and more on electronic transactions and as a result generate high frequent data that might reveal their activities. Not only personal identifiable information (PII) such as IP and email addresses, cookies [42] , etc., can be used to monitor and identify a user, but also information produced by systems' features and sensors e.g., browser fonts, GPS, accelerometer, and other, as reported in [43] [44] [45] .

Existing monitoring and identification techniques rely mainly on PII or on sensors that generate unique data. For instance, GPS sensors generate unique data that can be easily exploited for user identification [46] . Even, sensors generate less unique "data" such as accelerometer, magnetometer, etc., can be exploited to distinguish users [45] [47] [48] . However, existing approaches do not distinguish users which share computational resources e.g., PC, phone, etc. Furthermore, it should be mentioned that such techniques might not be suitable for user monitoring on devices lack specific sensors.

In that direction, we believe that general system based features such as memory, CPU, network stats, etc., can be used to distinguish and identify users sharing computational resources. Our goal is to deliver tools indicating to the user that during his activity on a computing device, his utilisation patterns can lead to his identification, profiling and consequently expose personal data. We should mentioned that on the one side such approaches can be used to employ a continuous authentication assessment, in which when an uncommon behaviour is detected would trigger users' authentication. While on the other side, the monitor of such features can drive to privacy and anonymity violations if enough data are available for analysis. For instance, in [49] the authors present a feasibility study of user identification by monitoring electrical appliances consumption.

Thus, in this work we report on the efficiency of user identification relying on non PII system features. To do so, we employed a real time monitor, based on SIGARAPI [50] , to collect memory, CPU and network stats, while we conduct a preliminary experiment of monitoring four users in sessions of time-window of one and three hours correspondingly. We study the effectiveness of well-known machine learning algorithms to identify and distinguish the monitored users.

2 Related Work

The core thinking of this work is that information, directly available or through collection techniques in computing devices, can be used to identify or profile a user. This can be done mainly monitoring the use computer resources like CPU, memory and networking or even by recording behaviour patterns and habits while the user is online. Though, we could identify various research areas that are related to our current work such as malware detection through monitoring systems features [51] [63] [64] [65] , we overview only the approaches focusing on user identification on the direction of developing a set of signatures or signals to identify a user.

2.1 Computer Resource Monitoring

CPU, memory and other performance counters are basic mechanisms used to monitor the current state of a computing device. Monitoring has also the advantage that can be done in real time, dynamically, and this way it allows actions to be taken as a reaction to accomplish various tasks. In our description for the online tools we have foreseen monitoring activity with the objective to investigate to what extend the collected information can be utilised for identification of the user and his activities. The activity level monitoring is crucial since every action or decision of the user is translated to function calls, processes, memory use, which in turn include all the idiosyncratic features of the user, which we target to capture.

In the work by Abrahao et. Al. [22] the focus is on CPU utilization. The basic idea is to record the usage traces for a number of applications (12) in order to optimize resource assignment. The experiments were conducted on a shared set of processors providing computing services in grid and utility setup. The basic technique used to characterize the workload for each application, was Principal Component Analysis (PCA). Collected datasets corresponding to twelve applications under examination were characterized with three attributes periodic, noisy, and spiky. The resulting principal components were used to classify the applications, to remove the linear trends of the CPU usage behaviour, and generate synthetic traces with amplification or suppression of the desired features.

Computing resource monitoring for an efficient use and share, particularly in the cloud environment, is an open problem, which is related to our research as well. In this work [21] the signatures of applications and hosts are determined based on signal processing techniques. The basic tool is pattern driven, which defines a signature for each process. The defined signatures then can be used to identify the type of process running and to arrange the resource usage on past based predictions. It was demonstrated that signature based tools reduce resource prediction errors up to 90%. When a larger number of processes are running the performance can be improved an additional 50%. These results were obtained in an implementation based on the Xen virtual machine simulating resource sharing.

In this context, authors in [20] introduces a new approach for application classification based on the Principal Component Analysis (PCA) and the k-Nearest Neighbour (k-NN) classifier in order to design an efficiently scheduling processing algorithm for accessing the various device resources. The developed methodology can be applied in heterogeneous computing setup with the target to improve scheduling and overall performance. The setup is based on the VMPlant a virtual machine model, where each application runs on a scheduled virtual machine hosted on a shared virtual memory. The approach uses classification of applications based on extracted features after the reduction of the dimensions of the performance feature space. The following four dimensions are used: CPU-intensive, I/O and paging-intensive, network-intensive, and idle, by the classification component.

2.2 Web Browsing

Yang et al. [52] build a profile model for user identification based on users' web browsing behavior patterns. Authors, monitor users' activities to extract the strongest indicators that can be used to develop a profile. According to their results the proposed approach performs more efficient compared to decision trees and support vector machine classifiers. Herrmann et al. [53] observe web user activities over a period of time and rely on Multinomial Naive Bayes classifier to identify users. In an alternative approach, Xu et al. [54] monitor users HTTP request in the direction of modeling user mouse clicks based on Hidden Semi-Markov model achieving an accuracy rate of 93%.

Melnikov et al. [55] 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. [56] focus on time value estimation that should be used for user identification approaches that rely on time threshold technique [57] . In that direction, Yang et al. [52] quantify the *amount* of data, in terms of web sessions, required to achieve better results on user identification.

2.3 User Tracking

It was pointed out recently, that even common settings information from browser in combination with other system information, can be used to create a device/user fingerprint [28]. By fingerprint in this context we mean, all the set of common attributes/settings that every device has, which because of the personal user choices they are so specific that identify the device. This fact has severe privacy and security consequences and should be taken into consideration by the average user when he is online. Countermeasures can be implemented particularly for services that are not occasional. Recently a number of works [27] describe tools and frameworks for the detection and analysis of web-based fingerprinters which in the present day internet are quite common. In this paper [27], the design, implementation and deployment of the tool FPDetective is described. This tool, based on font information capture, can detect fingerprinters employed by the web sites. It provides also an analysis

framework so that the user can defend his devices with specific countermeasures. This is an alternative approach to that of counting on known fingerprinters or tracking-sites blacklists. Experiments conducted at large scale using the FPDetective revealed fingerprinting is much higher levels than those estimated. In the same experiments it was shown that common countermeasures are usually not adequate and organisations and users have to update their tactics to protect Personal Identifiable Information.

2.4 Phone Sensors and Touch Screens

Feng et al. [51] 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 behaviour is detected. Authors evaluate their scheme on uncontrolled environments showing that their framework accuracy reaches up to 90%. Shi et al. [45] rely on users gesture patterns using, however, an n-gram~\cite{ngram} model. In this case, an accuracy of 75% is achieved. In another work, Blaica at al. [58] focus on user identification on multitouch displays with the ability to detect five or more touchpoints. Authors assess well-known classifiers and achieve accuracy between 92% and 96%.

Zahid et al. [59] propose a user identification scheme based on keystroke information of mobile devices. In this approach, an accuracy of 98% is achieved. Weiss et al. [47] collect data from mobile phone accelerometer for user identification using a supervised machine learning classifiers. A similar approach is also followed by [48] .

Pan et al. [60] introduce a method to utilize a camera sensor to provide a more secure identification for mobile user. Specifically, the proposed solution extracts and analyzes geometrical features from a human hand to identify a user. In this solution, authors achieve accuracy up to 94%. Mock et al. [61] rely on gray-scale images retrieved from the touch screen. Authors use multi-class support vector machine learning algorithm achieving an accuracy of 97%.

3 Framework for data analysis

Analyzing data to extract useful patterns and accordingly using this data to identify distinct users using a PC or a mobile phone based on their behaviour and usage patterns is a research domain that has attracted high interest over the last years. The goal is to have computing systems capable of inferring who is using them making use of only raw data as input. The overall research in this domain is generally classified as machine learning and there have been a lot of different algorithms in this respect. The main elements of machine learning for the purposes of user identification on PCs and smartphones include the following:

- Data collection: collection of data using sensors located on the users' device.
- Data processing: the collected raw data need to be processed in order to deduce some useful information features and characteristics that will assist in its classification.
- Data classification: use of the aforementioned features in conjunction with machine learning classification algorithms to classify data, i.e. assign classes to data instances.

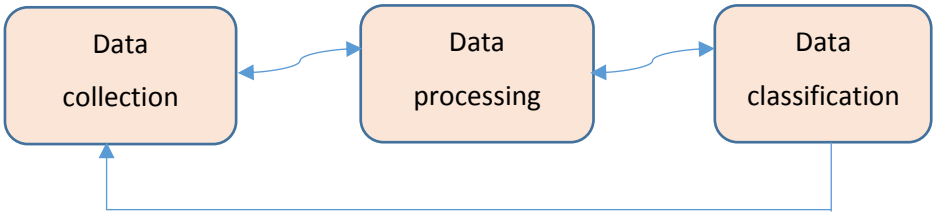

Figure 1. Machine learning stages, e.g. for activity recognition. These 3 steps are repeated for training and testing phases.

Figure 1 depicts these three different steps in the machine learning process, it needs nonetheless to be mentioned that this is a standard textbook process (see for example [16] [17] ,[12] ,[13]) and not a contribution of our work. We applied all three different stages on the collected training data from our experiments and appropriately configured the classification process to enhance its performance in respect to our requirements. It is noteworthy to be mentioned that this process is repeated for both the training data (in order for the classifier to be trained to classify actual data) and the actual test data (where actual classification of unclassified data takes place). We decided to use a systematic approach to tackle the problem of data processing to distinguish users and for this reason, we introduced a generic framework for the analysis of data coming from their daily usage of their computing devices.

The main starting goal is to build a comprehensive dataset for training statistical classifiers and to apply this to test data to establish possible patterns and thus identify distinct human activities for the purposes of identification. The reason we chose to define a generic framework is to establish a proper methodology/framework of doing similar experiments, as well as to facilitate further developments in

the domain. The framework is based and built on the principles of the aforementioned broad approach and is illustrated in Figure 2. Its main elements are detailed in the following:

- Define problem space: the particulars of the user identification tasks need to be carefully defined at a high-level, namely what needs to be achieved. They will serve as the requirements that will drive the rest of the analysis process.
- Define activities to be identified: not all scenarios for user identification rely on the observation of the same set of activities and patterns of usage. However, the selection of the interesting activities, e.g. network statistics, is important at an early stage since it drives the definition of the required data and monitoring schemes/applications to monitor these activities.
- Define monitors to be used: having defined the activities we are interested in, the next step concerns the selection of the most appropriate monitoring applications (see next Section) to support the identification of different users. A preliminary feasibility study on the capabilities of the available data collection applications implemented on the considered platform (PC or smartphone) should take place at this stage, complemented by a literature survey to pinpoint related work by other researchers that could serve as inspiration for our work here.
- Plan and conduct training experiments: the training phase is the first phase in the machine learning process and it involves a set of base experiments to collect reference data for the activities in question, in this case the usage/behavior of different users. The planning of these experiments is therefore of paramount importance, so as no configuration parameters or testing conditions become neglected during the following phase.
- Collect datasets for training: collect training data referring to elementary activities important for user identification as defined in previous steps of the process. The data should refer to more than one repetitions of the activity over a span of time.
- Pre-process training data: the collected data is in most cases noisy and needs to be pre-processed prior to being used by machine learning classifiers. Raw data has usually very fine granularity and it is difficult therefore to discern statistical properties on this data. Therefore it needs to be processed in order to extract statistical features over time and frequency domains as discussed in forthcoming sections. The pre-processing tasks involve the removal of outliers from the original dataset, the annotation of the data for classification purposes (applies only to training data, since the classifier will “predict” the class of the test data) and the extraction of the features as discussed before. The latter is the lengthiest task.
- Build classifier based on training data: amongst the large number of classification algorithm available in the related literature, the optimal one for the particular type of collected data and extracted features should be selected. 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 in order to conclude on the most appropriate classifier for the considered experiment's settings.

- Apply classifier on test data: 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 should be able to identify the class to which each of the test data belongs and it decides upon that based on the similarity and pattern matches of the test data features with the training data features.
- Evaluate accuracy: the accuracy of the classifier is evaluated against the ground truth, hence the need to properly and accurately annotate both the training and test data. In the absence of proper annotation it is very hard if not impossible to determine the accuracy of the classifier.
- Improve classifier: test and training data are of the same type but a lot of irregularities might appear on the test data that might not have been present in the training data. There are many reasons for this, most important of which is the fact that training data users are rarely the same as test data users and thus do not have the same PC/smartphone usage patterns. Therefore, the classifier might not predict the test data as efficiently as expected and further modifications need to be applied on it.

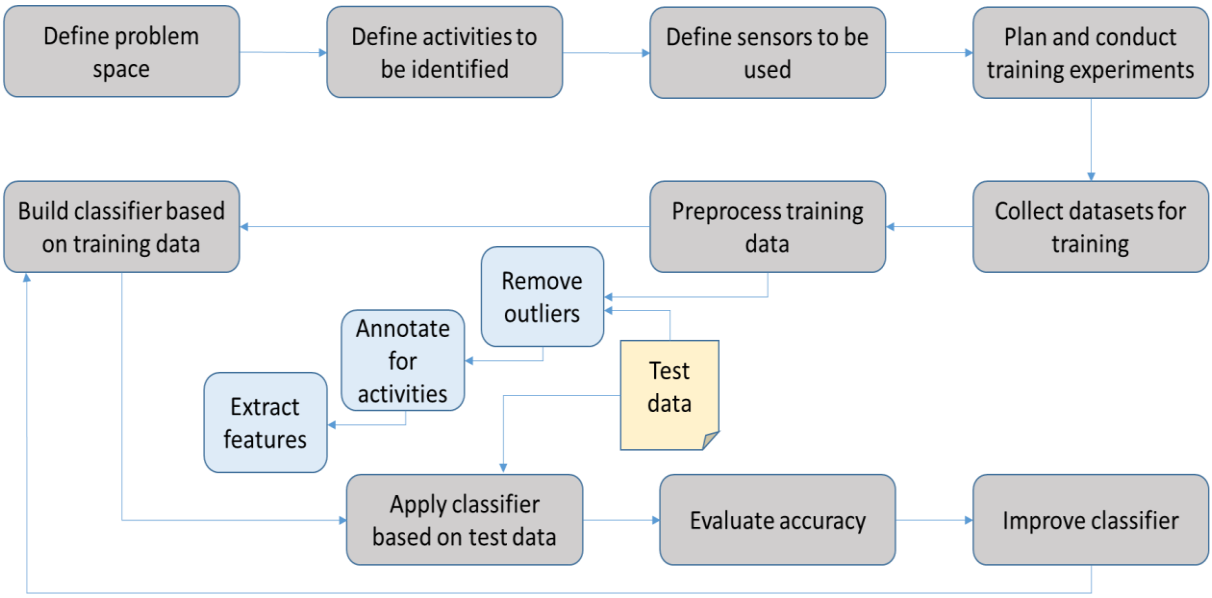

Figure 2. Generic framework for the analysis of user identification data

In what follows, we elaborate on these steps of our proposed framework in the context of the user identification project and initially just for PC usage.

4 Data collection

4.1 System Feature Extraction

While various built-in tools (*e.g.,* top, ps, *etc.,)* can be used to monitor system features, none of them provide a transparent and "uniform" access to the raw data. Thus, we designed a system monitor, as illustrated in Figure 3, consisted of a General and Process level monitor modules. The General monitor, as its name implies, collects system general health data. Specifically, monitors the following system's components:

- CPU
- Memory
- Network

Tables 1, 2, 3, and 4 overview the collected features. The process level monitor, on the other side, collects for each process the same type of information as Table 5 and 6 illustrate. The development of these modules relied on SIGAR API [50] .

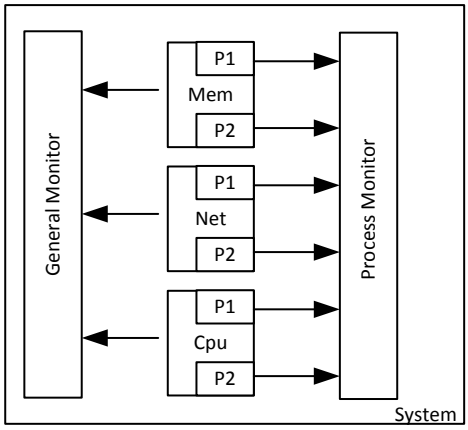

Figure 3. Monitor Architecture

Feature	Description
Idle time	Total time in which CPU does nothing
Time servicing interrupts	Total time in which CPU serves interrupts
Nice time	Total time in which CPU executes user level process with *nice* priority
Time servicing softirqs	Total time in which CPU serves software based interrupts
Involuntary wait time	Total time in which the real CPU was not available to the current virtual machine (useful only to cloud environments)
Kernel time	Total time in which CPU serves process requesting access to kernel services.

CPU user time	Total time in which CPU serves users' processes
System CPU time	Total time in which CPU serves users' processes, including the kernel time. This is, the accumulation of Kernel and user time.
System CPU io wait time	Total time in which CPU waits for input/output.

Table 1. CPU Features Short Description

Feature	Description
Free system memory	System's free memory
Random Access Memory	System's random Access memory
System memory	Total system's memory
Used system memory	Total used system memory

Table 2. Memory Features Short Description

Feature	Description
Active opens	The number of times TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state
Attempt fails	The number of times TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state
Current established	The number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT
Established resets	The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state
Input errors	The total number of segments received in error
Input segments	The total number of segments received, including those received in error. This count includes segments received on currently established connections
Out resets	The number of TCP segments sent containing the RST flag
Out segments	The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets
Passive opens	The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state
Retransmitted segments	The total number of segments retransmitted – that is, the number of TCP segments transmitted containing one or more previously transmitted

	octets

Table 3. TCP MIB Features Short Description (see RFC 1213)

Feature	Description
Inbound total	The total number of incoming packets
Outbound total	The total number of outgoing packets
TCP Close	The total number of TCP completed connection
TCP Close Wait	The total number of TCP connection received the first FIN signal from the client and the connection is in the process of being closed
TCP Closing	The total number of TCP connections waiting for termination request acknowledgment from the remote.
TCP Wait 1	The total number of the TCP connections is still active but not currently being used
TCP Wait 2	The total number of the TCP connections just received acknowledgment of the first FIN signal from the server
TCP Established	The total number of TCP established connections
TCP Idle	The total number of idle TCP connections
TCP Inbound	The total number of TCP inbound connections
TCP LastAck	The total number of TCP sockets waiting for the last acknowledgement to close the socket
TCP Listen	The total number of TCP sockets waiting for incoming connections
TCP SynRecv	The total number of TCP connection requests that has been received from the network
TCP Outbound	The total number of TCP outbound connections

Table 4. Net Stat Features Short Description

Feature	Description
Effective group id	Process effective group id
Effective user id	Process effective user id
Group id	Process group id
User id	Process user id
Process file descriptors	Total number of open file descriptors
I/O page faults	Total number of page faults due to I/O operations
Non I/O page faults	Total number of non I/O operations
Number of page faults	Total number of page faults
Process resident memory	Total process RAM memory

Process shared memory	Total process shared memory
Process virtual memory	Total process virtual memory
Processor number	The number of processor where process runs
Process status	The status of process
Active threads	The number of active threads
Process parent id	Process parent's id
Process Priority	Process kernel scheduling priority

Table 5. Process Level Information

Feature	Description
Process CPU kernel time	Total time in which CPU serves process requesting access to kernel services.
Process CPU user time	Total time in which CPU serves users' processes
Process CPU time	Total time in which CPU serves users' processes, including the kernel time. This is the accumulation of Kernel and user time.

Table 6. CPU Process Level Information

4.2 Experiments

As an initial test campaign we monitor and record the above mentioned features for four users, while they perform a number of common activities in a shared computer for three hours. We should mention that this is a first test campaign in the direction to validate our initial hypothesis for user identification relying on general purpose system's features.

5 Data processing

At a first stage, we focused on the four parameters concerning general system properties, i.e. CPU, memory, TCP MIB and network statistics, as detailed before. The collected training data refers to related measurements using the aforementioned tools and need to be pre-processed prior to making any analysis based on them. The aim of the data processing stage is to cleanse the collected data set in order to remove outliers and possible erroneous entries and to extract statistical features that characterize the data. Typical machine learning algorithms commence by cleaning the raw data and removing the outliers, followed by the application of a windowing technique to extract groups of data that could potentially expose repetitive activities or tasks with common characteristics. The next step involves extracting features from the actual raw data in order to derive statistical properties about them and the process concludes by passing the data through a classifier for analysis, classification and prediction if needed.

5.1 Cleaning the dataset

Removing outliers is a very important task in data pre-processing, since these values could influence the outcome of the analysis process as they are not conforming to the rest of the data and have thus been potentially generated by side activities to the one currently under examination [2] . The first technique we use to clean the dataset is to remove the influence stemming from the initialization and finalization of activities, since users are normally at that time are being acquainted with the process or they are completing it, respectively. We are interested in the execution of the actual activity and therefore we trim the dataset on both ends accordingly. This is similar to the technique used in [1] and we take into account the risk of significantly reducing the size of the dataset.

In addition, outliers in the midst of the dataset might exist. The seminal work by Krumm [3] suggests a number of ways that could be applied to identify and correct outliers that exist in the middle of the dataset and thus could not be removed by simply trimming the latter. Typical examples include the application of various signal processing filters such as mean, Kalman and particle filters [4] . Since the application of these filters is subject to intense processing requirements, we opted against applying any such technique on the collected data. In particular, the mean filters necessitate knowledge of the entire dataset and could therefore not be considered in our case since this knowledge will be absent in the case of real-time test data.

5.2 Windowing technique

Subject to the cleaning of the dataset having being successfully completed, a windowing technique/algorithm is applied to create logical instances, i.e. windows, of the original dataset. The

windows are used to reduce the problem space on one hand, but also to assist in grouping similar samples on the other hand. For example, a sole reading of a user's CPU utilization might not yield anything significant, whereas a collection of readings referring to a longer time period might be adequate to indicate the occurrence of a specific activity on behalf of the user. In [6] a discussion on the different windowing techniques applied to raw data for the purpose of recognition of human physiological activities is presented, but subject to generalization we can extrapolate similar techniques for the considered dataset and the patterns of user behaviour when utilizing a computer. Three different windowing techniques can be employed, namely:

- Sliding window: the dataset is divided into windows of equal size, with equal length and no gap between them. It can be easily applied to real-time data and most often is considered with overlap, i.e. a certain amount of data from the end of one window is repeated over the second one. It is quite popular due to its simplicity, e.g. in [1] and [5] it was used with and without overlap; when overlap was considered it was set to be 50%.
- Event-based window: an initial activity recognition process needs to take place in order to set the windows that correspond to particular events, e.g. user initializing a specific program or commencing a network transfer.
- Activity-based window: requires as before some initial data pre-processing and activity recognition. In this case, activities could refer to users performing specific tasks on a computer.

At the current analysis we use only the sliding technique with overlap 50% over the entire training data population as it is well test approach, and can be easily applied to real-time data as mentioned previously. We do not consider event and activity based windows since both require to predefine events and activities of user interests, while at this early stage we are looking into general computing properties for user identification. Table 7 illustrates the different values for the window size that we plan to experiment with and the corresponding duration of the window and generated instances in the dataset based on data collection frequency of approximately 1ms.

Table 7. Experimental settings for the window size and size of instances in the training/testing datasets.

Window size	Duration	Instances in dataset
64	64ms	96464
128	128ms	48232
256	256ms	24116
512	512ms	12058
1024	1.024 s	6029

5.3 Feature extraction

The original training dataset is very large (hundreds of thousands of entries in line with high frequency of data collection) 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 regards 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. In Figure 4 for example, an excerpt of the actual total used system memory is illustrated for a particular user. It is evident that it is difficult to discern any meaningful information from such a complex data series. This is the reason that the windowing technique is applied to the data on the one hand, and on the other hand, the feature extraction process is performed over the different instances (windows) of the data. The notion of features refers to statistical properties of data windows and provides some qualitative or quantitative information on them.

Broadly speaking, there exist two types of features, namely time-domain and frequency-domain ones. One can also consider wavelet analysis features, but they remain outside the scope of this work so far. We examined the features selected in the related literature and in particular the ones discussed[1] in [1] ,[6] ,[7] ,[8] ,[9] and we went ahead and calculated 13 features for all the types of data contained in each of the windows in the dataset (5 types of data for the CPU, 6 for the memory and 7 for the network statistics and TCP MIB). Out of these 13 features, 8 were in the time domain and 5 in the frequency domain. The latter necessitated that we first apply a Fast Fourier Transform (FFT) over the windows' data and then calculate the corresponding features. It has to be underlined that this feature set needs to be also calculated for the test set apart from the training dataset.

Moreover, the training of the classifier will determine whether all of these features will be used for the classification of the test data or just a subset of them and in general the optimal configuration of the classifier (most appropriate set of parameters based on the performance of the classifiers).

[1] We recognize that the literature mostly refers to activity recognition using accelerometer data and this is due to our previous work on the subject; nonetheless, statistical feature extraction can be generalized and accordingly applied to computer-related metrics.

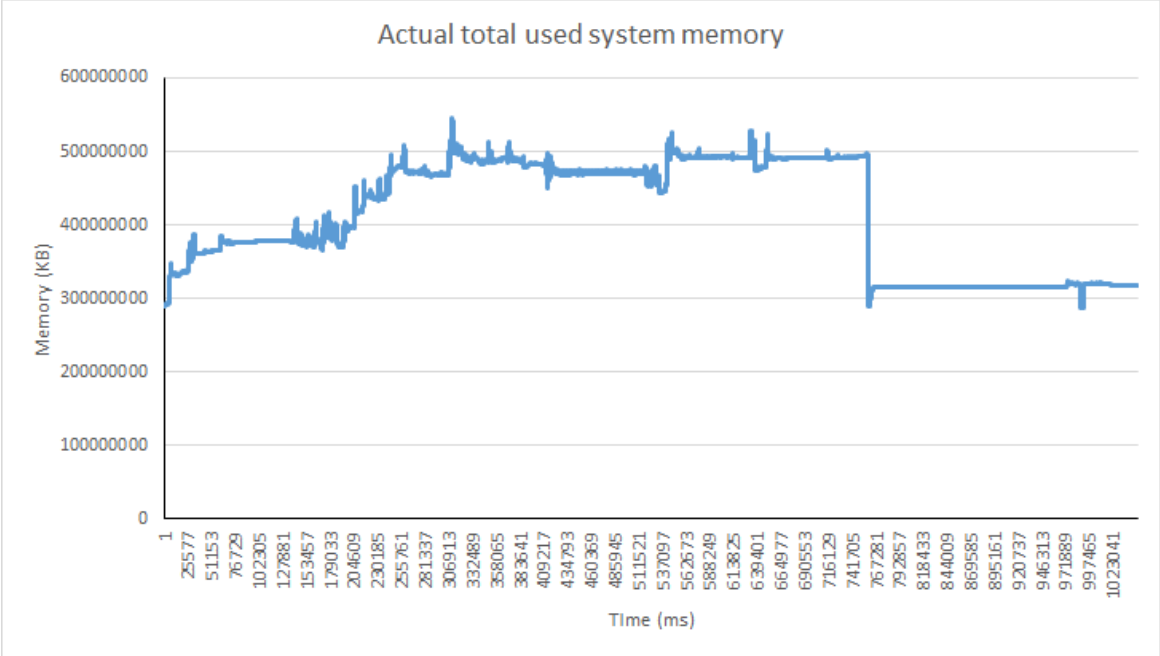

Figure 4. Actual total system memory for a particular user working on a PC.

5.3.1 Time domain features

We considered the values for each of the attributes as described in Section 2. We have collected data regarding parameters of the CPU, memory, TCP MIB and network statistics. For each of the considered instances (windows of data) and for each of the aforementioned computing parameters the following time-domain features were calculated:

- Mean (μ)
- Geometric mean
- Median
- Minimum
- Maximum
- Variance (σ^2)
- Standard deviation (σ)
- Root mean square: it is defined as shown in equation (1) for a variable x with N samples

$$rms\ x = \sqrt{\frac{1}{N}\left(x_1^2 + x_2^2 + \cdots + x_N^2\right)} \quad (1)$$

We plan to expand the feature set by considering correlations and covariances between different variables, using features such as:

- Covariance: it is symmetric and is computed pairwise for all parameters. It indicates a linear relationship between two variables, and for two variables x,y with a population of N is computed as in equation 2: $\sigma\ x, y = \sum_{i=1}^{N} \frac{x_i - \mu_x\ \ y_i - \mu_y}{N}$ (2)

- Pearson's correlation: it is symmetric and is computed pairwise for all parameters. It indicates any statistical relationship between two variables, and for two variables x,y with a population of N is computed as in equation 3: $cor\ x, y = \frac{\sigma\ x,y}{\sigma_x \times \sigma_y}$ (3)

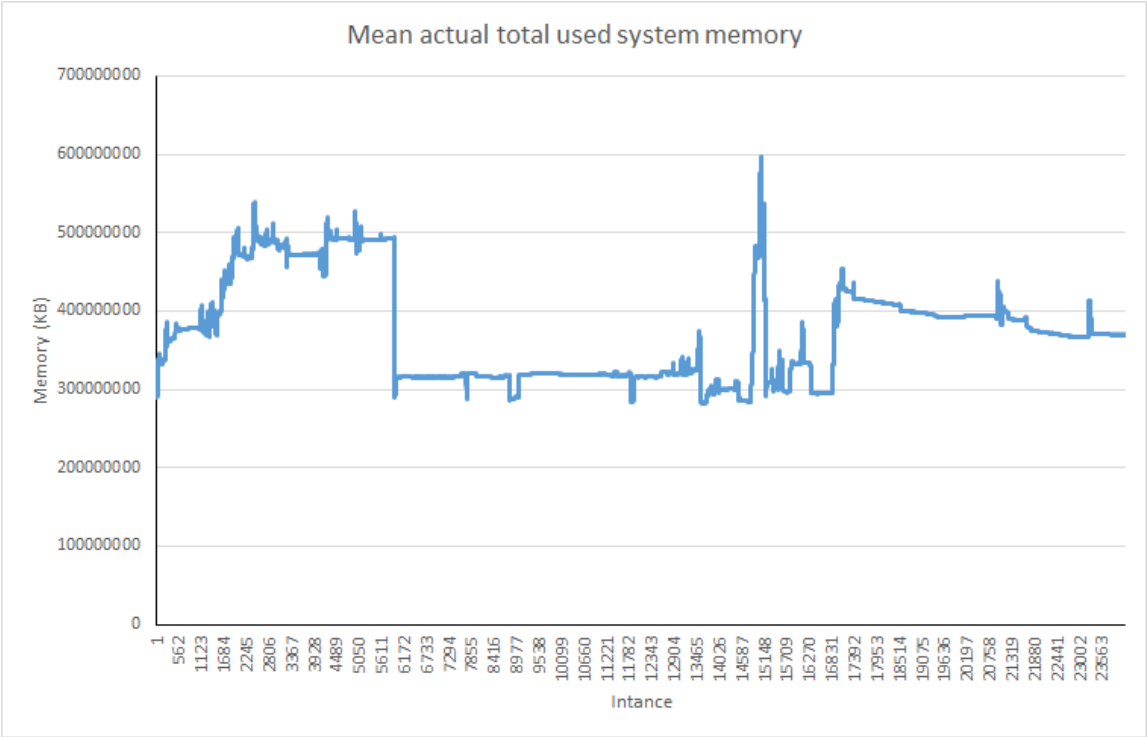

Figure 5. Mean (average) actual total used system memory for a particular user working on a PC.

After extracting the features from the raw data the dataset is significantly smaller and hence easier to be understood and interpreted. Nonetheless, it still does not allow for deduction of conclusions on classification and prediction of activity classes, as it is clear in Figure 5.

5.3.2 Frequency domain features

The previous features exposed interesting aspects about the data in the time domain. However, significant properties about the collected data can also be identified when considering the frequency domain. For example, repetitive or cyclic patterns of activities could be pinpointed, e.g. user accessing the browser every 5 minutes or user saving a file every couple of minutes, and used for the identification of distinct users' patterns, i.e. signatures. Therefore, on each of the instances that are derived from the sliding window process, we first apply FFT over each of the considered parameters to derive frequency domain features over the samples of each window. Using the FFT coefficients for these 25 parameters that we obtained from this process, we calculate the following features:

- Mean (also referred to as the DC – Dominant Component)
- Median
- Minimum

- Maximum
- (Spectral) energy: sum of the squared FFT coefficients for a particular window, divided by the window length for normalization

We plan to expand the feature set by considering correlations and covariances between different FFT variables, as well as using additional features such as:

- Entropy (normalized information entropy): According to [8] entropy is used to distinguish between diverse signals with similar energy, but reflect different activity patterns. It is computed as shown in equation 4, for a variable x with population N and normal distribution.

$$entropy\ x = -\sum_{i=1}^{N} \frac{x_i}{\sum_{i=1}^{N} x_i} \times \log \sum_{i=1}^{N} \frac{x_i}{\sum_{i=1}^{N} x_i} \quad (4)$$

- Pearson's correlation: it is symmetric and is computed pairwise for the FFT coefficients of all parameters. It indicates any statistical relationship between two variables, and for two variables x,y with a population of N is computed as in equation 3.

Cyclic patterns of user activities can be easily observed using such features and this is the main reason for their significance. Indicatively, Figure 6 depicts the median FFT for the actual total system memory used; cyclic patterns can be clearly seen in the activities of the considered user and these can be potentially exploited for distinction between users.

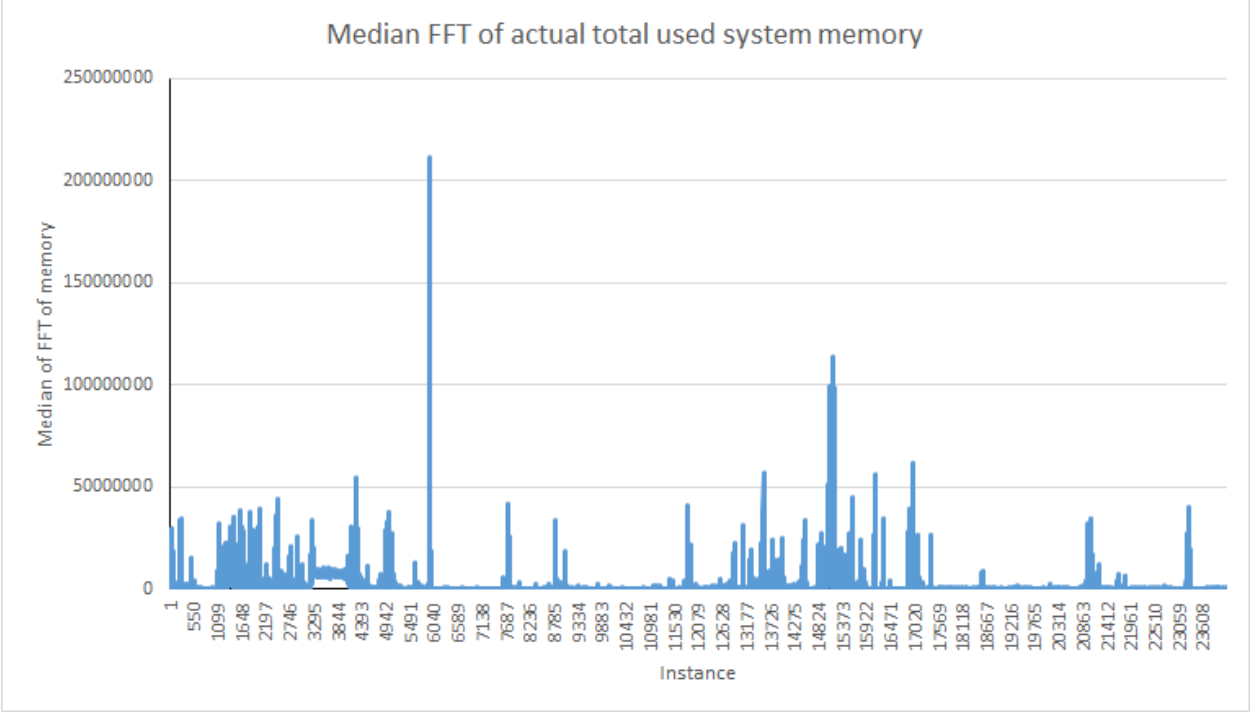

Figure 6. Median FFT of actual total used system memory for a particular user working on a PC.

6 Classification

In accordance to our generic framework (refer to Section 3), the next step in the process involves training the classifier, a machine learning classification algorithm, in order for it to be able to correctly analyze the collected data, but also to be able to respond to prospective queries on how new data should be classified, i.e. annotated, in line with previous data. In particular, the classifier should be able to reason on collected data that refer to user monitoring and deduce which user is using the computer at a particular moment in time.

There is a great variety of machine learning (or artificial intelligence or even signal processing) classification algorithms [12] [13] . To experiment with them there are a couple of options, spanning from custom implementation of their functionality using MATLAB for example to the use of tools specifically designed for this purpose, e.g. Weka[2] and PRTools[3]. We opted towards using Weka [11] for the following reasons:

- It has been used as the de-facto standard in academic literature for testing and experimenting with classification algorithms.
- It is open source software under the GNU GPL.
- It supports a rich set of functionalities, from data cleaning and preprocessing, to classification, clustering, visualization and analysis.
- The classifiers can be integrated into Java programs and can therefore be directly linked to real-time data collection and classification on Android or any other computing platform, and it can be easily integrated with the developed monitor (see Section 4).
- It allows for batch set of experiments to be conducted and for the corresponding analysis to take place in a holistic and comparative manner.
- The research team involved in the project has had previous experience with using this particular tool, namely the participatory surveillance project work of IPSC.G06.

We apply training based on 10-fold cross-validation on the training dataset mentioned before, where the data collection frequency was set to be once every 1 ms, the window size 256 with 50% overlap, 4 different users took place in the experiment that lasted of three sessions for an hour, were recorded for each users to account for diversity in their behaviour.

[2] Available at http://www.cs.waikato.ac.nz/ml/weka/, it allows for easy integration with Java programs.
[3] Available at http://prtools.org/ it is essentially a MATLAB toolbox.

6.1 Classifiers

We consider the following classifiers to apply in the extracted data set. The selection was made to accommodate classifier diversity and popularity in related academic literature.

J48: This classifier reflects Quinlan's C4.5 decision tree classifier reported in [29] 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 (smaller values incur more pruning), the minimum number of instances per leaf, the amount of data used for reduced-error pruning (one fold is used for pruning, the rest for growing the tree), whether reduced-error pruning is used instead of C.4.5 pruning, the seed used for randomizing the data when reduced-error pruning is used, whether to consider the subtree raising operation when pruning, whether pruning is performed and whether counts at leaves are smoothed based on Laplace.

Decision Table: This classifier refers to a simple decision table with a default rule mapping an unidentified instance to the majority class, as introduced in [34] Configurable parameters include the evaluation metric, which essentially is the measure used to assess the performance of attribute combinations used in the decision table. Possible options include accuracy, root mean squared error, area under ROC and mean absolute error, and the search method used to find good attribute combinations for achieving better results.

Bayes Networks (K2): This classifier is based on Bayes Network learning and utilizes a variety of search algorithms and quality measures [30] . The main configuration options for this algorithm include the selection of an estimator for finding the conditional probability tables of the Bayes Network (BayesNet, Simple, BMA, MultiNomialBMA), as well as the selection of a search algorithm used for searching network structures (K2, GeneticSearch, HillClimber, LAGDHillClimber, RepeatedHillClimber, SimulatedAnnealing, TabuSearch, TAN).

SVM[4]: The Support Vector Machine (SVM) classifier is based on supervised learning techniques and classifies training data instances into two possible output classes; it is therefore a non-probabilistic, binary, linear classifier [12] . There are numerous configuration parameters for this library, such as the type of SVM to be used (C-SVM, nu-SVM, one-class SVM), the cost parameter C or nu for the SVM, the coefficient, the type of kernel and many others that can affect the accuracy of the identification.

SMO: SMO (Sequential Minimal Optimization) [31] is used to train an SVM in a simplified manner and addressing the optimization problem present in training SVMs using traditional techniques.

k-NN: The k-NN is one of the most popular classifiers. Originally defined in [32] it is an instance-based learning algorithm, namely it compares new problem instances with instances seen in training. The

[4] SVM (Support Vector Machines) are not included by default in Weka. An open-source implementation by C.-C. Chang and C.J. Lin can be found at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/, and this is the one used for experiments in our work.

number k indicates how many “close by” instances in the training should be used, which have been stored in memory. Its simplicity is its major advantage, as well as the fact that it performs quite well. ~~T~~here are a number of configuration parameters for this algorithm, such as the number of neighbors to be used, whether to use distance weighting or not and the nearest neighbor search algorithm to be used (Linear Search, Ball Tree, KD Tree, Cover Tree) that can be used to achieve better accuracy.

MultiClass Classifier (meta-classifier boosting the performance of other classifiers): The MultiClass classifier is a meta-classifier, namely it exploits other base classifiers in order to produce more accurate classification results. It is capable of being applied to multi-class problems, allowing several parameters to be configured, e.g. selection of classifier, the method to use for transforming the multi-class problem into several 2-class ones, the use of pairwise coupling or not.

MultiLayer Perceptron Neural Network: This classifier [62] is based on a neural network that is using supervised learning techniques and a back-propagation algorithm to assign instances to classes.

7 Initial results

7.1 Experimental settings

As already mentioned data was collected for four distinct users working on the same machine for sessions of 1 hour and 3 hours correspondingly. The window size was set to be 256 with 50% overlap, and the users conducted activities as per their normal usage of a PC.

7.2 Evaluation criteria

To evaluate the considered classifiers one can use a variety of metrics, most prominent of which are the accuracy of the classifier, i.e. its ability to correctly predict the class of different instances (in this case the actual user that was active on the PC), and its overhead that is usually referring to the time required to train the classifier and to evaluate the test data against it. In both cases there has to be a training dataset, as well as a test dataset. The test dataset will not have any values for the class feature, since it is the job of the classifier to predict in which class each instance belongs to. However, to assess the accuracy of the classifier the ground truth regarding the dataset should be annotated (but not supplied to the classifier), in order to perform an a posteriori analysis of the accuracy of classification.

To test the accuracy of the classifiers presented in the previous section, we used the Experimenter tool of Weka and performed a comparative analysis of different configurations of the aforementioned algorithms using cross-validation with 10 folds and 10 iterations, with a training/test dataset percentage split of 66%. This means that two thirds of our training dataset was used for training, while one third was used for testing. Since we are aware of the ground truth in our training dataset, accuracy values can be calculated. Another similar way to achieve accuracy results regarding the classifier is to collect a second set of data from one of the users already in the training dataset, monitor him/her so as to know the ground truth (in this case the ground truth is the identity of the active user on a PC) and try to predict the class each instance belongs to (identify the actual user among all possible choices) using the different classifiers.

To gain a level of statistical confidence in the results obtained by the Weka Experimenter we applied the well-known statistical hypothesis test Students' T-Test, with a requested confidence of 0.05 (indicates statistical difference threshold when performing pairwise comparison between schemes), and managed to acquire measurements for a variety of aspects regarding accuracy and the performance of the classifiers accordingly. The list of tested classifiers and their parameters' configuration can be in seen in Table 8.

An important aspect in the experimentation involves the parameter selection for the classifiers that we evaluated. Relevant information can be seen in Table 8, while there are some further observations are detailed in the following.

- The J48 algorithm is extremely fast and overall produces great classification results for our training dataset, namely more than 99% in the majority of cases. Of course, it should be taken into account that the training dataset is extremely structured and has only 4 classes (the number of users who participated in our experiments), which clearly makes it a great candidate for tree-based classifiers. Modifying the confidence factor or the option to activate pruning or not did not significantly influence the accuracy of classification.
- Regarding the Bayes Network classifier, the most interesting setting involves the estimators regarding the network's parameters. We observed a slightly lower overhead in terms of time of the classifier when the Simple Estimator is being used. As far as the search algorithms to map the network's structure are concerned, K2 exhibits a great performance and at a small time, whereas HillClimber takes a longer time to build the model, but nonetheless performs really well.
- The sensitivity analysis of the SVM classifier focused on the type of SVM used for classification (S-SVM or nu-SVM) and consider the default values for the other parameters. The results that we collected regarding SVM were in general not at all encouraging. In order to improve the results we experimented with the remaining parameters, and it became evident that using a different kernel than the default one greatly increased the accuracy of the classifier. In particular, the default kernel was using a radial basis function that did not perform well. We modified it to accommodate a linear function and the results were much improved. It should be noted that the same results could be obtained using a polynomial function, while the sigmoid function is the worst in terms of both time to build the model and the accuracy of the classifier.
- We chose to use SMO with the default parameters and the results were not satisfactory. Moreover, due to the nature of the NETSTAT data, the SMO classifier could not be applied on it. SMO works by classifying instances in a pairwise manner according to the considered classes, e.g. userA vs. userB, etc. Different options for the kernel also yield similar results, however the default choice for polynomial kernel is the optimal one for our training dataset.
- The simplest of the algorithms, namely k-NN, was tested using various values for the amount of neighbors used, namely the variable k. We also experimented with the search algorithm. For the full property set regarding CPU, with a value of k equal to 5, 24.9573% accuracy can be reached, while a value of 10 brings it down to 24.8887%, thus highlighting that higher k values cannot bring much improvement to the classification process, if any at all. Moreover, for this dataset there is no influence of the search algorithm in the performance of the classifier, whereas the use of a distance weighting function only marginally increases accuracy.
- The choice to use pairwise coupling for the MultiClass classifier (proposed in [33]) has been shown to yield very good classification results. Evidently, the accuracy levels that can be reached using the multi-classifier are quite high; they come nevertheless at a relatively high processing cost and increased training time.
- It is important to note that MLP was found to be one of the most time consuming classifiers (from a low of approximately 59 seconds to a maximum of around 2500 seconds) when it comes to building the model for the training data. Conversely, the accuracy levels are

comparable to previous classifiers (an impressive 99.98% when considering all CPU properties), constituting the MLP as a non-preferable candidate for our work.

The MultiClass meta-classifier was tested with all other classifiers (with the same parameter settings) to examine the potential of improving their performance. It is important to note that it was not possible to test all algorithms on all datasets, due to the particularities of the considered datasets in terms of considered data types.

Table 8. Classifiers used for experimentation in Weka regarding prediction accuracy and classification performance.

#	*Classification algorithm*	*Parameters in Weka*
1	C4.5 decision tree	trees.J48 '-C 0.25 -M 2'
2	Bayes Network	bayes.BayesNet '-D -Q bayes.net.search.local.K2 -- -P 1 -S BAYES -E bayes.net.estimate.SimpleEstimator -- -A 0.5'
4	DecisionTable	DecisionTable –X 1 –S weka.attributeSelection.BestFirst –D 1 –N 5
5	SVM with linear	functions.LibSVM '-S 0 -K 0 -D 3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1 -seed 1'
6	SMO	functions.SMO '-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyKernel -C 250007 -E 1.0\"'
7	k-NN (k=5)	lazy.IBk '-K 5 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"'
8	Multilayer Perceptron Neural Network	meta.MultiClassClassifier '-M 0 -R 2.0 -S 1 -W functions.MultilayerPerceptron -- -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a'
9	MultiClass	meta.MultiClassClassifier '-M 3 -P -R 2.0 -S 1 -W “classification algorithm 1-7 with same parameters as above”'

As mentioned before we collected datasets regarding 4 distinct computing features of each user’s session, namely CPU and memory utilization, TCPMIB information and NETSTAT usage data. Results regarding the machine learning classifiers applied on these datasets are presented in Appendix A only for CPU as the remaining follow a similar trend. Briefly, we present all relevant results under the following conventions:

- Results are shown for each of the 4 computing features, considering all of the different properties for each as a whole and individually. For example, CPU has 5 properties associated to it, therefore we present one set of results considering all of them collectively, as well as each one individually. This is done to examine the influence of each of the monitored properties.

- Each of the 4 users conducted 3 different sessions using the same experimental configuration. We present results considering all the sessions of users collectively (3 sessions represent one user), as well as individually (each session of a user considered to have a unique signature). This is done to examine the potential of the classification algorithms to reason under higher levels of uncertainty; in this case, there are fewer data to consider per class, i.e. per user, and there are conceptually more users to classify.

As can be seen in the tables in Appendix A, the metrics we utilized for evaluation include (TP=true positive, FP=false positive, TN=true negative, FN=false negative):

- Accuracy: defined as the percentage of correctly classified instances over all classified instances. (TP+TN/TP+TN+FP+FN)
- % incorrect: defined as the percentage of incorrectly classified instances (1-accuracy)
- Mean absolute error: defined as the mean absolute error of classification
- Root mean squared error: defined as the root mean squared error of classification
- IR precision: defined as the proportion of instances that have truly been identified as belonging to a particular class amongst all the instances that have been identified as belonging to that class. (TP/TP+FP)
- IR recall: defined as the percentage of correctly classified instances of a class, over all instances of the class (TP/TP+FN)
- F-measure: defined as the weighted harmonic mean of IR precision and IR recall and is computed by 2*IR_Precision*IR_recall/(IR_precision+IR_recall)
- Area under ROC: defined as the area under the receiver operating characteristic; a value of 1 denotes excellent classification, with the quality of classification dropping as the value reaches 0.5.
- Elapsed time training: defined as the time required to train the classifier.
- Elapsed time testing: defined as the time required to classify the test data using the trained classifier.
- KB mean information: defined as the information-based evaluation criterion for each classifier's performance according to Kononenko and Bratko [14]
- Kappa statistic: measures the agreement of prediction compared to the true class (ground truth). A value of 1 indicates complete agreement.

7.3 Analysis

Our experimental analysis raised numerous observations regarding the collected results, which we summarize in the following.

The particularities of the collected data in conjunction with the operational functions of the classifiers (some of them are better at handling binary data over decimal ones for example) lead to great differences in terms of their performance. For example, when considering accuracy based on all CPU properties this can be as high as 100% (Bayes Network, C4.5 and Decision Table), but if SMO or SVM were to be considered these values would drop to 23.93 and 24.99 respectively.

In general, performance of the classifiers is higher when all 3 sessions of a user are considered in conjunction compared to when they are considered individually. For the CPU (all properties considered) SMO has only 7.9863% accuracy in the latter case, while this climbs to 23.93% in the former case. Similarly, as far as memory is concerned when all sessions are taken collectively into account then accuracy is found to be 79.7578%, a number that decreases to 70.2537% when these sessions are examined independently.

Regarding CPU (all properties and all user sessions collectively considered), we can clearly see the excellent accuracy exhibited by the Bayes Network, C4.5 and Decision Table classifiers, as well as the MultiClass classifier when the aforementioned 3 are applied on it. The latter observation is evident since the MultiClass classifier –being a meta-classifier itself- aims to boost the performance of other classifiers; in the particular case, since the accuracy of the classifiers originally was found to be 100%, it is evident that any boosting would lead to 100% accuracy as well.

In general, we can observe that for the particular datasets and configurations that we experimented with, the effect of the MultiClass classifier was not to be significant. Minor increase in the accuracy, at a significant cost in terms of elapsed training and testing time. Interestingly enough, the mean absolute error rate and the root mean squared error, were both found to be higher in the case of the MultiClass classifier despite the fact that the accuracy remained at 100% (in the case of CPU with all properties and all user sessions collectively considered the mean absolute error rose from 0 to 0.3 for the C4.5 and Bayes network classifiers). This is because of the multiple repetitions of the classification process in the case of the MultiClass classifier, which would inevitably lead to a higher number of errors to be produced. Nonetheless, as observed the accuracy on average should remain unaffected. Another interesting aspect to take into account when examining the results on the MultiClass classifier, is the fact that the KB mean information metric is significantly reduced compared to the standalone execution of the considered classifiers. For example, for the SMO classifier the KB mean information value dropped from 0.2072 to 0.0132 when the MultiClass classifier was considered. Kononenko and Bratko in [14] introduced an information-based evaluation criterion for each classifier's performance, which is quite interesting in that it excludes prior class probabilities and thus assesses better the performance of the classifier under uncertain conditions. Once again, the performance of the MultiClass classifier applied on standard classifiers was much lower. In our view, the reason is based on the construction of

the MultiClass classifier that inherently requires a larger number of classes and therefore the related probabilities are much lower by definition and hence its performance is bound to be reduced.

In the following, we discuss in detail results pertaining to the CPU related data. The C4.5, Decision Tale and Bayes Network classifiers all exhibit 100% accuracy. While this result was initially perceived to be due to some erroneous configuration of the classifier, in depth study showed that the accuracy is perfect because of the nature of the collected CPU data. In fact, these classifiers operate using a true/false logic and a highly distinguishing value in one of the considered features would explain the obtained results. The maximum frequency of the FFT transformation over the considered windows is the feature that has a very distinct value for each of the users that participated in the experiments and therefore has led to the 100% accuracy of the aforementioned classifiers. It is noteworthy that this feature has a highly distinguishing value for all 5 of the considered CPU properties, namely total system CPU time servicing softirqs, system CPU kernel time, system CPU time, system user time and total system CPU IO wait time, as well as when considering all of the properties collectively. At this stage and with the limitations of our experimental settings (in particular the small number of users and the fact that only one PC was used) we cannot be conclusive about our findings, with two directions being spawned:

- Assuming that the data collected is correct, namely there is no bias from the experimental PC and indeed these properties (actually the maximum FFT feature of these properties) have a distinguishing value for every different user independently of the time he/she used the PC, then the C4.5, Decision Tale and Bayes Network classifiers are deemed ideal to pinpoint the identity of users for whom they have been trained. A clear limitation is the fact that the algorithm needs to be trained with the data pertaining to a large number of users and the identification only takes place –albeit at 100% accuracy- only when there has been data collected for the particular user. Nonetheless, this could cause a significant threat to user anonymity when considering the perfect identification results. Having access to such information from a user could be used to identify the user in prospective scenarios and could threaten his/her privacy.
- Based on the previous discussion, it is essential that further experimentation takes place taking into account all possible parameters that might have influenced the collected data and led to possible erroneous calculation of the maximum FFT value for the aforementioned properties. Such parameters include the number of users, the number of sessions, the equipment/software used, e.g. PC and SIGAR API, the frequency of data collection and the time spent on each session.

Aside from these 3 classifiers that exhibited excellent performance in terms of accuracy, the MLP one that is essentially a classifier based on neural networks had a very good accuracy in the classification,

namely 99.9886%. Conversely, the 5-NN and 10-NN classifiers did not have a very good performance, with accuracies of 24.9573% and 24.8887% respectively, whereas similarly SMO had an accuracy of 23.9304% and SVM of 24.9971%. Interestingly enough, despite the 100% accuracy of C4.5, Decision Tables and Bayes Networks, the Decision Table classifier had a mean absolute error of 0.0033 (root mean squared error of 0.0043). These errors increased almost 100 times when considering the equivalent MultiClass class of the Decision Table (0.3002 and 0.3467 respectively), while C4.5 and Bayes Networks also exhibited large error values when MultiClass was employed. As mentioned before, this is due to the iterative nature of the MultiClass classifier. Moreover, the fact that certain classifiers exhibit excellent accuracy but still lead to erroneous classifications is based on the fact that while these classifiers managed to correctly classify a larger number of instances, the misclassifications were so big that they succeeded in increasing the mean errors significantly (for example, the MultiClass Bayes Networks classifier has a mean absolute error value of 0.3 and accuracy 100%, whereas the MultiClass 5-NN has a comparable mean absolute error of 0.3749 and accuracy of merely 24.9573%). This observation is consistent with our belief that evaluation of a machine learning classifier is not a simple process of computing a couple of metrics, but rather an extensive procedure where a series of quantitative metrics should be taken into account and considered in parallel, while additionally one should not neglect qualitative analysis (namely confusion matrices), as discussed later.

Two very important metrics in assessing classifiers are the precision and the recall (these two have been borrowed from the field of Information Retrieval - IR) [35] . Precision measures successful assignments to a class over all assignments to that class (including incorrect ones) and in this respect, and in accordance to information retrieval theory, it refers to the fraction of classified instances that are relevant, i.e. correct. Conversely, recall refers to the fraction of relevant instances that have been classified. Therefore, high recall values indicate that the classifier was successful in classifying correctly most of the instances, whereas high precision means that the classifier performed more correct classifications than incorrect ones.

As far the CPU (all properties collectively incorporated) is concerned, precision follows in general the trend exhibited by the results regarding accuracy. The only exception is the fact that while SMO and SVM have comparable accuracy levels (23.93% and 24.99% respectively), the precision for the SMO is 4 times higher than that of the SVM (0.2393 and 0.0574 respectively). This is an indication of the better quality classification that can be achieved by SMO in comparison to SVM and even 5-NN and 10-NN and once again highlights the need to consider a number of evaluation metrics when comparing classification algorithms. This result becomes clearer when considering the recall rate, which for the SMO is perfect (equal to 1), same as the C4.5, Decision Table and Bayes Network classifiers. MLP has very good performance in terms of both precision and recall. Accordingly, a very important metric is the

F-measure that combines both precision and recall, also known as F1 score or F1 measure since precision and recall are evenly weighted [35] . It is actually the harmonic mean of precision and recall and is used to express the accuracy of the classification process and is widely considered to be more useful than the percent of correct classifications as expressed by the accuracy metric. According to the F-measure, the MLP, C4.5, Bayes Network and Decision Tables perform the best, while the SVM and 5-NN the worst.

Moreover, the area under ROC (receiver operating characteristic) is a metric that was first introduced in the theory of radar signal detection, but has also proven to be extremely useful in evaluating classification algorithms, as discussed by Spackman in his seminal article [36] , although its value has been recently heavily criticized and thus undermined, e.g. in [37] [38] . The ROC curve for a binary classifier plots in a 1x1 space the true positives (notion of sensitivity similar to recall) versus the false positives (1 – specificity, where specificity reflects the fraction of correctly identified false instances) and therefore an ideal classification would be at point (0, 1) in this space. The area under the ROC curve would thus be equal to 1 for a perfect classification and would drop as classification quality drops. In our experiments, as before the MLP, C4.5 Bayes Network and Decision Tables perform ideally (value of 1) in terms of the area under ROC, whereas the remaining classifiers all perform mediocre (value of 0.5). Other metrics that we considered in order to compare the performance of the considered classifiers include the Kappa statistic and the KB mean information. Kappa statistic is used to indicate the agreement of prediction compared to the ground truth and is important since it is a probabilistic value that takes into account not only the comparison to the ground truth, but also the probability that a correct assignment to a class was by chance. It is interesting to note that while the results follow the same trend as with the previous metrics, the quality of the classifications of SMO, SVM and k-NN is minimal as far as the Kappa statistic is concerned (they all rank the lowest possible value of 0). In terms of the KB (Kononenko and Bratko) mean information value, there is no discernible variation in the trend of results: in particular, MLP, C4.5, Decision Table and Bayes Network all achieved high values (1.9952, 1.9991, 1.9895 and 1.9991 respectively), whereas the lowest values where those corresponding to 5-NN and 10-NN (0.1736 and 0.1442 respectively).

The last metric that we considered was the aspect of time. In particular, we examined the time required to train the classifiers and the time required for them to perform classifications over the test data. Since the classifiers were trained and tested on the same sets of data the values obtained for these metrics are directly comparable. MLP, which is the most complex of the considered classifiers (back-propagation calculations inside the neural network), requires the most time for training, averaging 59.4759 seconds. This is however not reflected in the testing phase that only takes 0.007 seconds. All

other classifiers require less than 1 second for training (indicatively the C4.5 one requires a mere 0.1073 seconds), and even less to test, e.g. 0.0001 for C4.5.

7.4 Discussion

It is clear that there are tradeoffs to be considered when choosing the optimal classifier for the user identification project's needs in identifying which user –out of a pool of possible users- is actually active on a shared PC. We need to consider accuracy, precision, recall, as well as the overhead in terms of time for each of the classifiers since they will need to be considered in real time operation. The training phase will only occur once, so long timespans for this phase can be sidestepped, but long times for testing can be used to exclude certain classifiers from our candidates' list.

Evidently, quantitative results as those previously presented are important, since they provide a thorough evaluation of the performance of the different classifiers in regards to a variety of aspects. It is however equally important to be able to qualitatively analyze the classification process and in particular to be able to analyze why classification errors occur. The best way to do this is by checking the confusion matrix (also known as contingency table) of the classification process. Confusion matrices represent the classification results versus the ground truth. In particular, for n classes the confusion matrix is n*n and each row represents the actual classes, while each column the predicted class by the classifier.

A perfect classification process should yield a confusion matrix, all the entries of which aside to the main diagonal are 0 and the main diagonal entries correspond to the total number of instances in the considered dataset. Confusion matrices are important because they allow us to diagnose which classes were confused to each other and therefore be able to draw conclusions to why this occurred in the first place. For example, it is now thing to be aware of the accuracy of the classifier being 80%, and another thing to know that this is due to misclassification of userA to userB (this could for example indicate that some pattern of behavior between users A and B is very common). We can therefore gain additional knowledge and modify the classifiers accordingly to be able to distinguish better between these two activities, e.g. by considering or not certain features. We chose not to provide such an analysis here, since the set of users is quite small.

As mentioned previously, similar observations can me made regarding the other computational characteristics, namely memory, network and TCP MIB statistics. The main elements of the analysis referring to the optimal classifiers for the needs of this project are the ones presented regarding CPU. The results concerning the other characteristics broadly support our findings, albeit with some small variations in terms of the values for accuracy and other evaluation metrics. More details about the outcomes could be provided to the interested reader after a request to the authors.

8 Conclusions

We believe that initial outcomes are encouraging, however, as the examined user basis is limited we should extend it in order to validate the current observations. Also, in the current analysis we rely on time and frequency domain features, while there are proposals [7] applying wavelet transforms over the collected data to extract the appropriate features. We are considering using such a proposal in order to achieve better accuracy for user identification.

We should explicitly mention that training of the classifier and subsequent use of it to test and predict the classes of non-annotated data occurs with the consideration of all the extracted features. Thus, we plan to perform a thorough sensitivity analysis of the considered parameters, in particular the frequency of data collection and the window size. The reason to do so is the fact that when real-time classification of instances is concerned, not all features need to be extracted from the data corresponding to an instance. This could save much time and many computational resources and thus improve performance and maintain the real-time nature of the classification process.

Lastly, we need to highlight the need to perform a more widespread experiment with the participation of a greater number of users. This would be beneficial to establish the correlation of the number of users to the accuracy and overall performance of the algorithms. Moreover, it would serve as an indication of the scalability of our approach.

9 References

[1] Bao, L. and Intille, S.S., Activity recognition from user-annotated accelerometer data, in Proceedings of the 2nd International Conference on Pervasive Computing (PERVASIVE), Springer LNCS 3001, pp. 1-17, Austria, 2004.

[2] Kriegel, H.-P., Kröger, P., and Zimek, A., Outlier detection techniques, Tutorial at the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining. 2009.

[3] Krumm, J., Processing sequential sensor data, in Krumm, J. (ed.), Ubiquitous Computing Fundamentals, CRC Press, pp. 353-380, 2010.

[4] Oppenheim, I.V., Schafer, R.W. and Buck, J.R., Discrete-Time Signal Processing (3rd Ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[5] Ravi, N., Dandekar, N., Mysore, P., and Littman, M.L., Activity recognition from accelerometer data, in Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence (IAAI), MIT Press, pp. 1541-1546, 2005.

[6] Preece, S.J., Goulermas, J.Y., Kenney, L.P.J., Howard, D., Meijer, K., and Crompton, R., Activity identification using body-mounted sensors – a review of classification techniques, Physiological Measurement, Vol. 30, pp. 1-33, IOP Science, 2009.

[7] Preece, S.J., Goulermas, J.Y., Kenney, L.P.J., and Howard, A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data, IEEE Transactions on Biomedical Engineering, Vol. 56, No. 3, March 2009.

[8] Figo, D., Diniz, P.C., Ferreira, D.R., and Cardoso, J.M.P., Preprocessing techniques for context recognition from accelerometer data, Personal and Ubiquitous Computing, Vol. 14, No. 7, Springer, pp. 645-662, 2010.

[9] Hemminki, S., Nurmi, P., and Tarkoma, S., Accelerometer-based transportation mode detection on smartphones, ACM SenSys Conference, November 2013.

[10] Weka 3 - Data Mining with Open Source Machine Learning Software in Java, Official homepage, http://www.cs.waikato.ac.nz/ml/weka/, accessed November 2013.

[11] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I.H., The WEKA data mining software: an update, ACM SIGKDD Explorations Newsletter, Vol. 11, No. 1, pp. 10-18, November 2009.

[12] Witten, I.H., and 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, CA, USA, 2005.

[13] Han, J., Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[14] Kononenko, I., and Bratko, I., Information-Based Evaluation Criterion for Classifier's Performance, Machine Learning, Springer, Vol. 6, No. 1, pp. 67-80, January 1991.

[15] Dolan-Gavitt, B., Leek, T., Zhivich, M., Giffin, J., Wenke Lee, Virtuoso: Narrowing the Semantic Gap in Virtual Machine Introspection, 2011 IEEE Symposium on Security and Privacy (SP), pp.297-312, 22-25 May 2011.

[16] Aha, D.W., Kibler, D. and Albert, M.K., Instance-based learning algorithms, Machine Learning, Vol. 6, No. 1, pp. 37-66, Kluwer Academic Publishers, 1991.

[17] Bishop, C.M., Pattern recognition and machine learning, Springer-Verlag New York, ISBN 0-387-31073-8, 2006.

[18] Manuel Egele, Theodoor Scholte, Engin Kirda, Christopher Kruegel "A Survey on Automated Dynamic Malware Analysis Techniques and Tools" https://iseclab.org/papers/malware_survey.pdf

[19] Riboni D.; Bettini C., "COSAR: hybrid reasoning for context-aware activity recognition" *Pers Ubiquit Comput, 2011*. vol. 15 , no., pp.10 pp. 271-289.

[20] Jian Zhang; Figueiredo, R.J., "Application classification through monitoring and learning of resource consumption patterns," *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International* , vol., no., pp.10 pp.,, 25-29 April 2006 http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1639378

[21] Zhenhuan Gong; Xiaohui Gu, "PAC: Pattern-driven Application Consolidation for Efficient Cloud Computing," *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on* , vol., no., pp.24,33, 17-19 Aug. 2010 http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5581610

[22] Abrahao, Bruno; Zhang, Alex, "Characterizing application workloads on CPU utilization for Utility Computing", HP Technical Report HPL=2004-157, 2004 http://www.hpl.hp.com/techreports/2004/HPL-2004-157.html

[23] Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiaoyong Zhou, XiaoFeng Wang "Effective and Efficient Malware Detection at the End Host" https://www.usenix.org/legacy/event/sec09/tech/full_papers/kolbitsch.pdf

[24] Darren Mutz, Fredrik Valeur, Christopher Kruegel, Giovanni Vigna "Anomalous System Call Detection" http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.7101&rep=rep1&type=pdf

[25] Narcisa Andreea Milea, Siau Cheng Khoo,David Lo, Cristian Iuliu Pop "NORT: Runtime Anomaly-Based Monitoring of Malicious Behavior for Windows" http://www.mysmu.edu/faculty/davidlo/papers/rv11.pdf

[26] John Demme, Matthew Maycock, Jared Schmitz, Adrian Tang, Adam Waksman, Simha Sethumadhavan, and Salvatore Stolfo. 2013. "On the feasibility of online malware detection with performance counters" *SIGARCH Comput. Archit. News* 41, 3 (June 2013), 559-570. DOI=10.1145/2508148.2485970 http://doi.acm.org/10.1145/2508148.2485970

[27] G. Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, Bart Preneel "FPDetective: Dusting the Web for Fingerprinters" *CCS'13,* November 4–8, 2013, Berlin, Germany.

[28] Eckersley, P. (2010) 'How Unique Is Your Web Browser?', PETS'10 Proceedings of the 10th international conference on Privacy enhancing technologies, Lecture Notes in Computer Science, Berlin, Germany, Springer Berlin / Heidelberg. pp. 1-18.

[29] Quinlan, J.R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[30] Heckerman, D., Geiger, D., and Chickering, D.M., Learning Bayesian networks: the combination of knowledge and statistical data, in *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence* (UAI'94), Ramon Lopez De Mantaras and David Poole (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 293-301, 1994.

[31] 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, David A. Cohn (Ed.). MIT Press, Cambridge, MA, USA, pp. 557-563, 1998.

[32] Aha, D.W., Kibler, D. and Albert, M.K., Instance-Based Learning Algorithms, Machine Learning, Vol. 6, No. 1, pp. 37-66, January 1991.

[33] Wu, T.F., Lin, C.J., and Weng, R.C., Probability Estimates for Multi-class Classification by Pairwise Coupling, Journal Machine Learning, Vol. 5, pp. 975-1005, 2004.

[34] Ron Kohavi, The Power of Decision Tables. In Proceedings of the 8th European Conference on Machine Learning (ECML '95), Nada Lavrac and Stefan Wrobel (Eds.). Springer-Verlag, London, UK, UK, 174-189, 1995.

[35] Van Rijsbergen, C. J., Information Retrieval (2nd ed.), Butterworth-Heinemann, Newton, MA, USA, 1979.

[36] Spackman, K.A., Signal detection theory: valuable tools for evaluating inductive learning, in Proceedings of the sixth international workshop on Machine learning, Alberto Maria Segre (Ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 160-163, 1989.
[37] Hand, D.J., Measuring classifier performance: a coherent alternative to the area under the ROC curve, Machine Learning, Vol. 77, No. 1, Springer, pp. 103-123, October 2009.
[38] Lobo, J.M., Jimenez-Valverde, A., and Real, R., AUC: a misleading measure of the performance of predictive distribution models, Global Ecology and Biogeography, Vol. 17, No. 2, pp. 145-151, March 2008.
[39] Georgios Kambourakis. Anonymity and closely related terms in the cyberspace: An analysis by example. Journal of Information Security and Applications, vol. 19, no. 1, pp. 2-17, February 2014.
[40] Istemi Ekin Akkus, Ruichuan Chen, Michaela Hardt, Paul Francis, and Johannes Gehrke. Non-tracking web analytics. In Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12). ACM, New York, NY, USA, 687-698, 2012.
[41] Weining Yang, Ninghui Li, Yuan Qi, Wahbeh Qardaji, Stephen McLaughlin, and Patrick McDaniel. Minimizing private data disclosures in the smart grid. In Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12). ACM, New York, NY, USA, pp. 415-427, 2012.
[42] Andrew F. Tappenden and James Miller. 2009. Cookies: A deployment study and the testing implications. ACM Trans. Web 3, 3, Article 9, 49 pages July 2009.
[43] Nikiforakis, N.; Kapravelos, A; Joosen, W.; Kruegel, C.; Piessens, F.; Vigna, G., "Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting," Security and Privacy (SP), 2013 IEEE Symposium on , vol., no., pp.541,555, 19-22 May 2013.
[44] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. FPDetective: dusting the web for fingerprinters. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS '13). ACM, New York, NY, USA, pp. 1129-1140, 2013.
[45] Weidong Shi; Jun Yang; Yifei Jiang; Feng Yang; Yingen Xiong, "SenGuard: Passive user identification on smartphones using multiple sensors," Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on , vol., no., pp.141,148, 10-12 Oct. 2011
[46] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on GPS data. In Proceedings of the 10th international conference on Ubiquitous computing (UbiComp '08). ACM, New York, NY, USA, pp. 312-321, 2008.
[47] Gary M. Weiss and Jeffrey W. Lockhart. Identifying user traits by mining smart phone accelerometer data. In Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data (SensorKDD '11). ACM, New York, NY, USA, pp. 61-69, 2011.
[48] Kwapisz, J.R.; Weiss, G.M.; Moore, S.A, "Cell phone-based biometric identification," 2010 Fourth IEEE International Conference on Biometrics: Theory Applications and Systems (BTAS), pp. 1-7, 27-29 Sept. 2010.
[49] Daniel Garnier-Moiroux, Fernando Silveira, and Anmol Sheth. Towards user identification in the home from appliance usage patterns. In Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication (UbiComp '13 Adjunct). ACM, New York, NY, USA, pp. 861-868, 2013.
[50] SIGAR (System Information Gatherer) API, Hyperic, http://www.hyperic.com/products/sigar, accessed October 2014.
[51] Tao Feng, Jun Yang, Zhixian Yan, Emmanuel Munguia Tapia, and Weidong Shi. TIPS: context-aware implicit user identification using touch screen in uncontrolled environments. In Proceedings of the 15th Workshop on Mobile Computing Systems and Applications (HotMobile '14). ACM, New York, NY, USA, Article 9, 6 pages, 2014.

[52] Yinghui (Catherine) Yang, Balaji Padmanabhan, Toward user patterns for online security: Observation time and online user identification, Decision Support Systems, Volume 48, Issue 4, pp. 548-558, March 2010.
[53] Dominik Herrmann, Christoph Gerber, Christian Banse, and Hannes Federrath. 2010. 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 (NordSec'10), Tuomas Aura, Kimmo Järvinen, and Kaisa Nyberg (Eds.). Springer-Verlag, Berlin, Heidelberg, 136-154.
[54] C. Xu, C. Du, G.F. Zhao, S. Yu, A novel model for user clicks identification based on hidden semi-Markov, Journal of Network and Computer Applications, Volume 36, Issue 2, pp. 791-798, March 2013.
[55] Nikolay Melnikov and Jurgen Schoenwaelder. 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 (AIMS'10), Burkhard Stiller and Filip De Turck (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 167-170, 2010.
[56] Jozef Kapusta, Michal Munk, Peter Svec, Anna Pilkova, Determining the Time Window Threshold to Identify User Sessions of Stakeholders of a Commercial Bank Portal, Procedia Computer Science, Volume 29, pp. 1779-1790, 2014.
[57] B. Berendt, B. Mobasher, M. Spiliopoulou, and J. Wiltshire, Measuring the Accuracy of Sessionizers for Web Usage Analysis, Proceedings of the Web Mining Workshop at the First SIAM International Conference on Data Mining, April 2001.
[58] Bojan Blažica, Daniel Vladušič, Dunja Mladenić, MTi: A method for user identification for multitouch displays, International Journal of Human-Computer Studies, Volume 71, Issue 6, pp. 691-702, June 2013.
[59] Saira Zahid, Muhammad Shahzad, Syed Ali Khayam, and Muddassar Farooq. Keystroke-Based User Identification on Smart Phones. In Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection (RAID '09), Engin Kirda, Somesh Jha, and Davide Balzarotti (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 224-243, 2009.
[60] Shijia Pan, An Chen, and Pei Zhang. Securitas: user identification through RGB-NIR camera pair on mobile devices. In *Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices* (SPSM '13). ACM, New York, NY, USA, pp. 99-104, 2013.
[61] Philipp Mock, Jörg Edelmann, Andreas Schilling, and Wolfgang Rosenstiel. User identification using raw sensor data from typing on interactive displays. In *Proceedings of the 19th international conference on Intelligent User Interfaces* (IUI '14). ACM, New York, NY, USA, pp. 67-72, 2014.
[62] Kurt Hornik, Maxwell Stinchcombe, Halbert White, Multilayer feedforward networks are universal approximators, Neural Networks, Volume 2, Issue 5, pp. 359-366, 1989.
[63] John Demme, Matthew Maycock, Jared Schmitz, Adrian Tang, Adam Waksman, Simha Sethumadhavan, and Salvatore Stolfo. 2013. On the feasibility of online malware detection with performance counters. SIGARCH Comput. Archit. News 41, 3 (June 2013), 559-570. DOI=10.1145/2508148.2485970 http://doi.acm.org/10.1145/2508148.248597
[64] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. Journal of Computer Security, pages 151 – 180, 1998.
[65] Sarat Kompalli, “Using Existing Hardware Services for Malware Detection” In Proceedings of the First Language-theoretic Security (LangSec) IEEE Security & Privacy Workshop, 2014

Appendix A – Experimental results

CPU – all properties – individual sessions considered as standalone for each user

Classifier	*C4.5*	*Bayes Net*	*5-NN*	*10-NN*	*SVM*	*SMO*	*MLP*	*Decision Table*	*MultiClass Bayes*	*MultiClass C4.5*	*MultiClass sSMO*	*MultiClass DTable*	*MultiClass 5-NN*	*MutliClass 10-NN*
Evaluation Metric														
Accuracy	99.9686	100	8.2287	8.2545	8.3229	7.9863	100	100	100	99.9686	7.9863	100	8.3544	8.3002
% incorrect classifications	0.0314	0	91.7713	91.7455	91.6771	92.0137	0	0	0	0.0314	92.0137	0	91.6456	91.6998
Mean absolute error	0.0001	0	0.1528	0.1528	0.1528	0.1530	0.0010	0.0067	0.1503	0.1503	0.1528	0.1503	0.1528	0.1528
Root mean squared error	0.0023	0	0.3034	0.2892	0.3909	0.2818	0.0044	0.0121	0.2719	0.2719	0.2764	0.2719	0.2764	0.2764
IR precision	1	1	0.0168	0.0088	0.0088	0.0799	1	1	1	1	0.0799	1	0.0120	0.0064
IR recall	1	1	0.2100	0.1100	0.1100	1	1	1	1	1	1	1	0.1500	0.0800
F-measure	1	1	0.0311	0.0163	0.0163	0.1479	1	1	1	1	0.1479	1	0.0222	0.0118
Area under ROC	1	1	0.5	0.5	0.5	0.5	1	1	1	1	0.5	1	0.5	0.5
Elapsed time training	0.4150	0.4263	0.0004	0.0004	5.3239	0.3026	170.6869	2.2914	3.0306	1.4285	0.2558	18.9243	0.0198	0.0205
Elapsed time	0.0004	0.0250	1.9299	1.8950	0.1392	0.0015	0.0157	0.0011	0.1380	0.0025	0.0078	0.0166	20.5448	20.0178

testing														
KB mean information	3.5828	3.5839	0.4231	0.3241	0.1832	0.2715	3.5755	3.5248	0.2381	0.2381	0.0204	0.2372	0.0223	0.0212
Kappa statistic	0.9997	1	0	0	0	0	1	1	1	0.9997	0	1	0	0

CPU – property1 – individual sessions considered as standalone for each user

Classifier	***C4.5***	***Bayes Net***	***5-NN***	***10-NN***	***SVM***	***SMO***	***MLP***	***Decision Table***	***MultiClass Bayes***	***MultiClass C4.5***	***MultiClass SMO***	***MultiClass DTable***	***MultiClass 5-NN***	***MutliClass 10-NN***
Evaluation Metric														
Accuracy	99.8488	100	8.2287	8.2545	8.3229	7.9863	99.4835	100	100	99.8232	7.9863	100	8.3544	8.3002
% incorrect classifications	0.1512	0	91.7713	91.7455	92.0137	91.6771	0.5165	0	0	0.1798	92.0137	0	91.6456	91.6998
Mean absolute error	0.0003	0	0.1528	0.1528	0.1528	0.1530	0.0037	0.0067	0.1503	0.1503	0.1528	0.1503	0.1528	0.1528
Root mean squared error	0.0099	0	0.3034	0.2892	0.3909	0.2818	0.0250	0.0121	0.2719	0.2719	0.2764	0.2719	0.2764	0.2764
IR precision	0.9993	1	0.0168	0.0088	0.0088	0.0799	0.9993	1	1	0.9962	0.0799	1	0.0120	0.0064
IR recall	1	1	0.2100	0.1100	0.1100	1	0.9993	1	1	1	1	1	0.1500	0.0800
F-measure	0.9997	1	0.0311	0.0163	0.0163	0.1479	0.9993	1	1	0.9981	0.1479	1	0.0222	0.0118
Area under ROC	1	1	0.5	0.5	0.5	0.5	1	1	1	1	0.5	1	0.5	0.5

Elapsed time training	0.0509	0.0554	0.0008	0.0003	1.1315	0.1234	13.4694	0.2318	0.2379	0.1385	0.0526	1.4517	0.0079	0.0065
Elapsed time testing	0.0004	0.0041	0.1750	0.1781	0.0142	0.0009	0.0018	0.0006	0.0143	0.0013	0.0020	0.0037	2.1547	1.6034
KB mean information	3.5784	3.5839	0.4231	0.3241	0.1832	0.2715	3.5442	3.5248	0.2381	0.2378	0.0204	0.2372	0.0223	0.0212
Kappa statistic	0.9984	1	0	0	0	0	0.9944	1	1	0.9981	0	1	0	0

CPU – property2 – individual sessions considered as standalone for each user

Classifier	***C4.5***	***Bayes Net***	***5-NN***	***10-NN***	***SVM***	***SMO***	***MLP***	***Decision Table***	***MultiClass Bayes***	***MultiClass C4.5***	***MultiClass SMO***	***MultiClass DTable***	***MultiClass 5-NN***	***MutliClass 10-NN***
Evaluation Metric														
Accuracy	99.8745	100	8.2231	8.1945	8.3627	7.9863	99.7089	100	100	99.8460	7.9863	100	8.3430	8.2574
% incorrect classifications	0.1255	0	91.7769	91.8055	91.6373	92.0137	0.2911	0	0	0.1540	92.0137	0	91.6570	91.7426
Mean absolute error	0.0002	0	0.1528	0.1528	0.1527	0.1530	0.0029	0.0067	0.1503	0.1503	0.1528	0.1503	0.1528	0.1528
Root mean squared error	0.0083	0	0.3021	0.2893	0.3908	0.2818	0.0201	0.0121	0.2719	0.2719	0.2764	0.2719	0.2764	0.2764
IR precision	1	1	0.0200	0.0136	0.0064	0.0799	0.9986	1	1	1	0.0799	1	0.0160	0.0104
IR recall	1	1	0.2500	0.1700	0.0800	1	1	1	1	0.9964	1	1	0.2000	0.1300

F-measure	1	1	0.0370	0.0252	0.0118	0.1479	0.9993	1	1	0.9982	0.1479	1	0.0296	0.0192
Area under ROC	1	1	0.5	0.5	0.5	0.5	1	1	1	0.9982	0.5	1	0.5	0.5
Elapsed time training	0.0476	0.0525	0.0004	0.0004	1.1169	0.1193	13.2268	0.2195	0.2708	0.1449	0.0550	1.4600	0.0087	0.0070
Elapsed time testing	0.0003	0.0031	0.1860	0.1922	0.0139	0.0008	0.0018	0.0005	0.0088	0.0014	0.0022	0.0037	2.1713	1.8335
KB mean information	3.5793	3.5839	0.4269	0.3237	0.1841	0.2715	3.5546	3.5248	0.2381	0.2378	0.0204	0.2372	0.0221	0.0215
Kappa statistic	0.9986	1	0	0	0	0	0.9968	1	1	0.9983	0	1	0	0

CPU – property3 – individual sessions considered as standalone for each user

Classifier	***C4.5***	***Bayes Net***	***5-NN***	***10-NN***	***SVM***	***SMO***	***MLP***	***Decision Table***	***MultiClass Bayes***	***MultiClass C4.5***	***MultiClass sSMO***	***MultiClass DTable***	***MultiClass 5-NN***	***MutliClass 10-NN***
Evaluation Metric														
Accuracy	99.9686	100	8.2231	8.1945	8.3627	7.9863	94.1221	100	100	99.9686	7.9863	100	8.3430	8.2574
% incorrect classifications	0.0314	0	91.7769	91.8055	91.6373	92.0137	5.8779	0	0	0.0314	92.0137	0	91.6570	91.7426
Mean absolute error	0.0001	0	0.1528	0.1528	0.1527	0.1530	0.0208	0.0067	0.1503	0.1503	0.1528	0.1503	0.1528	0.1528
Root mean	0.0023	0	0.3021	0.2893	0.3908	0.2818	0.0855	0.0121	0.2719	0.2719	0.2764	0.2719	0.2764	0.2764

squared error														
IR precision	1	1	0.0200	0.0136	0.0064	0.0799	0.9938	1	1	1	0.0799	1	0.0160	0.0104
IR recall	1	1	0.2500	0.1700	0.0800	1	0.9961	1	1	1	1	1	0.2000	0.1300
F-measure	1	1	0.0370	0.0252	0.0118	0.1479	0.9946	1	1	1	0.1479	1	0.0296	0.0192
Area under ROC	1	1	0.5	0.5	0.5	0.5	1	1	1	1	0.5	1	0.5	0.5
Elapsed time training	0.0445	0.0536	0.0003	0.0003	1.1956	0.1164	12.9578	0.2313	0.2619	0.1393	0.0553	1.5622	0.0083	0.0076
Elapsed time testing	0.0003	0.0032	0.1958	0.1837	0.0154	0.0008	0.0017	0.0005	0.0086	0.0013	0.0022	0.0037	2.1661	1.9550
KB mean information	3.5828	3.5839	0.4269	0.3237	0.1841	0.2715	3.3385	3.5248	0.2381	0.2381	0.0204	0.2372	0.0221	0.0215
Kappa statistic	0.9997	1	0	0	0	0	0.9359	1	1	0.9997	0	1	0	0

CPU – property4 – individual sessions considered as standalone for each user

Classifier	***C4.5***	***Bayes Net***	***5-NN***	***10-NN***	***SVM***	***SMO***	***MLP***	***Decision Table***	***MultiClass Bayes***	***MultiClass C4.5***	***MultiClass sSMO***	***MultiClass DTable***	***MultiClass 5-NN***	***MutliClass 10-NN***
Evaluation Metric														
Accuracy	99.9087	100	8.2231	8.1945	8.3627	7.9863	99.1814	100	100	99.9315	7.9863	100	8.3430	8.2574
% incorrect classifications	0.0913	0	91.7769	91.8055	91.6373	92.0137	0.8186	0	0	0.0685	92.0137	0	91.6570	91.7426

Mean absolute error	0.0002	0	0.1528	0.1528	0.1527	0.1530	0.0050	0.0067	0.1503	0.1503	0.1528	0.1503	0.1528	0.1528
Root mean squared error	0.0063	0	0.3021	0.2893	0.3908	0.2818	0.0319	0.0121	0.2719	0.2719	0.2764	0.2719	0.2764	0.2764
IR precision	1	1	0.0200	0.0136	0.0064	0.0799	0.9997	1	1	0.9993	0.0799	1	0.0160	0.0104
IR recall	1	1	0.2500	0.1700	0.0800	1	0.9993	1	1	1	1	1	0.2000	0.1300
F-measure	1	1	0.0370	0.0252	0.0118	0.1479	0.9995	1	1	0.9997	0.1479	1	0.0296	0.0192
Area under ROC	1	1	0.5	0.5	0.5	0.5	1	1	1	1	0.5	1	0.5	0.5
Elapsed time training	0.0457	0.0527	0.0003	0.0003	1.1452	0.0982	12.8017	0.2137	0.2766	0.1412	0.0555	1.5260	0.0092	0.0081
Elapsed time testing	0.0003	0.0031	0.1755	0.1805	0.0168	0.0008	0.0016	0.0006	0.0099	0.0013	0.0021	0.0037	2.1594	2.0779
KB mean information	3.5806	3.5839	0.4269	0.3237	0.1841	0.2715	3.5296	3.5248	0.2381	0.2380	0.0204	0.2372	0.0221	0.0215
Kappa statistic	0.9990	1	0	0	0	0	0.9911	1	1	0.9993	0	1	0	0

CPU – property5 – individual sessions considered as standalone for each user

European Commission

EUR 27054 EN - Joint Research Centre - Institute for Protection and Security of the Citizen

Title: **Utilizing CPU, Memory and other features signals to control processes and related data in computing devices with potential to identify user**

Author(s): D. Geneiatakis, A. Malatras, I. Vakalis

Luxembourg: Publications Office of the European Union

2015 – 45 pp. – 21.0 x 29.7 cm

EUR - Scientific and Technical Research series - ISSN 1831-9424 (online), ISSN 1018-5593 (print), 1831-9424 (CD-ROM)

ISBN 978-92-79-45027-3 (pdf)
ISBN 978-92-79-45025-9 (print)
ISBN 978-92-79-45026-6 (CD-ROM)

doi: 10.2788/259848

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 instead 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.

LB-NA-27054-EN-N

JRC Mission

As the Commission's in-house science service, the Joint Research Centre's mission is to provide EU policies with independent, evidence-based scientific and technical support throughout the whole policy cycle.

Working in close cooperation with policy Directorates-General, the JRC addresses key societal challenges while stimulating innovation through developing new methods, tools and standards, and sharing its know-how with the Member States, the scientific community and international partners.

Serving society
Stimulating innovation
Supporting legislation

doi:10.2788/259848

ISBN 978-92-79-45027-3