Integrating machine learning to the sub-classification of monitoring operations for anomaly detection based on smart automation technology data center.

Ahmed Yassine Chakor (Yassine.ckr@gmail.com)
Abdelmalek Essaâdi University
Abdellah Azmani
Abdelmalek Essaâdi University
Monir Azmani
Abdelmalek Essaâdi University

Case Report

Keywords: Monitoring, machine Learning, AI, supervised learning

Posted Date: April 14th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1516631/v1

Abstract

Intelligent monitoring of a computer network provides a clear understanding of its behavior at various times and in various situations it also provides relief to support teams who spend most of their time troubleshooting problems caused by hardware or software failures. This type of monitoring ensures the accuracy and efficiency of the network to meets the expectations of its users.

However, to ensure intelligent monitoring, it is necessary to start by automating this process which often leads to long and costly interventions. The success of such automations supposes the establishment of predictive maintenance as a prerequisite for good governance of preventive maintenance. However, even when it is practiced effectively, preventive maintenance requires a great deal of time and the mobilization of several full-time resources, especially for large IT structures.

This paper gives an overview on the monitoring of a computer network and explains its process and the problems encountered. This article also proposes a method based on machine learning to allow prediction and support decision making to proactively anticipate interventions.

1. Introduction

Monitoring a computer network involves collecting data to provide real-time statistics and analyze network performance. As well as checking the state of the system continuously and by several possible ways [1]. These resources are often complementary such: logs, audit trails, system performance data, etc. This is a difficult task, because it requires a lot of time and human effort, to monitor all the devices connected to a computer network associated with an information system at regular intervals and in real time [2].

Monitoring a computer system provides several key performance indicators that we call "performance counters". These refer to the resources of a system example of the activity of the CPU [3] the speed of reception of network frames by network interfaces [4]. and the availability of network equipment. In our paper, we will remind monitoring fundamentals, followed by the presentation of the best-known solutions on the market then we will present a state of the art on monitoring and a generic model of the monitoring process. We will then list the problems relating to the current monitoring process, then propose a model based on artificial intelligence which can be integrated into the five monitoring logical measurement functions illustrated by Sihyung. Lee et al [5], The proposed layers (learning and prediction) will allow the monitoring process to predict potential failures through scores that will be assigned to each device. The article ends with a discussion and a conclusion giving some perspectives on our research work.

2. Monitoring

Monitoring, also known as computer surveillance or supervision, is the process of controlling and verifying the activity of a computer system to obtain relevant information [6].

Many terms have been used in the literature to refer to the observation and evaluation of computer technologies such as: technological surveillance [7], AIRI (Associazione Italiana per la Ricerca Industriale [8] and EIRMA (European Industrial Research Management Association [9].), technological intelligence [10], [11], technological forecasting [12], [13]and technology assessment [14]. [15]

IT systems monitoring is one of the daily activities of an IT team which allow supervision, analysis, and management by acting directly after receiving alerts informing of potential failures. Monitoring meets the 4 following objectives [16]:

1. Help with the decision to achieve the expected results.
2. Document the supervision project to feed the learning process, communication and even advocacy processes.
3. Report to the actors concerned.
4. Contribute to strengthening the skills of the actors involved.

By collecting information from the network, monitoring help IT teams to gain a better understanding of the behavior of employees within the same network and to study problems, such as the misuse of resources or the overload of IT resources.

Many existing tools aims to evaluate the use of the system and to monitor performance measures, such as Munin [17], OpenNMS [18], Nagios [19], Cacti [20], Zabbix [21] and many others. These tools generally represent graphs of system performance measurements using Simple Network Monitoring Protocol (SNMP) [22].

To identify the network model, we need to follow a process of five layers that correspond to the functions of logical measures shown in Fig. 1 which are: collection, representation, reporting, analysis, and presentation.

Earlier works have focused on improving the efficiency of this monitoring process by handling the large amounts of data to be collected, analysed, and stored. [23], [24], [25] and even if we manage to overcome the problems of critical limits, induced by the actions of the monitoring process related to the processing power, the overload of the bandwidth and the storage capacity, it remains nevertheless a major challenge which is characterized by the identification of important events among vast quantities of measurement data, their visualizations, and their interpretation to make decisions and detect failures [26]. In this context, a large body of previous work on surveillance [27], [28], [29], [30], [31], [32], were particularly interested in the analysis layer. According to Sihyung [5] Lee et al high-level interpretations of derived measurement data are used to decide whether to alert the operator, record reported measurements or reconfigure the network.

Although much work addresses issues in more than one layer, this logical separation of different monitoring functions clarifies their functionality and interactions. Also, in our approach we will be interested in the "Collection" layer, to solve the management problems of the large quantities of recorded

and analysed data, then in the "Analysis" layer to allow IT teams to predict failures in their terminals instead of just displaying them in a graph. Our approach will be based on the principles of big data and on the use of Multi-Agent Systems.

In what follows, we will analyse all the surveillance operations based on the five levels of the surveillance process in Fig. 1 illustrated by Sihyung Lee et al [5], then we present a sub-classification, to put highlight, the different logical functions of surveillance. We will then analyse, in a detailed manner, the functions of each layer, then we will present two issues which combine all the monitoring operations.

2.1 Collection layer

This layer represents the collection of measurement data from a computer network. The collection can be done actively or passively [34]. Active monitoring can directly measure what users want to observe without waiting for a particular event to occur. However, active monitoring should minimize the impact on normal traffic and continued network operation. The size and frequency of the active poll are the two parameters that determine the impact of surveillance on the operation of a computer network [35].

Passive monitoring runs on dedicated devices where it is performed by network devices such as routers and switches. Passive surveillance is non-intrusive and affects network behavior less than active surveillance. In addition, it observes the actual behavior of the network.

Many works on active and passive monitoring raises several problems such as latency to obtain monitoring information, the rate of loss of data received through packets and storage capacity [36], [37], which must be very important, to not block the monitoring process and reduce the performance of the network. Other works talk about bottlenecks [38], [39], which limits the operation of the process in the event of oversized processing of the monitoring data obtained.

To solve these performance issues, several studies suggest the use of both active and passive surveillance [40], others suggest adding new functions to existing routers to make active monitoring easier and more accurate.

1. R. Kompella, et al [41], S. Machiraju [42] and P. Papageorge [43] suggest that routers should transmit measurement packets with modulable and configurable priority.
2. C. Estan, [43] and E.A. Hernandez [44] Provides sampling that reduces overhead on monitoring nodes and performs efficient monitoring. However, the unsampled information is lost.

As a result, most of the work on this layer focuses on improving the accuracy of the measurements collected. But do not deal with the problems that can be generated because of data filtering and their processing.

2.2 Presentation layer

The role of this layer is to give a unique form to the information sent by the equipment of the network. It ensures that the application layer information of the OSI model in transit is readable by the application layer of another system. The representation layer manages three aspects including the format of the data which must be Postscript, binary or ASCII, compatibility with the host's OS and the encapsulation of the data for transmission via the network

According to R. Kompella, [45] there are three important problems related to the "Representation" layer:

1. First, the representation of collected metrics should be normalized to ensure that each analysis function uses the same form and to prevent these metrics from being converted into separate intermediate representation forms. In this perspective, several works express the use of these standards through SNMP MIB, CIM, IPFIX: IP Flow Information Export (ipfix), and YANG monitoring tools.
2. The second problem consists in synchronizing in time the measurements coming from heterogeneous and distributed collection devices [46].
3. The last problem concerns the representation of measures which must be concise, to save storage space and network bandwidth [47].

This is achieved by identifying and sorting the duplicate measures and by combining values to generate derived values.

The data encoded in this layer is represented as strings of characters, integers, and floating-point numbers. It is handled differently by different machines depending on the encoding methods followed by them.

2.3 Report Layer

The reporting layer consists of producing a measurement report of the data taken through a computer network. It provides standard methods for signing flow records and verifying them [48]. It also specifies an encapsulation format that can be used for encryption as well as for digital signatures.

The reports sent are triggered either:

1. Periodically, with a predefined frequency, generally of the order of a few minutes. This periodic method decreases the number of measurements transferred, as well as the frequency of data request.
2. Only when a predefined event occurs.

Periodic polling is less efficient, but it allows the identification of new undefined events. On the other hand, polling triggered by an event is more efficient, and the events to be reported must be carefully defined, because undefined events go unnoticed. These two methods present a trade-off between efficiency and precision of monitoring. This reporting must be efficient in terms of bandwidth consumed during the transfer of measurement data to the supervision equipment.

Several recommendations are found in the literature to improve the efficiency of reports, by reducing the number of measurements:

1. Reporting and elimination of redundant measures [49].
2. Aggregation of measures with similar characteristics [50].
3. Consolidation of the same interrogation requests from different applications into a single request [51].
4. Use of bandwidth-saving measurement encodings: IPFIX [52] defines such encodings.

Despite the various research carried out at this layer, reporting is a problem when the data is very large. Our approach will make it possible to refine the report phase to allow more efficiency and precision at the same time.

2.4 Analyse Layer

This layer is the most important in our research, it consists in analyzing the data collected by the preceding layers, because each network problem has its own characteristics and it is impacted by many factors such as CPU load, hard disk temperature, ping, RAM memory occupation and response time.

In this layer, we also find the characteristics appropriate to network problems and the intervals of data measurements, which makes it possible to fully unleash the potential of the analyzed data.

Operations classified in this layer extract high-level interpretations of network condition by analyzing collected metric data. In general, the measurements collected give a result:

- Yes or no type boolean to indicate whether a component is available or not.
- Digital, eg response time.
- Qualitative, during the execution of a query, for example the state of the network at a specific time (Good, medium, or poor).

Although there are a variety of analysis functions identified in [53], [54] and [55], here we present the six most common ones:

1. General purpose traffic analysis:
2. Estimate of traffic demand.
3. Classification of traffic by application.
4. Extraction of communication models.
5. Management of faults.
6. Automatic update of network documentation.

The results of these analysis functions are widely used in the design of configuration changes.In our approach we will focus on the application of machine learning on fault management, several research have been done in this direction and which we will cite in section 5.

2.5 Presentation Layer

The presentation layer is responsible for coding the application data. It is easier to monitor a network through visual representations, rather than digital data. The Multi-Router Traffic Grapher (MRTG) [56] and the SNMP Network Analysis and Presentation Package (SNAPP) [57] provide visual representations of SNMP data, such as: bandwidth usage, physical state of 'a machine (CPU load, RAM, etc.), application availability or known attacks on a firewall.

However, visualizing usually large metric data proves to be a challenge. Because a good visualization tool must be able to deduce the importance and relevance of information and must also visually represent information in an intuitive way. So, it will become even more important to summarize and present the important events among vast amounts of measurement data, as internet usage is increasing rapidly more and more data is being collected especially with the emergence of new intelligent sensors and the use of IoT. To overcome this definition, for its part, SNAPP [58] identifies the correlated events and presents these events in the form of groups, so that the operators can have an efficient judgment of the relevance of the alerts via a simpler presentation.

3. Issues Across Multiple Layers Of The Monitoring Operations

Based on the above, in this section we highlight two major issues related to the collection, analysis and representation layers:

- Simultaneous improvement in the efficiency, accuracy, and flexibility of monitoring.
- Storage, analysis, and presentation of large amounts of measurement data.

We also mention an emerging problem, which is the monitoring of smart sensors in a network.

3.1 Improving efficiency, accuracy, and flexibility

Current monitoring systems allow IT Teams to monitor their network in real time. The analyzes that these systems present allow preventive and corrective maintenance of the network. But when the latter is very large, maintenance becomes a difficult task, because it requires a lot of time and human effort due to very large numbers of alerts related to the status and warning of the system.

Currently, the operational team contains three groups of people, which we will classify according to three levels of intervention:

1. Operations who handle customer calls and solve a subset of easily located problems, in this case we talk about first level interventions.
2. Network engineers solve problems that arise from machines and services under their supervision, these are second level interventions.

3. Network designers deal with issues that previous groups could not handle as these issues require a deep understanding of network configurations and may also involve interactions between multiple devices, this group solves third level issues.

As we move from group to group, the time it takes to locate their root causes increases and the number of unresolved issues and the effectiveness of their resolution decreases.

Even when they are analyzing the visual representation of information generated at runtime, it is difficult to establish strong informative correlations between attributes, to derive valuable actionable insights from the dataset.

We seek through our work to improve system efficiency through intelligent algorithms capable of learning network behaviors and making real-time decisions without human intervention. In fact, algorithms and computers make decisions and execute transactions faster and more efficiently than any human, while human decisions can be biased and impacted by their emotions.

3.2 Managing large amounts of data

Between the Presentation layer and the Analysis layer, multiple data are continuously stored to enable analysis of the network condition, causing the size of data collected to grow rapidly. In addition to real-time monitoring, operators must keep a history of incidents and logs to allow easy diagnosis if the problem reoccurs.

The most common way to reduce the amount of data stored and analyzed is consolidating it [59]. This consists of gradually reducing the volume of data from past measurements recorded, by consolidating this data using different functions such as the average, the maximum, or the total. Consolidation also makes it possible to reject data relating to measurements collected prior to a periodic threshold. This is data considered obsolete that is kept for a period generally a week and sometimes a month. However, this consolidation approach by removing (obsolete) data to anticipate possible situations is not an effective solution, because the number of data is reduced.

However, in our approach we will rely on all the data collected by the sensors to allow intelligent algorithms to learn to predict failures and rely on big data concepts and technologies to face the problems of the very large amount of data accumulated during the history of a network.

4. Related Works

Several works have evoked machine learning in the monitoring process. B. Nguyen, et al [60] present fault detection based on the use of machine learning in mobile networks. This approach is based on the use of user data by geographic region, device, and service operator, which requires additional deployment of service monitoring in addition to network monitoring. S.Zidi. T. Moulahi. and B. Araya [61] use the “support vector machine” method to classify received smart sensor data to detect faults through normal data behavior. This approach requires that traffic be redirected to the server where the classifier is

deployed, resulting in long data processing delay and additional communication overhead. V. C.Ferreira. et al [62] presents work based on system logs as input, for the detection and diagnosis of solar powered mesh network failures. The authors used existing data in the database methodology as well as a predefined list of failures, based on their previous experience. J. M. Navarro [63] described an online failure prediction system built on Apache Spark, which takes a repository of network management events and forms a random forest model, this approach uses this model to predict the occurrence of future events in real time. However, the failure prediction system described by the authors has been shown to be reliable only in a network with a very small number of devices. Our approach will be based on scores which are generally only a few KB, which is much smaller than the data traffic. In addition, the two cited works focus on wireless / mobile networks, our work will try to bring together wired and wireless.

5. Machine Learning Applied To Monitoring Process

Machine learning algorithms have been widely used in various fields of research and have shown good performance in learning and recognizing patterns. In proactive fault detection, we assume that before a fault occurs, there are system parameters that show signs of weakness which may cause outages in the future. When this data is collected and analyzed by the learning algorithms, certain characteristics of the system (in good and in bad condition) can be modeled during the learning phase.

5.1 Machine learning steps into monitoring process

Figure. 2 illustrates the general process of supervised learning and which we explain, in the context of monitoring, step by step below.

Machine learning is not just a set of algorithms, but follows a succession of steps which we summarize as follows:

5.1.1 Collection data

This step is very important to be able to apply machine learning. Regarding monitoring, it is necessary to collect, from the maintained network, relevant data, in large quantity and good quality, while avoiding any possible bias in their representativeness.

Example of network data collected as the CPU load which indicates the intensity with which the processor processes the programs and the running processes. This value indicates the percentage of the total time used by the processor or the processor core for processing data. The maximum CPU load is 100%. This data is therefore an indicator that allows to know the degree of use of the processor as well as the capacities still available.

5.1.2 Preparing and cleaning the data

The data collected must be filtered, or even retouched, before being used. This is because some attributes are unnecessary, and others must be modified to be suitable for use by an algorithm. It also happens that

some items are unusable because their data is incomplete. Several techniques such as data visualization, data transformation or even standardization are then used. For example, router status settings should contain two attributes x and y, and when receiving information, the router sends only x. In this case, either it is necessary to eliminate the attribute or to complete it to make the automatic learning data more reliable.

5.1.3 Choosing and building a learning model

A wide choice of algorithms exists, and it is necessary to choose one which is adapted to the problems and data. In the following, we present an approach based on the decision tree, which allows supervised learning, whose algorithms automatically select the discriminant variables from unstructured and potentially large data. They can thus make it possible to extract logical cause-and-effect rules (of a deterministic nature) that did not initially appear in the raw data.

5.1.4 Train, evaluate and optimize

The machine learning algorithm is trained and validated on a first set of data to optimize its hyper parameters. To push further the learning process, the algorithm will evaluate its predictions based on a second set of data, called test, to verify the relevance of the predictions. This action makes it possible to evaluate its effectiveness on a set of data independent of those used in the training phase. Its objective is to verify whether this algorithm manages to predict the situations reflected by the test data set and to prevent it from over-learning.

5.1.5 Deploy

The model is then deployed in production to make predictions, and it can potentially use new input data on a regular basis to re-train and be improved.

6. Proposed Layer

To draw correlations from several attributes in a high dimensional space, and to answer the current monitoring problems treat above. We cannot simply depend on the use of graphs and visualizations of network monitoring data.

As a reminder, the process relating to the monitoring operations mentioned in [5], is characterized by the following layers:

- Layer collection
- Representation layer
- Report layer
- Analysis layer
- Presentation layer

These make it possible to identify or encounter a certain number of problems which we list as follows:

- Simultaneously improving the efficiency, accuracy, and flexibility of monitoring.
- Storage, analysis, and presentation of large amounts of measurement data.

To improve the monitoring process in FIG. 3, it is imperative to propose an approach which resolves the problems. For this, we propose the addition of two layers at the level of the sub-classification of the monitoring process (Learning layer and prediction layer).

Figure 3. Sub classification of monitoring operations including learning and prediction, integration of the machine learning system on the monitoring layer.

In Figure. 3 we have merged the layers of monitoring with machine learning to have an intelligent monitoring system that can predict potential problems in an information system. The two learning and prediction layers will integrate after the Data Analysis phase to allow supervised learning algorithms to perform the following two blocks of operations:

6.1 Observation and learning

Each decision is based on a question related to one of the incoming variables (parameters sent by the monitored equipment). Based on the responses, the data instance is then categorized. This succession of questions and answers and the resulting divisions create a tree-like structure.

The algorithm will collect the analyzed learning datasets which correspond to a set of input-output pairs:

$(x_n, y_n)_{1 \leq n \leq N}$ (1) with $x_n \in X$(2) and $_{(x_n, y_n)_{1 \leq n \leq N}}$(3)

This retouching is necessary because some attributes are unnecessary, and others are incomplete and therefore unusable. To help control this complexity when collecting input data, we opt to suppress some question, then create a cluster of small, simple trees rather than a single tree with many branches. This variant of classification and regression trees characterizes a random forest. Each of these small trees evaluates a portion of the data, and then the results are put together to create a final prediction.

6.2 Prediction

This is phase two of supervised learning, which involves predicting the labels of new data, knowing the previously learned pattern.

This structure will contain algorithms and data structures. It will only have two components:

- Branch nodes, which represent a single input variable and offer a single split point on the variable.
- The leaf nodes, which represent the two output variables.

When the algorithm is executed by the machine, failure prediction is made by following the divisions of the branch node until it reaches a leaf node. This leaf node is the prediction or the output of the class

value. As shown in Figure. 5, within the process of learning and predicting, four subsystems will be needed: score collection, learning, prediction, and scoring.

7. Alrogithm

7.1 Used technology

In our paper, we used supervised learning techniques. For this, we will use the input data we have (CPU load, memory state, network state, processor temperature, etc.), then we will determine the expected results per failure (memory blocking, hard disk full, network unavailable, etc.). This learning will allow us to make predictions based on a model that is obtained from failure history data and the chosen algorithm.

Our choice will fall on a monitoring system developed under Python, because of its adaptation and ease of use to develop machine learning algorithms. Most libraries used for data science or machine learning have Python interfaces. As a result, this language has become the most popular high-level command interface for libraries of machine learning and other digital algorithms.

Python is erected as the best programming language for Big Data, thanks to its different data science packages and libraries, like Pandas, Agate, Bokeh, NumPy, Scikit-lear, TensorFlos, ect.

7.2 Data and application

The data collected in this project was generated in real time by the servers in the Smart Automation Technologies data center through a structured data collection script running in parallel (via SSH) on the 5 servers chosen with almost identical performance. Our data was collected for 1 year and was available in 300 CSV files, each containing one day's data. We export the data from the monitoring application in CSV format, we obtain the following results from several monitored equipment:

Table 1
Extract of the monitoring data imported from monitoring system

Machine Name	Internal Temperature	Memory Usage	License	At risk
Server HP prolian 220 G2 fixe RH	31 C	90%	Yes	0
Server HP prolian 220 G2 fixe Ref 32456T1	70 C	100%	Yes	1
LENOVO Server intel pentium C20 L7	70 C	4%	Yes	0
Server HP Prolian NTM	27 C	4%	No	

The measured data are: Internal temperature, use of RAM memory and license, Disk usage... We choosed these parameters based on the constructors guide lines for the most critical parameters impacting the

controled devices. Our goal is to allow the monitoring system to predict whether the condition of the machine is critical:

- Good: no intervention, referenced as 0.
- Alert: preventive maintenance planning, referenced as 1.
- Critical: corrective maintenance, referenced as 2

It seems that some variables are highly correlated with each other, this might have influence during modelling, and we might need to drop/merge some variable data.

Other columns should be deleted as they are all blank. This could indicate that values are not changing, let's take a closer look at the correlation

The above characteristics are perfectly correlated with each other, which means that we can remove some columns for the analysis.

Now let's look at the distribution of the different columns of data.

From the distribution plot above, some features do not change and can be removed from the analysis as well as discrete variables.

7.3 Pretreatment

In this section, we prepare data for modeling. There are four main steps we follow:

Divide the data into a training set, a validation set (to help us develop our models), and a test set (to help us evaluate the final version of each model); Resample the training set so that all classes are representative and nearly equally. Scaling the data, which will help ensure that the PCA and some machine learning algorithms work well. And Principal Component Analysis (PCA), which will reduce the dimensions of the data and eliminate any multicollinearity. We first do all these steps on the training data so that we can check the result at each step. Once done, we put the essential pre-processing steps into a pipeline that we can use to transform the validation and test sets.

7.4 Segregation train-validation-test

First, we take 60% of the total dataset to use as training data. Then we take the remaining 40% of the original dataset and dedicate half of it to validation and the other half to be used as a true retention set, which we'll only use to evaluate releases finals of each of our models. We also do stratified splits so that each subset contains all classes in the same proportions as in the original dataset.

7.5 Resample training data

This pre-processing step consists of dealing with the class imbalance. Our strategy is to oversample the three classes using two methods (OverSampling and SMOTE) so that each class has a representative

size and roughly the same size. First, we put the training data back into a DataFrame, then we review the current counts of all the classes.

Table 2
Oversampling result

Method	Normal	Alert	Critical
OverSampling	30000	2838	160
SMORE	30000	22014	19336

7.6 Scaling

Our next transformation is to standardize (or standardize) functionality. It's required for complex algorithms like random rorest.

The features have now been standardized so that they all have a mean of 0 and a standard deviation of 1. Let's get a quick visual confirmation of this change with a few distros.

Now that the training data has been scaled, we can move on to PCA. Principal component analysis takes the features we have and tries to combine them in a way that maximizes the amount of variance that the components can explain in the data. For datasets where there are many features, PCA can help us avoid the negative effects of having many features while allowing us to account for as much variance as possible. For our analysis below, we aim to find a number of components that will explain at least 90% of the variance in the data.

Based on these 11 features, the first 5 principal components seem to capture over 95% of the variance in the data, which is what we expected based on the feature importance plot above. Using fewer features/components will help us avoid overfitting the training data, but at the cost of some accuracy.

7.7 Preprocessing pipeline

We transform our data so that it is standardized and has 5 components. We use a pipeline to do this so that we can easily transform our validation and test sets in the same way as the training set without data leakage from one set to another.

We fit the pre-processing pipeline to the training data (which has been resampled) and then use the pipeline to separately transform the validation and test sets.

8. Result

According to the confusion matrix, our model based on the random drill correctly predicted 805 times that the machine is faulty and 31 times that it is a warning while it failed 110 times by saying that the machine is in good condition while she is failing.

For all the data on which we did the prediction test, our intelligent algorithm allowed us to predict the condition of the equipment, based on the training data with an accuracy of 93.66. We can see that the best hyperparameter values are not the default ones. Each dataset will have different characteristics, and the model that works best on one dataset will not necessarily be the best for all datasets.

9. Discussion

With the right implementation, the proposed solution will allow IT teams to spend less time and effort on daily and less important alerts. The platform will learn through algorithms and machine learning, to update the knowledge acquired over time and improve the behavior and efficiency of the monitoring software. These tools will also ensure continuous monitoring and will allow IT teams to focus on more serious and complex issues, as well as improve business performance and stability.

Our approach will be able to observe cause and effect relationships between different systems, services, and resources, aggregating and correlating disparate data sources. These analysis and machine learning capabilities will enable in-depth root cause analysis, making it faster to identify and resolve difficult and unusual problems. Our approach will improve collaboration and workflows between different IT teams, and between IT teams and other operational divisions. Equipped with personalized reports and dashboards, teams quickly understand their tasks and obligations, and communicate with each other without having to learn everything the other team needs to know.

Although this technology will reach maturity, there is still a long way to go to combine the results and put them into practice. The quality of our results depends on the data received and the algorithms learned, and the storage, protection and retention of data play a vital role.

10. Conclusion

The integration of machine learning into the IT monitoring process is a big step towards optimizing the time of IT teams and reducing downtime, which is often very expensive. However, machine learning requires large amounts of data to function properly. So, it can be difficult to control the integrity of data sets, especially in the case of data generated by real-time monitoring systems.

The quality of decisions made by a Machine Learning algorithm depends not only on the quality, that is, the consistency and reliability of the data used for training, but above all on their quantity. Indeed, the ability to make good decisions during Machine Learning depends on the size of the data.

In our article, we have presented network monitoring issues and guidelines for future surveillance and monitoring systems. We have identified two major issues: (1) simultaneously improving the efficiency, accuracy, and flexibility of monitoring, and (2) storing, analyzing, and presenting the vast amounts of data. of measurement. Next, we created a model that uses the decision tree algorithm to predict whether a failure will occur with an accuracy of 93.66. This model will therefore integrate the learning process that we have proposed (figure. 4); This model could be used by IT teams to decide whether preventive or

corrective maintenance is needed. In our next research, we will combine our prediction based on the collected data with real-time prediction, using Markov chain to obtain more precise fault results and build a more reliable system. Then come up with an intelligent algorithm that allows potential solutions to be assigned according to the degree of failure to automate problem solving.

Declarations

Author Contributions Statement

Ahmed Yassine Chakor, gathered all the data and applied it using classification algorithm for machine learning, created all the figures and wrote the submitted article.

All authors reviewed and validated the manuscript.

Funding

There is no funder(s) of the research described in this manuscript.

References

1. Salfner, F., Malek, M.: 2010. “Architecting Dependable Systems with Proactive Fault Management.” Architecting Dependable Systems VII, 171–200..Papalambrou, A., Voyiatzis, A. G., Serpanos, D. N., & Soufrilas, P. (2011). Monitoring of a DTN2 network. 2011 Baltic Congress on Future Internet and Communications. doi:10.1109/bcficriga.2011.5733226
2. Salfner, F., Malek, M.: 2007. “Using Hidden Semi-Markov Models for Effective Online Failure Prediction.” 26th Int’l Symposium on Reliable Distributed Systems (SRDS 2007).https://www.opennms.org/en/opennms
3. Eeckhout, L., Sundareswara, R., Yi, J.J., Lilja, D.J., and Paul Schrater. 2005. “Accurate Statistical Approaches for Generating Representative Workload Compositions.” In In Proceedings of the 2005 IEEE International Symposium on Workload Characterization, 55–66.
4. Magalhaes, J.P., Silva, L.M.: 2012. “Anomaly Detection Techniques for Web- Based Applications: An Experimental Study.” In 2012 11th IEEE International Symposium on Network Computing and Applications (NCA), 181–90. doi:10.1109/NCA.2012.27
5. Sihyung, L.,⇑, Hyong, K.L.: S. Kim c Network monitoring: Present and future http://dx.doi.org/10.1016/j.comnet.2014.03.007 1389 – 1286/_ 2014 Elsevier B.V. All rights reserved
6. Fu, S.: 2009. “Failure-Aware Construction and Reconfiguration of Distributed Virtual Machines for High Availability Computing.” In, 372–79. IEEE. doi:10.1109/CCGRID.2009.21
7. Hoffmann, G.A., and M. Malek. 2006. “Call Availability Prediction in a Telecommunication System: A Data Driven Empirical Approach.” In 25th Symposium on Reliable Distributed Systems
8. Porter, A.L., Rossini, F., Mason, T.W., Banks, J., Roper, T.: 1991. Forecasting and Management of Technology. Wiley, USA

9. AIRI (Associazione Italiana per la Ricerca Industriale), 2002. Il monitoraggio tecnologico, Edizioni AIRI
10. EIRMA (European Industrial Research Management Association), 1999. Working group 55. Technology monitoring for business success, Edizioni EIRMA
11. Ashton, W.B., Bryan, A., Richardson, A., et al.: Keeping Abreast of Science and Technology: Technical Intelligence for Business. Battelle Press, Columbus; (1997)
12. Christensen, C.M., 1997. The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail. Harvard business school, Boston, M.A., Iansiti, M. 2000. How the incumbent can win: managing technological transitions in the semiconductor industry. Management Science 46 (2), 169–185
13. Lichtenthaler, E.: Technological change and the technology intelligence process: a case study. J. Eng. Tech. Manage. **21**, 331–348 (2004a)
14. Twiss, B.: Managing Technological Innovation. Pitman Publishing, London (1993)
15. Lemos, A., Porto, A.: Technology forecasting techniques and competitive technology intelligence: tools for improving the innovation process. Industrial Manage. Data Syst. **98**(7), 330–337 (1998)
16. Vanston, J.H.: 2003. Better forecast, better plan, better results. Research Technology Management 47–58 gen-feb
17. Loveridge, D.: Special publication on technology assessment. Int. J. Technol. Manage. **11**, 5–6 (1996)
18. Garud, R., Ahlstrom, D.: Technology assessment: a socio cognitive perspective. J. Eng. Tech. Manage. **14**(1), 25–48 (1997)
19. Papalambrou, A., Voyiatzis, A.G., Serpanos, D.N., Soufrilas, P.: (2011). Monitoring of a DTN2 network. 2011 Baltic Congress on Future Internet and Communications. doi:10.1109/bcficriga.2011.5733226
20. opennms.org. (2017). opennms. (OpenNMS Group) Retrieved December 16, 2021, from : https://www.opennms.org/en/opennms
21. Barth, W.: Nagios: System and Network Monitoring. No Starch Press, San Francisco (2006)
22. The Cacti Group, "Cacti Monitoring Tool," https://www.cacti.net/, 2018, online; Accessed: 24-Sept-2021
23. Zabbix, L.L.C.: "Zabbix monitoring system," https://www.zabbix.com/, 2018, online; Accessed 24-Sept-2021
24. Case, J., Fedor, M., Schoffstall, M., Davin, J.: "Simple network management protocol (snmp)," Internet Requests for Comments, Internet Engineering Task Force, RFC 1157, May 1990. [Online]. Available: http://www.ietf.org/rfc/rfc1157.txt? number = 1157
25. Kind, X., Dimitropoulos, S., Denazis, B., Claise: Advanced network monitoring brings life to the awareness plane. IEEE Commun. Mag **46**(10), 140–146 (2008)
26. Kumar, S., et al., "Algorithms to Accelerate Multiple Reg-ular Expressions Matching for Deep Packet Inspection,"Proc. SIGCOMM '06 Conf. Apps., Technologies, Archi-tectures, and Protocols for Comp. Commun., Pisa, Italy,Sept. 2006, pp. 339–50

27. Dimitropoulos, X., Hurley, P., Kind, A.: Probabilistic LossyCounting: An Efficient Algorithm for Finding Heavy Hit-ters. SIGCOMM Comp. Commun. Rev. **38**(1,Jan), 7–16 (2008) “,”,
28. Zemouri, R., and Noureddine Zerhouni: Autonomous and Adaptive Procedure for Cumulative Failure Prediction. Neural Comput. Appl. **21**(2), 319–331 (2011). doi:10.1007/s00521-011-0585-7. “.”) .
29. Watanabe, Y., and Yasuhide Matsumoto: Online Failure Prediction in Cloud Datacenters. FUJITSU Sci. Tech. J. **50**(1), 66–71 (2014) “.”)
30. Murray, J.F., Hughes, G.F., Kreutz-Delgado, K.: 2003. “Hard Drive Failure Prediction Using Non-Parametric Statistical Methods.” In Proceedings of ICANN/ICONIP
31. Liang, Y., Zhang, Y., Jette, M., Sivasubramaniam, A., Sahoo, R.: 2006. “BlueGene/L Failure Analysis and Prediction Models.” In Dependable Systems and Networks, 2006. DSN 2006. International Conference on, 425–34. doi:10.1109/DSN.2006.18
32. Watanabe, Y., Otsuka, H., Sonoda, M., Kikuchi, S., Matsumoto, Y.: 2012. “Online Failure Prediction in Cloud Datacenters by Real-Time Message Pattern Learning.” In 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), 504–11. doi:10.1109/CloudCom.2012.6427566
33. Pitakrat, T., Van Hoorn, A., Grunske, L.: 2014. “Increasing Dependability of Component-Based Software Systems by Online Failure Prediction (Short Paper).” In Dependable Computing Conference (EDCC), 2014 Tenth European, 66– 69. doi:10.1109/EDCC.2014.28
34. Zseby, T., Hirsch, T., Claise, B.: (n.d.). Packet Sampling for Flow Accounting: Challenges and Limitations. Passive and Active Network Measurement, 61–71. doi:10.1007/978-3-540-79232-1_7
35. Anagnostakis, K.G., Greenwald, M., R.S. Ryger, cing: measuring network-internal delays using only existing infrastructure, in:Proc. IEEE INFOCOM, March: 2003, pp. 2112–2121., http://allendowney.com/research/clink/
36. Hu, N., Li, L., Mao, Z.M., Steenkiste, P., Wang, J., Locating internet bottlenecks: algorithms, measurements, and implications, in: Proc.ACM SIGCOMM, October: 2004, pp. 41–54
37. Floering, B., Brothers, Z., Kalbarczyk, R.K., Iyer, An adaptive architecture for monitoring and failure analysis of high-speed networks, in: Proc. IEEE/IFIP DSN, 2002
38. dos Santos, G.L., Guimaraes, V.T., Silveira, J.G., Vieira, A.T., de Oliveira Neto, J.A., da Filho, R.I.T., Balbinot, R., UAMA: a unified architecture for active measurements in IP networks; End-to-end objective quality indicators, in: Proc. IEEE/IFIP IM, 2007
39. Kompella, K., Levchenko, A.C., Snoeren, G., Varghese, Every microsecond counts: tracking fine-grain latencies with a lossy difference aggregator, in: Proc. ACM SIGCOMM, August: 2009
40. Luckie, M.J., McGregor, A.J., Braun, H., Towards improving packet probing techniques, in: Proc. ACM SIGCOMM IMW, 2001
41. Papageorge, P., McCann, J., Hicks, M.: Passive aggressive measurement with MGRP. ACM SIGCOMM Comput. Commun. Rev. **39**(4), 279–290 (2009)
42. Estan, K., Keys, D., Moore, G., Varghese, Building a better NetFlow, in: Proc. ACM SIGCOMM, 2004

43. Hernandez, E.A., Chidester, M.C., George, A.D.: “Adaptive sampling for network management”, J. Netw. Syst. Manage. 9 (4) (2001)
44. Machiraju, S., Veitch, D., A measurement-friendly network (MFN) architecture, in: Proc. ACM SIGCOMM Workshop on INM, 2006
45. Kompella, R., Levchenko, K., Snoeren, A.C., Varghese, G., Every microsecond counts: tracking fine-grain latencies with a lossy difference aggregator, in: Proc. ACM SIGCOMM, August: 2009
46. Kompella, R., Levchenko, K., Snoeren, A.C., Varghese, G., Every microsecond counts: tracking fine-grain latencies with a lossy difference aggregator, in: Proc. ACM SIGCOMM, August: 2009
47. Simple, A. Network Management Protocol (SNMP), RFC-1157, May: 1990.
48. CIM Standards, <http://www.dmtf.org/standards/cim/> (accessed 12.08.13).
49. IP Flow Information Export: (ipfix), <http://datatracker.ietf.org/wg/ipfix/charter/> (accessed 12.08.13)
50. Network Configuration Protocol (NETCONF), RFC-6241, June 2021
51. YANG – A Data: Modeling Language for the Network Configuration Protocol (NETCONF), RFC-6020, October 2010
52. Fraleigh, S., Moon, B., Lyles, C., Cotton, M., Khan, D., Moll, R., Rockell, T., Seely, S.C., Diot: Packet-level traffic measurements from the Sprint IP backbone. IEEE Netw. Mag **17**(6), 6–16 (2003)
53. IP Flow Information Export: (ipfix), <http://datatracker.ietf.org/wg/ipfix/charter/> (accessed 12.08.13).
54. Lin, Y., Chan, M.C.: A scalable monitoring approach based on aggregation and refinement. IEEE J. Sel. Areas Commun. **20**(4), 677–690 (2002)
55. Cheikhrouhou, M., Labetoulle, J., An efficient polling layer for SNMP,in: Proc. IEEE/IFIP NOMS, 2000
56. Network monitoring: Present and future Sihyung Lee a,⇑, Kyriaki Levanti b, Hyong S. Kim c http://dx.doi.org/10.1016/j.comnet.2014.03.007 1389 – 1286/_ 2014 Elsevier B.V. All rights reserved
57. Trammell, B., Boschi, E., Mark, L., Zseby, T., Wagner, A., Specification ofthe IP Flow Information Export (IPFIX) File Format, RFC-5655, October. 2009.
58. Trammell, B., Gates, C., NAF: the NetSA aggregated flow tool suite, in: Proc. USENIX LISA, 2006
59. Nguyen, B., Ge, Z., van der Merwe, J.E., Yan, H., Yates, J.: 2015. ABSENCE: Usagebased Failure Detection in Mobile Networks. In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, MobiCom’15. 464– 476
60. Plonka, FlowScan: a network traffic flow reporting and visualization tool, in: Proc. USENIX LISA, 2000.
61. Oetiker, T., MRTG: the Multi Router Traffic Grapher, in: Proc. USENIX LISA, 1998
62. idi, S.Z.. Moulahi, T.: and B. Araya," Faull Derecnon in Wirele" Senso r Networks Through SYM Cla"ilier:' IEEE Sensors Journal. vol.1 8. no. I. pp, 3-W-3-1.7, Jan. 2018

Figures

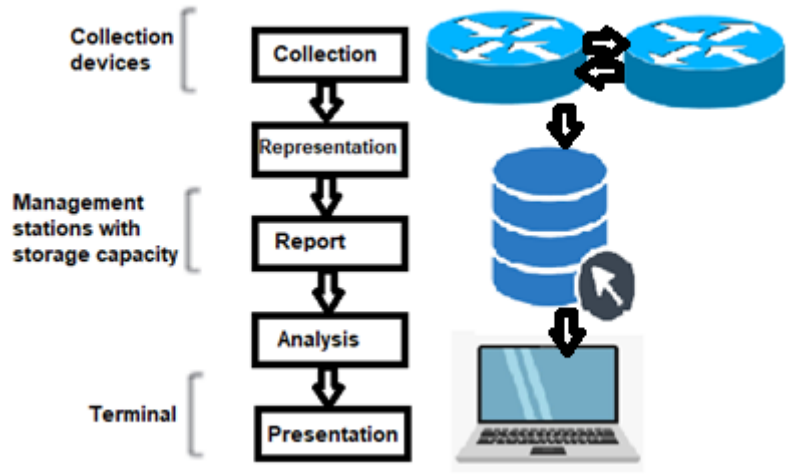

Figure 1

Sub-classification of monitoring operations

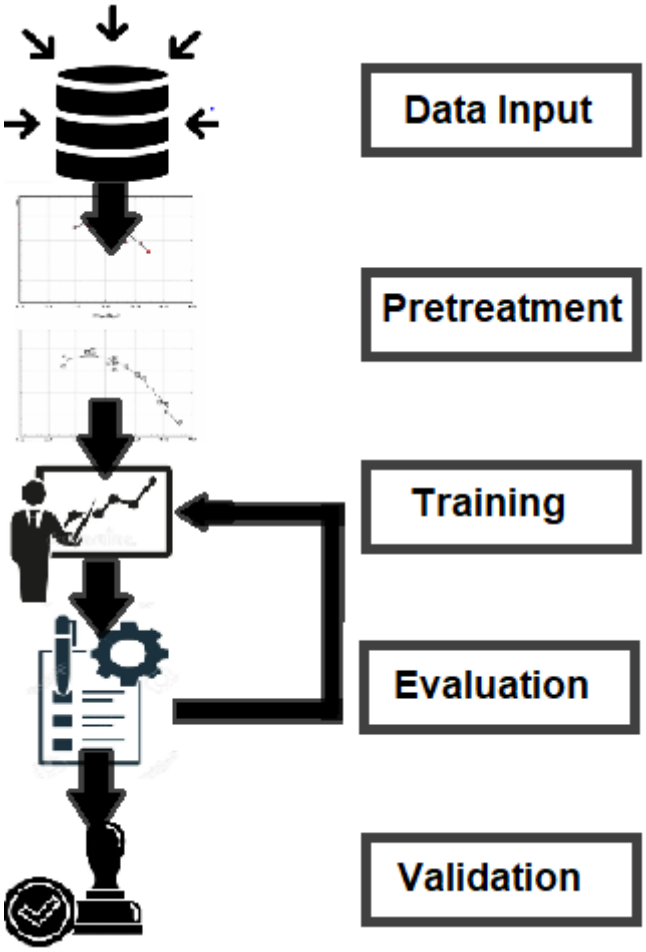

Figure 2

Learning layer using machine learning process

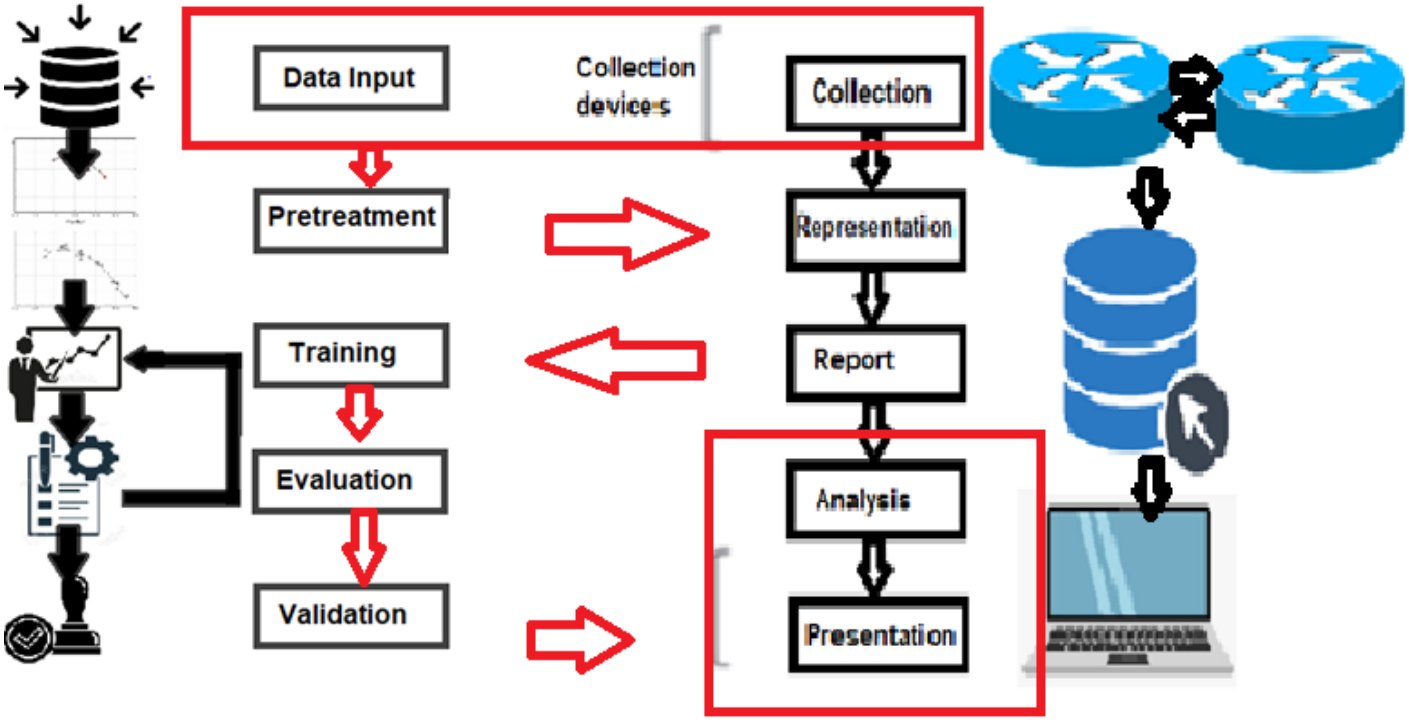

Figure 3

Sub classification of monitoring operations including learning and prediction, integration of the machine learning system on the monitoring layer.

```
# Import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import itertools
import pickle
from sklearn.tree import export_graphviz
from subprocess import call
from IPython.display import Image

from collections import Counter
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import (accuracy_score, confusion_matrix,
                             classification_report, roc_curve, auc)
```

Figure 4

We Import the necessary packages from python, for that we will use Anaconda Jupyter notebook

```
# Import method  cont'd: read in the data
df = pd.read_csv("dataset_.csv")
df.head()
```

Figure 5

We import the CSV file containing the monitoring data.

	time(s)	TOTAT_RAM_GB	Free_Size_RAM_GB	RAM_USED_GB	CPULoadPercentage	Criticité	CPUTemperature_C	HDDsize
0	0	3.889988	1.640015	2.249973	0	0	39	465.15
1	41	3.889988	1.358124	2.531864	57	0	44	465.15
2	76	3.889988	1.222076	2.667912	33	0	38	465.15
3	111	3.889988	1.216736	2.673252	0	0	40	465.15
4	145	3.889988	1.172710	2.717278	34	0	38	465.15

Figure 6

Then we will display our data on python notebook

Figure 7

verifying correlations

```
corr_pairs=find_corr_pairs(corr,0.9)
for c in corr_pairs:
    print(c)
```

```
('HDDfree', 'HDD_USED_GB', -1.0)
('Smart_9', 'Smart_193', 0.9205008743336093)
```

Figure 8

Finding High Correlation Pairs:

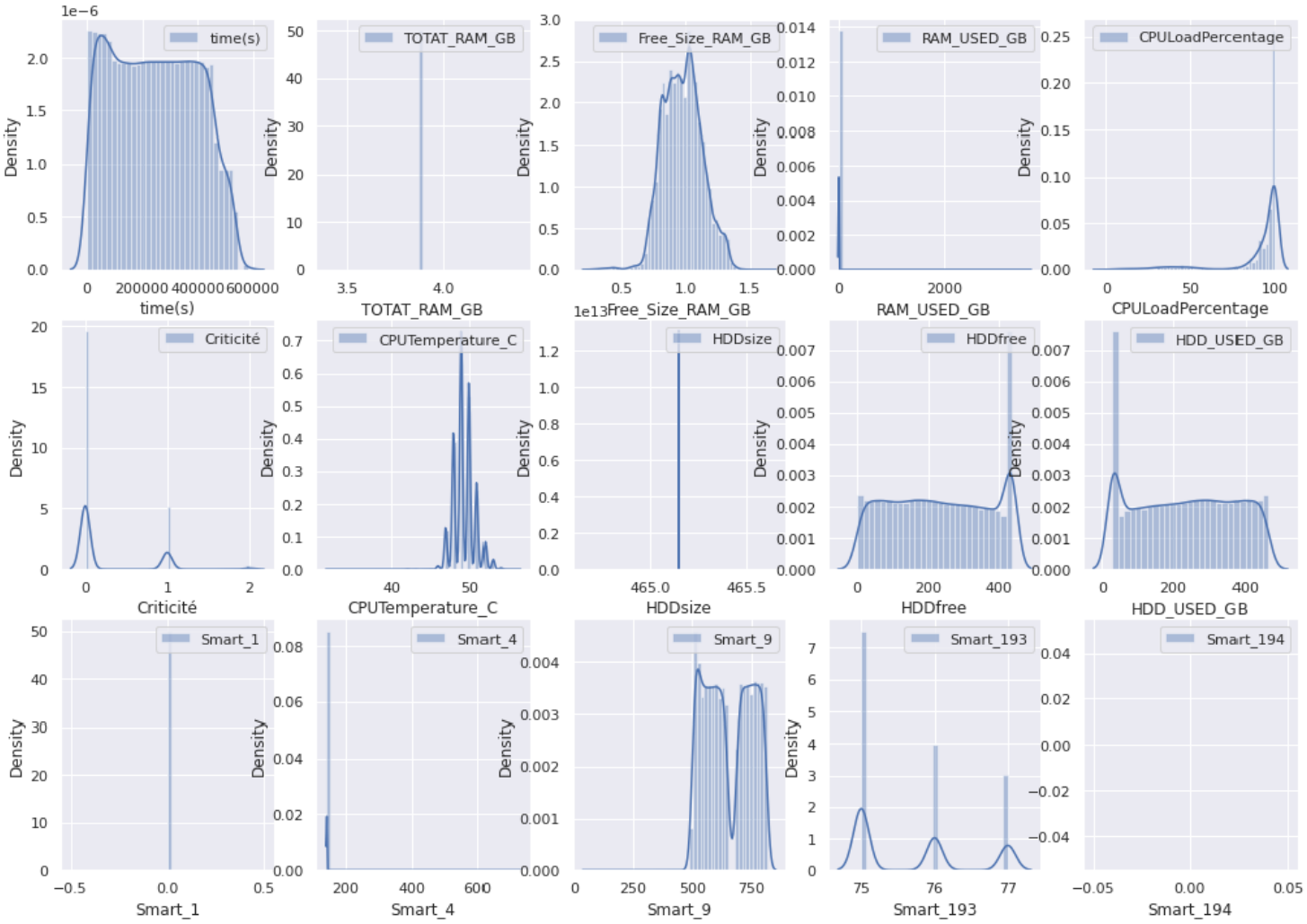

Figure 9

Column distribution

```
# Division en ensembles de données d'entraînement et de test
y = df['Criticité']
X = df.drop('Criticité', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state=1,
                                                    stratify=y)
```

Figure 10

Division into training and test datasets

```
# Sur-échantillonner la plus grande classe
rus = RandomOverSampler(random_state=3,
                        sampling_strategy={0:30000,})


X_rus, y_rus = rus.fit_resample(X_train, y_train)

# Vérifier le nombre de classes
Counter(y_rus)
```

Figure 11

Oversampling model

```
# Randomly oversample the smaller classes
smote = SMOTE(random_state=3, sampling_strategy={1:22014, 2:19336})

X_resampled, y_resampled = smote.fit_resample(X_rus, y_rus)

# Vérifier le nombre de classes
Counter(y_resampled)
```

Figure 12

SMORE method

	0	1	2	3	4	5	6	7	8
count	1.382200e+04	13822.0	1.382200e+04	1.382200e+04	1.382200e+04	1.382200e+04	13822.0	1.382200e+04	1.382200e+04
mean	3.860318e-17	0.0	4.067632e-16	2.486496e-17	-3.833972e-16	6.375509e-16	1.0	3.083596e-17	1.369425e-16
std	1.000036e+00	0.0	1.000036e+00	1.000036e+00	1.000036e+00	1.000036e+00	0.0	1.000036e+00	1.000036e+00
min	-1.649700e+00	0.0	-4.776165e+00	-2.995673e-02	-4.739070e+00	-1.305447e+01	1.0	-1.718871e+00	-1.503545e+00
25%	-8.800028e-01	0.0	-7.332259e-01	-1.173088e-02	3.122417e-02	-2.602683e-01	1.0	-8.707801e-01	-8.832663e-01

Figure 13

Scaling

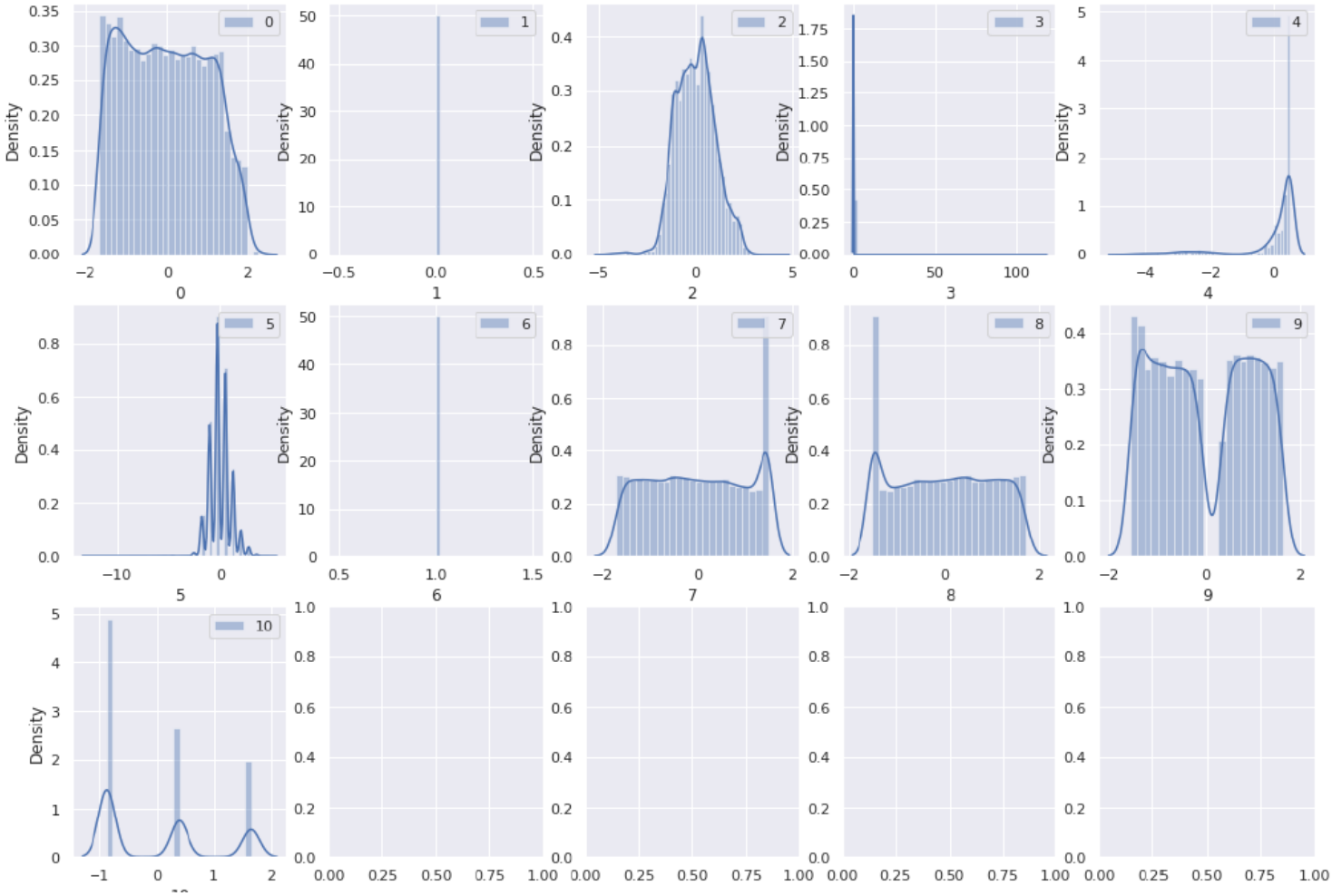

Figure 14

column distribution

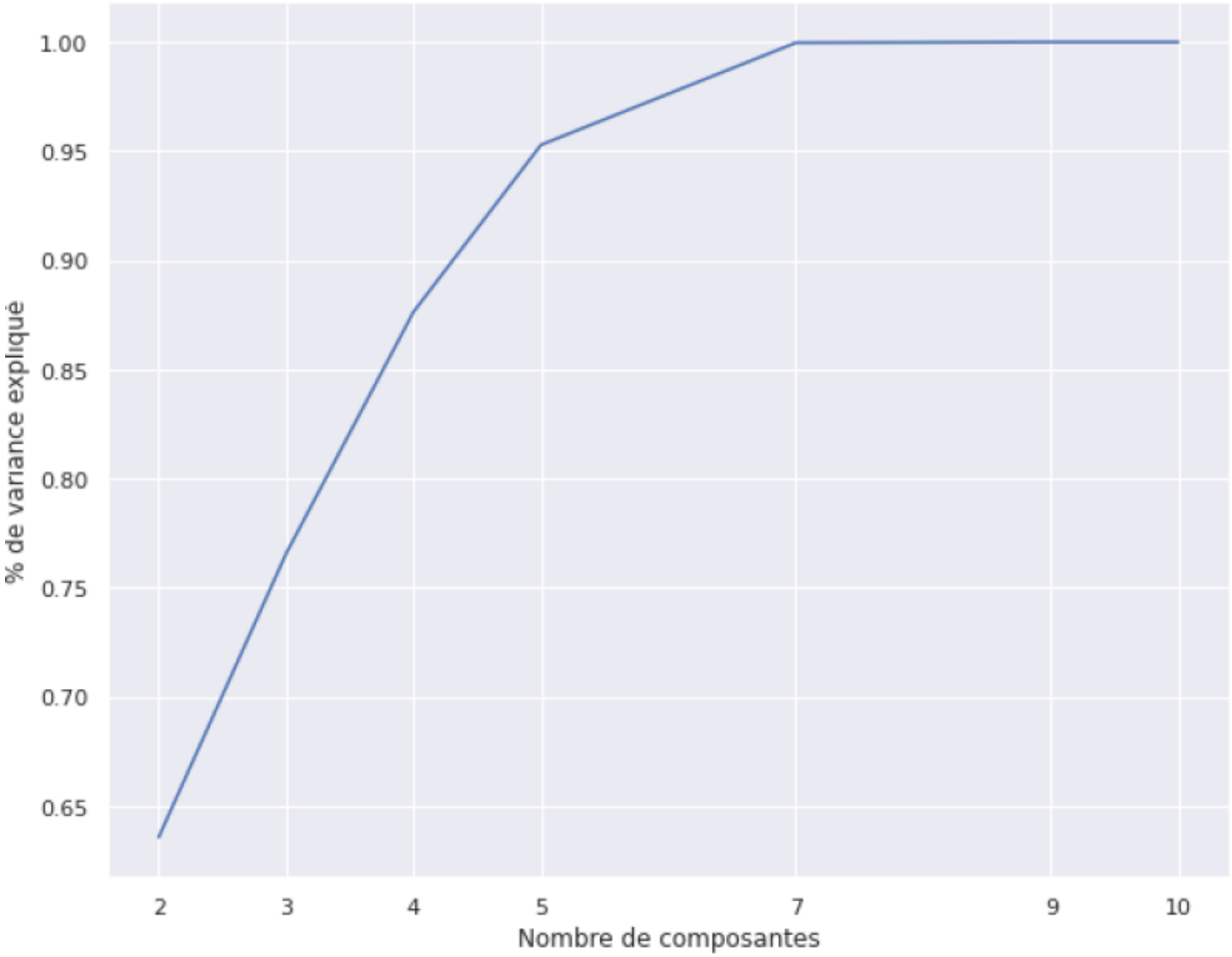

Figure 15

Important components

```
# Build a pipeline for the preprocessing steps
pipe_prepro = Pipeline([('scaler', StandardScaler()),
                        ('pca', PCA(n_components=5))])
```

Figure 16

Building a pipeline for the preprocessing steps

```
# Fit and score a baseline RandomForestClassifier
forest2 = RandomForestClassifier()
forest2.fit(X_train_trans, y_train_res)
```

Figure 17

Random Forest classification.

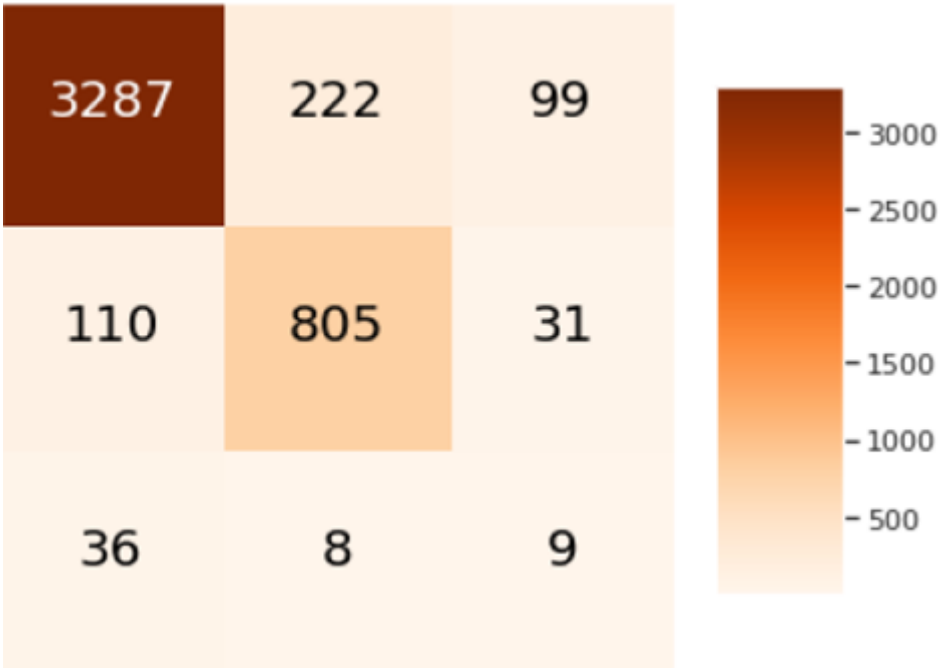

Figure 18

Confusion matrix