Automatic Learning coupled with Interpretability: MBDA in Action

José Camacho*, Rasmus Bro†, David Kotz‡, *Fellow, IEEE*
*Department of Signal Theory, Telematics and Communications, CITIC, University of Granada, Spain
†Chemometrics and Analytical Technology, University of Copenhagen, Denmark
‡Department of Computer Science, Dartmouth College, Hanover, NH 03755, United States

***Abstract*—In this paper, we illustrate the application of Multivariate Big Data Analysis (MBDA), a recently proposed interpretable machine-learning method with application to Big Data sets. We apply MBDA for the first time for the detection and troubleshooting of network problems in a campus-wide Wi-Fi network. Data includes a seven-year trace (from 2012 to 2018) of the network's most recent activity, with approximately 3,000 distinct access points, 40,000 authenticated users, and 600,000 distinct Wi-Fi stations. This is the longest and largest Wi-Fi trace known to date. Furthermore, we propose a new feature-learning procedure that solves an inherent limitation in MBDA: the manual definition of the features. The extended MBDA results in a methodology that allows network analysts to identify problems and diagnose them, which are principal tasks to troubleshoot the network and optimize its performance. In the paper, we go through the entire workflow of the approach, illustrating its application in detail and discussing processing times.**

***Index Terms*—Interpretable Machine Learning, Multivariate Big Data Analysis, Anomaly Detection, Big Data, Parallel Hardware, Dartmouth Campus Wi-Fi**

I. Introduction

Multivariate exploratory analysis has been recognized as an outstanding approach in several domains, including industrial monitoring [1], network security [2] and activity patterns mining [3], marketing [4], weather modeling [5], bioinformatics [6], food research [7], and so forth. In this methodology, visualization, interpretation and data interaction are the principal tools for an analyst to understand the problem the data reflects. This is an alternative data-driven approach to the one currently dominating machine learning, i.e., deep learning, in which the model is built as a black box to approximate an output of interest and little interpretation is left to the analyst. This alternative approach to data analysis, recently referred to as interpretable machine learning, has raised a lot of attention of the research community in recent years [8], [9].

In this paper, we present an automatic-learning extension and case study of the Multivariate Big Data Analysis (MBDA) tool [10] as applied to monitoring and troubleshooting a campus-wide Wi-Fi network [11]. MBDA is a complete multivariate anomaly detection and analysis approach, based on a workflow of five steps, that can handle large amounts of data from disparate sources. When an anomaly is identified, the output includes the log entries of raw information associated with it. These, in turn, can be presented to the analyst, so as to elucidate the root causes for the anomaly. In a context where the number of log entries is massive (i.e., Big Data), MBDA works as a magnifying glass, conveniently highlighting anomalous events. As such, this is one of the first efforts done so far to apply interpretable multivariate analysis in the context of Network Traffic Monitoring and Analysis of Big Data [12], [13].

Our contributions in this paper are as follows.

- We illustrate the application of MBDA to a real case study, showing what it can provide to network analysts and presenting the workflow in detail, including the parallelization of the code and processing time results.
- MBDA has a main limitation: data features used in the analysis are manually defined. We propose an automatic feature-learning procedure, consistent with the MBDA methodology.
- We incorporate state-of-the-art exploratory analysis visualizations within the central step of MBDA, to make the data more interactive.

The rest of the paper is organized as follows. Section II introduces the data under analysis. Section III presents the MBDA methodology in brief. Section IV introduces the learning procedure contributed. Section V illustrates the five steps of MBDA in the Wi-Fi data. Section VI provides conclusions.

II. The Dartmouth Wi-Fi network

Dartmouth College has a compact campus with over 200 buildings on 200 acres. The original evolution of the network is documented in the series of early papers [14], [15]. In this paper we analyse a data capture containing the connections of users to the network in a seven-year time span: from 2012 to 2018 [11]. This data contains Simple Network Management Protocol (SNMP) traps [16] sent from wireless controllers to a collector. The capture reveals the statistics in Table I. The data set contains a total of 5 Billion traps and 7 TB of data. A total of 38K authenticated users and an undetermined number of non-authenticated users have been connected to the network in the last seven years, using 600K devices.

To collect the trace, the Wi-Fi network controllers forwarded SNMP traps with a record of network activity to the Dartmouth team's servers. Figure 1 shows an example of an SNMP trap

Corresponding author: J. Camacho (email: josecamacho@ugr.es).

978-3-903176-27-0

TABLE I: Details of the SNMP trap capture at Dartmouth College.

Statistic	Number
Capture period	Jan 1st 2012 - Dec 31st 2018 (2556 days)
log entries (SNMP traps)	5 Billion
Data Size (raw)	7 TB
Access points	3,330
Authenticated Users	38,096
Stations	624,903
SSIDs	20

as received. Each trap comprises a header, with timestamp and sender and collector information, followed by a variable number of triplets representing SNMP object identifiers (OIDs) with the format '<OID> = <type>: <value>' and separated by hashes (#). OIDs are partly represented in ASN.1 notation, which can be translated into more meaningful OID names using the relevant Management Information Base (MIB). An important OID is the trap type (TT), in which the value is also an OID: '<TT> = OID: <OID>'. Please, refer to [11] for more details on the data capture.

III. Multivariate Big Data Analysis

MBDA makes use of two open software packages available on Github: the MEDA Toolbox [17], [18] and the FCParser [19]. The FCParser is a python tool for the parsing of both structured and unstructured logs. With the MEDA Toolbox, multivariate modeling and data visualization can be performed.

Intensive parsing requires a parallel computer. We used the Anthill Compute Cluster hosted by the Computer Science Department at Dartmouth. It is a 100 node, 1200 core, 4,288GB RAM compute cluster, managed with the grid engine [20] as parallelization software. Matlab and Python scripts using the FCParser and the MEDA Toolbox, respectively, run on top of the parallel hardware as grid jobs.

The MBDA approach consists of 5 steps:

1) Parsing: the raw data coming from structured and unstructured sources are transformed into quantitative features.
2) Fusion: the features of the different sources of data are combined into a single data stream. In the example under analysis there is a single source of data: SNMP traps. Thus, fusion is not required.
3) Detection & Analysis: featured data is visualized and anomalies are identified in time using Principal Component Analysis (PCA) [5], [21] and Multivariate Statistical Network Monitoring (MSNM) [22], [23], [24], [25].
4) Pre-diagnosis: the features associated with an anomaly are found.
5) De-parsing: Using both detection and pre-diagnosis information, the original raw data records related to the anomalies are identified and presented to the analyst.

The first three previous steps are equivalent to what it is commonly done in other machine-learning methodologies. However, steps 4 and 5, which perform the diagnosis of the anomalies, are a major advantage of MBDA. These steps are possible thanks to the white-box, interpretable characteristics of PCA as the core of the MSNM approach in step 3). PCA and associated visualizations provide a means to explore high-dimensional data in a systematic way, easy to interpret in terms of the connection between anomalies and features.

PCA transforms the original features into a lower number of uncorrelated features: the so-called principal components. The principal components are ordered by captured variance. PCA follows the expression:

$$\mathbf{X} = \mathbf{T}_A \cdot \mathbf{P}_A^t + \mathbf{E}_A, \tag{1}$$

where $\mathbf{X}$ represents the matrix of data, with N rows and M columns, $\mathbf{T}_A$ is the $N \times A$ scores matrix containing the projection of the objects in the principal components sub-space, where $A << M$ is the number of principal components, $\mathbf{P}_A$ is the $M \times A$ loadings matrix containing the linear combination of the variables represented in each of the principal components, and $\mathbf{E}_A$ is the $N \times M$ matrix of residuals.

We call the PCA model in eq. (1) a matrix factorization, since the information in $\mathbf{X}$ is factorized in the scores in $\mathbf{T}_A$, the loadings in $\mathbf{P}_A$ and the residuals $\mathbf{E}_A$. This factorization is useful for visualization, because we can explore the distribution of the observations (rows) and of the features (columns) of $\mathbf{X}$ in separate plots of $\mathbf{T}_A$ and $\mathbf{P}_A$, respectively. The latter are of much lower dimension than $\mathbf{X}$, and so easier to visualize, while they retain most of the information in the data.

Scores, loadings and residuals can be visualized using line and scatter plots to gain data understanding. Also, the data can be further compressed in a pair of statistics, the D-statistic and Q-statistic, where anomaly detection can be performed following the MSNM approach. The D-statistic and the Q-statistic for observation n are computed with the following equations:

$$D_n = \mathbf{t}_n \cdot (\Sigma_T)^{-1} \cdot \mathbf{t}_n^t \tag{2}$$

$$Q_n = \mathbf{e}_n \cdot \mathbf{e}_n^t \tag{3}$$

where $\mathbf{t}_n$ is a $1 \times A$ vector with the scores for observation n, $\mathbf{e}_n$ is a $1 \times M$ vector with the residuals, and Σ_T represents the covariance matrix of the scores.

As a summary, Table II describes the general data pipeline. Steps 1) and 3) perform two steps of compression of the data, first from raw data to features and then from features to components/residuals using PCA and from these to statistics using MSNM. Once anomalies are found, steps 4) and 5) allow to identify the root cause in the raw information associated to them.

IV. Learning Counts in Big Data

As noted above, MBDA works as a magnifying glass into massive amounts of data, with a configurable trade-off between the level of detail for data visualization and the capability

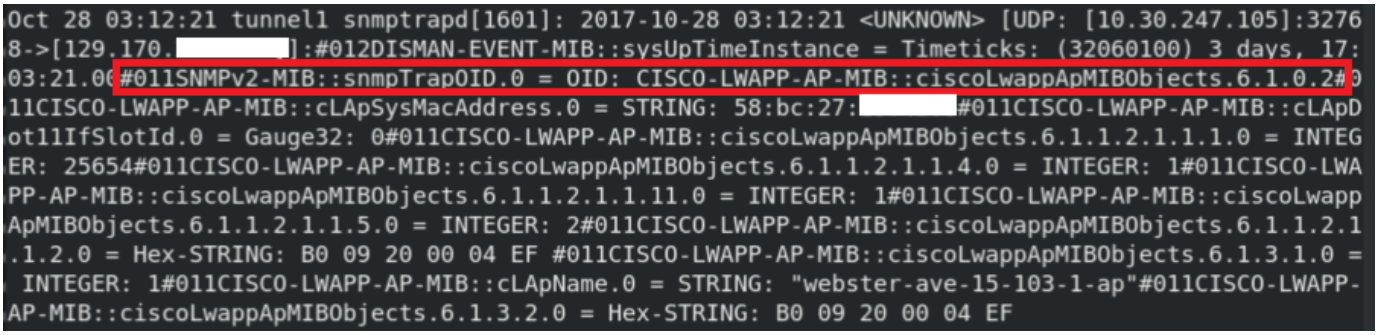


Fig. 1: Example of an SNMP trap in the data capture. The second OID, highlighted by a rectangle, represents the trap type. Parts of an IP and a MAC address have been hidden.

TABLE II: Summary of the five steps in MBDA.

STEP	INPUT	OUTPUT	SOFTWARE
1. Parsing	Raw data stream	Stream of features per source	FCParser
2. Fusion	Stream of features per source	Single feature stream	FCParser
3. Detection	Single feature stream	Timestamps for anomalies	MEDA Toolbox
4. Pre-diagnosis	Single feature stream & Timestamps for anomalies	Features for anomalies	MEDA Toolbox
5. De-parsing	Raw data stream & Timestamps & Features for anomalies	Raw log entries for anomalies	FCParser

for data compression. The key to this trade-off is the parsing step, where we set the features and the time resolution for the subsequent analysis. In the original MBDA proposal [10], the features were manually defined. This represents a strong limitation for the application of MBDA to increasing volumes of data. In this paper, we define an automatic feature learning procedure that is consistent with the parsing methodology in MBDA. In the following subsection, we introduce the parsing methodology and, subsequently, we present the learning procedure.

A. Feature-as-a-counter parsing

MBDA makes use of the feature-as-a-counter (FaaC) approach [2] in step 1). Each feature contains the number of times a given event takes place during a pre-defined time interval. Examples of suitable features are the counts of a given word in a log or the number of traffic flows with given destination port in a *Netflow* file. This flexible feature definition makes it possible to integrate, in a suitable way, most sources of information.

To implement the FaaC, the FCParser defines *variables* and *features*. Variables represent general entities in the data. In the previous two examples, the variables would be *word* and *destination port*. The features are defined for a specific value of a variable. Examples of features would be *word='food'* and *destination port='80'*. This representation in variables and features has the relevant advantage that allows for the definition of *default* features, e.g. *word=<ANY>*, useful to count the instances of a variable, regardless its value, in a data record.

Variables and features are defined using regular expressions in configuration files, where we also set the time resolution of the parsing. Each configuration file typically contains several variables and several features per variable. The FCParser applies this configuration to the data to compute a feature vector for each interval of time present in the original data. This is done using a multi-threading configuration to speed-up computation. By selecting the time resolution and the features, we define the trade-off between level of detail and compression. Defining more features and/or using a lower time resolution result in more detail, while defining fewer features and/or using a higher time resolution leads to more compression.

Take the example in Figure 1, which represents an SNMP trap of October 28. One suitable variable to represent this data could be the trap type, which corresponds to the information enclosed in the red rectangle. The variable can be parsed using a regular expression that identifies the string 'snmpTrapOID.0 = OID: <trap_type>'. To define features for this variable, we have many choices. One choice is to define one feature per different trap type in the data. Imagine there are 100 different trap types in the data, and that we set the time resolution to one day (as we do below, in the case study). Then, the parser would pick all SNMP traps corresponding to October 28, count the number of instances of each trap type, and store the result in a 100-feature vector that represents that day. That way, we compress the logs of a single day (which may be in the millions) into 100 counts. We can increase the level of detail (or conversely compression) in two ways. First, we may reduce the time resolution to, e.g., one hour, so that we represent the entire day with a total of 24 feature vectors, with 100 counts each. This increases the level of detail 24-fold in comparison to a time resolution of one day. We can also raise the time resolution for a higher compression, e.g., to weekly or monthly time intervals, which is useful to manage huge data sets where the number of days is too many for proper visualization. Second, we can also control the level of

detail by increasing or reducing the number of variables and features. For example, adding a feature for each single trap type, as suggested above, may result in a representation that is too sparse and a poor trade-off between compression and detail. Alternatively, we can count only most common trap types, say the top-most 9, and store the count of total traps in a default feature. This leads to feature vectors with 10 features, increasing the compression level 10-fold in comparison to using 100 features.

The default features play a similar role to residuals in PCA and MSNM. Both steps 1) and 3) in MBDA work as lossy compression steps, which are fundamental to visualizing data when the data volume is massive. However, when doing anomaly detection, it is a good idea to include in the model a summary of what is left out in the compression. This allows the analyst to retain the ability to find uncommon patterns in the residual part. A simplistic example of when default features can be useful follows. Imagine we capture traffic from a network where the main services are *http* (destination port 80) and *dns* (destination port 53). To monitor the traffic, we define two features: *destination port='80'* and *destination port='53'*, and a default feature *destination port=<ANY>*. During traffic monitoring, there is a sudden burst of Internet Relay Chat (*IRC*) traffic with destination port 6667, mixed into the stable *http* and *dns* traffic. If we only consider the first two features for anomaly detection, we cannot identify this burst, since the counts of *http* and *dns* traffic remained stable. However, we will see an increase of the counts in the default feature, as a consequence of the burst. As follows from this example, the default features are useful to detect general events, but they are not very useful for diagnosis: in the example, the default feature will be affected by any type of burst, but we miss the information about the type of traffic that is causing it, information that we do have for specific features. If something anomalous is detected in a default feature and the analyst needs to diagnose the problem, she can decide whether to redefine features to account for the new situation or simply look at the raw data for diagnosis.

B. Learning procedure

The original MBDA [10] relies in the manual definition of the features in the configuration files of the FCParser. To write such configuration files, the analyst needs to get familiarized with the data. Let us come back to the example in Figure 1. To understand the best approach to code the SNMP traps into variables and features, we need to get an idea of the distribution of the OIDs in the entire data set: is the trap type a good variable to represent the data? or conversely all the traps correspond to the same type, and then this information is useless? In a regular-size data set, we can identify useful variables/features by visually inspecting the data and/or taking simple statistics. Unfortunately, in a practical Big Data problem like the one under analysis, the data capture is simply too massive for direct inspection. Besides, the fact that SNMP traps are unstructured data, that is, traps have a disparate structure of fields and values, renders infeasible the derivation of simple statistics. If we want to obtain a good description of the content, we need to apply an automatic feature-derivation technique that can handle unstructured records. The definition of this technique is not straightforward, since it needs to be consistent with the subsequent multivariate analysis, so that we maximize compression while retaining the observability required for anomaly detection and diagnosis.

There are two basic properties we would like to have in the learning procedure. First, the main sources of variance need to be captured. Second, uncommon characteristics with low variance should also be modeled somehow, in a summary of residual information. The second feature is built-in to the FaaC methodology with the definition of default features, as already discussed.

We developed a learning algorithm to automatically identify a list of common FaaC features in a Big Data set, and included it in the FCParser repository at Github with the name `fclearning.py`. The learning algorithm extracts the features in the data ordered by their percentage of presence, measured as the portion of the log entries where a feature appears. It also defines default features and automatically writes the configuration files.

We used this algorithm in two steps to identify high variance OIDs in the Wi-Fi data. First, the algorithm was parallelized in 2556 processing jobs, one per different day in the capture. The resulting 2556 configuration files contain the FaaC features with a percentage of presence above 5% during that day. Second, these configuration files were combined in a single configuration file, where we discarded all features with variance below a threshold.[1] This resulted in a total of 90 features, including default features.

The whole learning process using the parallel hardware and multi-threading (4 threads per processor) took 12 hours, during which a maximum of 150 jobs were processed in parallel. This means that the processing time could be further reduced 17-fold using a larger computer cluster, where as many as 2556 jobs could be run in parallel. The final combination of the regular expressions learned for variables and associated features into a single configuration file took another 30 minutes.

V. MBDA IN ACTION

In this section, we discuss the application of MBDA to the Wi-Fi data set using the learned features. The structure of the section follows the steps in Table II. Since there is one single source of data, the second step (fusion) is not necessary.

A. Feature extraction

We use the FCParser to generate the feature vectors with the aforementioned configuration file learned from the data. In agreement with the learning phase, we consider feature vectors for intervals of one day. These contain the number of traps, the total number of OIDs and the number of learned features. This makes a total of 92 features. Each day in the original SNMP capture is transformed into a feature vector

[1] We used a threshold of 0.01%, taking the variance in the number of log entries per day as a reference.

we call 'observation', and each feature in the observation is the number of times the corresponding element is found in the traps of that day. This results in a compression of the data from 7TB to less than 1MB, yielding 2556 observations (days) of 92 features each in matrix $\mathbf{X}$. The compression conveniently transforms a Big Data set into a handleable data set by any analysis package in a common computer. Again, we can vary the level of detail by using different time resolutions or number of features.

The parsing was parallelized in 2556 processing jobs, one per day, and the whole process using the Anthill Computer Cluster and multi-threading (with a maximum of 150 jobs) took 15 hours.

B. Detection & Analysis

Basically, PCA factorizes the data $\mathbf{X}$ into a part for the observations (the scores) and a part for the features (the loadings), making the visualization simpler and more interpretable. It compresses data, highlighting the main patterns of variance. Therefore, by only inspecting a small set of principal components, we can gain accurate insight into the data. For the compression, the number of principal components to use has to be determined. There are many methods to aid in that decision [21], [26]. In our case, the data exploration is barely affected by the selection of the number of components, because we also visualize a summary of the residuals.

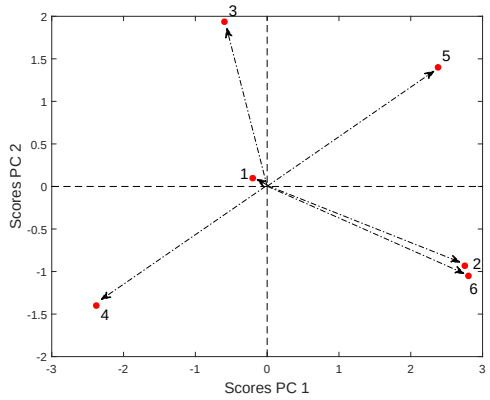


Fig. 2: Illustration of interpretation in score and loading plots.

Once we have selected the number of principal components, we can visualize the data using scatter plots of scores and loadings. We can infer relationships between elements in the data from their location in those plots. Figure 2 shows a simple example of a score plot, where we can see six points representing observations (rows in the data) scattered around the center of coordinates. The relationship between observations is revealed by the distance between observations and the angle between vectors. Observations with low scores are close to the center of coordinates, as illustrated with observation 1, for which the plot does not provide much useful information. Vectors with the same direction indicate similarity, and the closer the points the more similar, as is the case for observations 2 and 6. Vectors in the opposite direction show an inverse relationship around the center of coordinates, like observations 4 and 5. Vectors in different directions reflect they are somehow different (not directly or inversely related). Thus, observation 3 is different from the group of 2 and 6 and from the group of 4 and 5. PCA is a good visualization tool because it tends to separate different elements in the data, providing a good description of their differences. This property is useful: if the score plot highlights different groupings of observations, we can conclude that there is an underlying difference in the information content in the features of these groups.

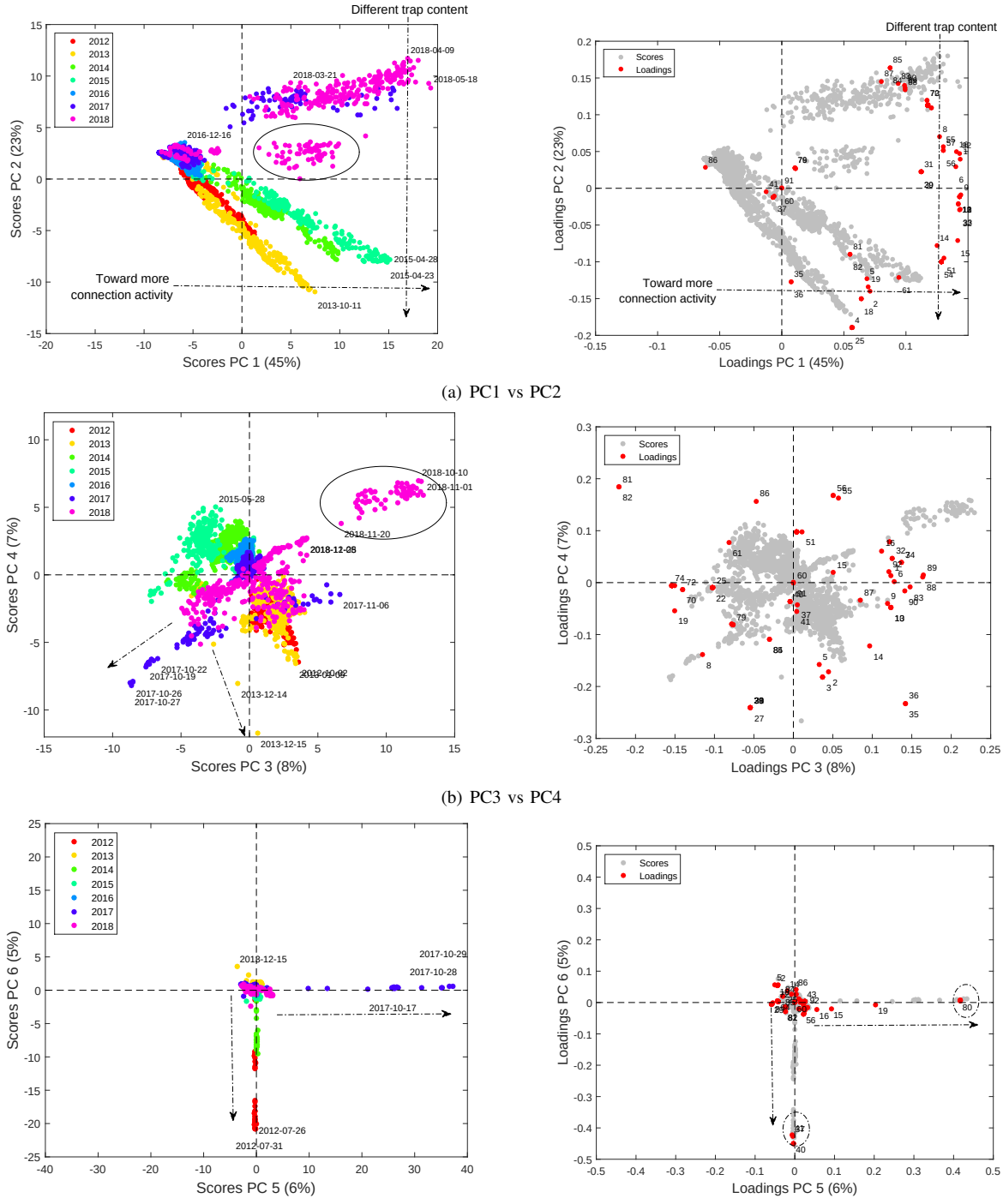
Loading plots are similar to score plots, but represent the spacial distribution of features (columns in the data) instead of observations. Again, clusters, trends or outliers can be identified in the loading plot, and lead to interesting conclusions on the features' distribution. We can also combine scores and loadings in a single plot, commonly called a *biplot* [27]. Well-designed biplots allow similar comparisons in distance and angle between observations and features. Thus, if one observation is located close to a feature in the biplot, we expect this observation to have a high value of that feature. This property is useful to draw connections between the patterns of observations and features: e.g., to identify which features make an outlier different from the rest of observations.

The plots corresponding to the first 6 principal components in the Wi-Fi data are depicted in Figure 3. For simplicity, we show scatter plots of consecutive PCs, but other choices are possible (e.g., we could show PC1 vs PC4). Recall that matrix $\mathbf{X}$ contains 2556 rows, representing days of the capture, and 92 features. We present score plots at the left of the figure and loading plots at the right. Visualizing score plots and loading plots together provides a bird's-eye view of the data. In the score plots, points represent the 2556 days of the data capture and are colored according to the year. In the loading plots, (red) points represent the 92 features and, in order to facilitate interpretation, we also plot a (gray) shadow of the scores, like in a biplot.

Figure 3(a) represents the score and loading plot for PC1 vs PC2. These two principal components represent 68% (45% + 23%) of the variance in the data. Given that variance is a measure of the variability within the data set, these two PCs show the main patterns of change in $\mathbf{X}$. As a matter of fact, a variance of 68% roughly indicates that only 1/3 of the patterns of variability in the data is missing in this plot, giving an idea of how powerful PCA is for visualization.

The score plot at the left of Figure 3(a) shows that the dots (days) with different colors are in different locations. This means that they are different in content, from which follows that there are large differences in prevalence of OIDs in different years.

The loading plot at the right of Figure 3(a) can be interpreted in connection with the score plot. Recall that in a combined score/loading plot, the location of an observation (a day) will approach the location of a feature (which represents counts of a specific OID) as the value of that feature increases in the observation. Thus, days with a large content on specific OIDs



(a) PC1 vs PC2

(b) PC3 vs PC4

(c) PC5 vs PC6

Fig. 3: PCA scores: PC1 vs PC2 (a), PC3 vs PC4 (b) and PC5 vs PC6 (c).

will be located closer in the plot to the loading representing that OID. The loading plot shows that a large majority of the features are located far from the center of coordinates towards the right side. Therefore, any day toward the right in the score plot will have a generally higher content of OIDs. Thus, as we traverse from left to right in the score plot, the days will have more connection activity. Busy periods are represented towards the far right of the plot, and vacations are clustered to the left, and we could say that the first PC (the horizontal direction in the score and loading plots) represents the general activity in the network. We annotated this in both plots using a horizontal arrow.

The loading plot in Figure 3(a) also shows that the variables are distributed from the bottom to top, and we see a similar

distribution for the different years in the score plot: the first two years are in the bottom and the last two in the top, with middle years in between. We also see a separated cluster of days in 2018, highlighted with a circle. A closer look reveals that all the days in the cluster belong to the period from September to November, when eduroam replaced Dartmouth Secure. The vertical pattern in the loading and score plots shows that the distribution of traps has changed across the years: days towards the top have a higher content of traps and OIDs represented by the features in the top and less of those in the bottom, and vice-versa. Again, we annotated this in the score and loading plots using a vertical arrow. Questioned about this difference, the network staff replied that there was an update in the controllers' software, which changed the types of traps that were collected. This variability in traps for different temporal periods makes the analysis of the data a real challenge.

Figure 3(b) represents PC3 vs PC4. Here we see again some differences among the years and the cluster of 2018. We also see a set of consecutive days from 2017 that depart from the rest of days, which is reflecting that something unusual took place during those days. This type of pattern is sometimes referred to as an excursion of scores. Another 2-day excursion is shown in 2013. Both excursions are annotated with an arrow. The corresponding loading plot is difficult to interpret in connection with the observed deviations, but there are additional tools [28], [29], [30] that can be used to provide more information. The use of these tools will be illustrated in the next step of the approach: the pre-diagnosis.

Figure 3(c), representing PC5 vs PC6, shows again the excursion of 2017 and another one occurring both in 2012 and 2014. The loading plot allows us to associate the excursions of 2012-2014 and 2017 to specific features (OIDs) annotated with circles. The excursions of 2012 and 2014 are associated to a high number of failures in the RADIUS server, and the one in 2017 to a high number of restarts of APs. We provide more detail below.

No additional information was revealed from inspecting the next two principal components (not shown), so we decided to stop the initial analysis here. Besides, the residuals account for only 6% of the variance in the data, so we can be confident that the main patterns of change are already in the previous plots.

After the inspection of score and loading plots, one can visualize a summary of the whole data distribution in one single plot using MSNM: a scatter plot of the observations in terms of the D-statistic and the Q-statistic. In this plot we can also show upper control limits (UCLs) to facilitate the detection of anomalies. UCLs leave below-normal observations with a certain confidence level, e.g., 99%. They can be used as hypothesis tests, so that all observations above the limits reject the null hypothesis that they are normal. More information on how to compute these limits can be found elsewhere [24], [22].

The MSNM plot for the Wi-Fi data is shown in Figure 4. Anomalies are expected to surpass any of the two control limits. This plot is optimized for anomaly detection, and the

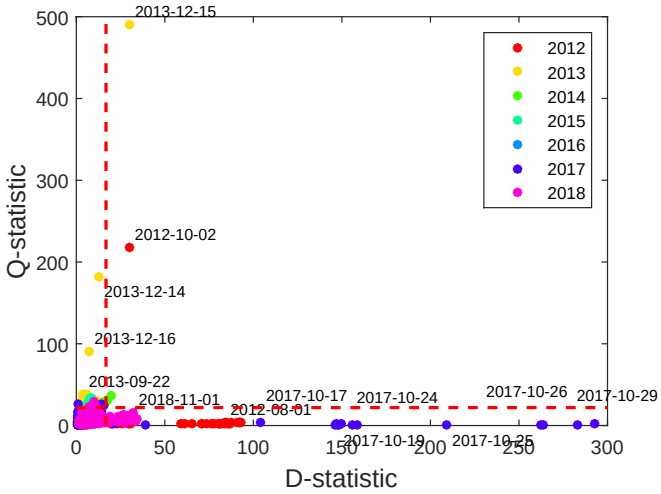


Fig. 4: Multivariate Statistical Network Monitoring (MSNM) plot: D-statistic vs Q-statistic.

excursions mentioned before are clearly observed. However, in the plot we miss other details, like the yearly and seasonal patterns, as well as the difference in trap contents. A main advantage of this plot is that it also includes residuals, containing the remaining 6% of the variance that is not accounted for in the 6 principal components. The Q-statistic, which comprises a summary of the residuals, clearly identifies the excursion in 2013 and another anomaly in 2012.

Regarding processing burden, the analysis performed in this section is completely interactive in a regular computer, meaning that the time to obtain each of the plots shown in this section is in the order of seconds.

Note that the parsing (and thus of the learning process) has a principal impact in the visualization and anomaly detection of MBDA. For instance, we can only detect anomalies (e.g. excursions) at the day level when using one-day resolution. If we want to detect anomalies in other time resolutions, we need to modify the parsing configuration. Furthermore, differences in OID content can only be visualized if we include features for those OIDs. Therefore, learning features of high variance is paramount to obtaining accurate insights of the data distribution.

C. *Pre-diagnosis*

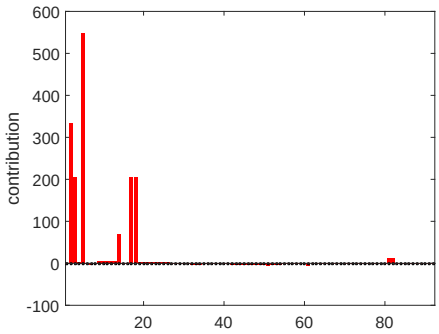
The plots discussed in the previous step provide a general view of the data and can assist in detecting patterns, like trends, outliers, excursions or clusters. However, if we want to have more detail on the interaction between observations and features, multivariate diagnosis tools are more accurate and easier to interpret.

There are many multivariate diagnosis tools [31], [32]. The MEDA Toolbox includes the oMEDA plot for that purpose. It is a bar plot of the features, built to compare two groups of observations: e.g., to compare observations in an excursion with normal observations. Each bar represents the difference

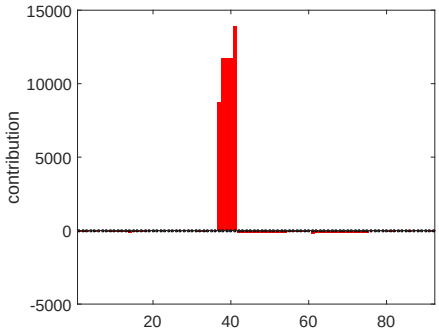
TABLE III: Pre-diagnosis of the excursions of 2013 and 2017 with oMEDA.

Timestamps	Features selected
2013-12-14 – 2013-12-16	bsnDot11StationAuthenticateFail, bsnAuthenticationFailure, bsnDot11StationAssociateFail, bsnStationReasonCode, bsnAuthFailureUserType, bsnAuthFailureUserName
2017-10-16 – 2017-10-30	ciscoLwappApIfUpNotify, ciscoLwappApIfDownNotify cLApAdminStatus, cLApSysMacAddress, cLApPortNumber

between both groups in that feature. A positive bar implies that the first group of observations presents a higher value in the corresponding feature than the second group. A negative bar reflects the opposite. A bar close to zero for a feature means that both groups of observations have a similar value in it.



(a) Excursion in 2013



(b) Excursion in 2017

Fig. 5: Pre-diagnosis of the excursions of 2013 and 2017 with oMEDA.

To illustrate the use of oMEDA, we selected the excursions in 2013 and 2017, which where shown as the main outliers in the Q-statistic and in the D-statistic, respectively. The plots are shown in Figure 5. Both plots indicate that each of the excursions are related to a different set of features, meaning that days within the excursions showed high prevalence of specific OIDs, from which we infer that root causes for each excursion are different. The specific OIDs are listed in Table III. We can use this information to infer the root causes. Thus, we determined that the first excursion is related to a large number of Authentication Fails (a particular type of SNMP trap), with a prevalence of one order of magnitude higher than usual. The second excursion is related to an unprecedentedly high number of re-starts of APs – two orders of magnitude higher than usual.

As for 2018, the network staff did not have any additional records for these old anomalies, besides the SNMP traps, but they suggested that the second one could be related to the installation of a security patch after the publication of a vulnerability. Recall that on 16 October the famous KRACK attack against WPA2 [33] and the corresponding patch were released to the public. Even if a restart is necessary after a patch installation, the number and duration (15 days) of the event is remarkable, evidencing that a major management activity took place.

Again, the pre-diagnosis step is complete in seconds and easily done in a regular computer.

D. Deparsing

While the visualizations already provided to the analyst are informative, it is a good idea to identify the raw log entries related to the patterns found to obtain more detail about them. At this point, MBDA works as an aid that allows analysts to make the most of their time by focusing on the relevant information alone. Using the raw logs retrieved, we can provide some additional interpretation with our preferred log analysis tool.

The deparsing algorithm [10] takes as inputs the timestamps of a given anomalous pattern, detected in step 3), and the features associated with it, identified in step 4). With this information, the FCParser matches the regular expressions of the features in the specific raw data files. The output is the set of raw log entries that matches at least one of the features, ordered by the number of features they match. With these log entries, the analyst can extract detailed information about the anomalies.

The deparsing algorithm was applied to the excursions in 2013 and 2017 in the Wi-Fi data set. We parallelized the processing using the Anthill Computer Cluster and multi-threading (4 threads per processor), an with as many parallel jobs as days in the excursions. The first excursion took 30 minutes to be processed and retrieved 5.4M traps, and the second one 135 minutes, retrieving 19M traps.

VI. Conclusion

In this paper, we show a case study of the application of the Multivariate Big Data Analysis (MBDA) tool. The application is concerned with the detection and diagnosis of communication failures in a Wi-Fi campus network. The results illustrate that MBDA can bring light into complex massive data sets.

MBDA can work on top of parallel hardware in order to speed up computation. We analyzed 7TB of data in a little more than a day, and this time can be reduced to a couple of hours in a high-throughput cluster. To handle Big Data with MBDA, we had to extend the original methodology with a new feature-learning procedure, which is a main contribution of this paper.

ACKNOWLEDGEMENT

This work was supported by Dartmouth College, and in particular by the many network and IT staff who assisted us in configuring the Wi-Fi network infrastructure to collect data, and who patiently answered our many questions about the network and its operation. We furthermore appreciate the support of research colleagues and staff who have contributed to our data-collection and data-analytics infrastructure over the years: most notably Wayne Cripps, Tristan Henderson, Patrick Proctor, Anna Shubina, and Jihwang Yeo. Some of the Dartmouth effort was funded through support from ACM SIGMOBILE and by an early grant from the US National Science Foundation under award number 0454062. This work was also supported by the Ministerio de Educación, Cultura y Deporte under the Programa Estatal de Promoción de Talento y su Empleabilidad en I+D+i, Subprograma Estatal de Movilidad, del Plan Estatal de Investigación Científica y Técnica y de Innovación 2013-2016, grant number PRX17/00320 (Associated to a Fulbright Scholarship), and the Plan Propio de la Universidad de Granada, grant number PP2017.VS.02. Jose Manuel García-Giménez is acknowledged for his enthusiastic work on the FCParser.

REFERENCES

[1] A. Ferrer, "Latent structures-based multivariate statistical process control: A paradigm shift," *Quality Engineering*, vol. 26, no. 1, pp. 72–91, 2014.

[2] J. Camacho, G. Maciá-Fernández, J. Díaz-Verdejo, and P. García-Teodoro, "Tackling the Big Data 4 Vs for anomaly detection," in *Proceedings of IEEE INFOCOM*, 2014, pp. 500–505.

[3] G. Poucin, B. Farooq, and Z. Patterson, "Activity patterns mining in wi-fi access point logs," *Computers, Environment and Urban Systems*, vol. 67, pp. 55 – 67, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0198971516300904

[4] J. Hernández-Méndez, F. Muñoz Leiva, and J. Sánchez-Fernández, "The influence of e-word-of-mouth on travel decision-making: consumer profiles," *Current Issues in Tourism*, vol. 1-14, pp. 1–21, 12 2013.

[5] I. Jolliffe, *Principal component analysis*. EEUU: Springer Verlag Inc., 2002.

[6] H. Zou, T. Hastie, and R. Tibshirani, "Sparse Principal Component Analysis," *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 265–286, 2006.

[7] R. Bro, "Multi-way analysis in the food industry: models, algorithms, and applications," Ph.D. dissertation, 1998, kode for udgivelsesland: 'dk'.

[8] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[9] C. Molnar, *Interpretable machine learning*. Lulu. com, 2019.

[10] J. Camacho, J. M. García-Giménez, N. M. Fuentes-García, and G. Maciá-Fernández, "Multivariate Big Data Analysis for Intrusion Detection: 5 steps from the haystack to the needle," *Computer &Security (COSE)*, vol. 87, p. 11, August 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818307909?dgcid=rss_sd_all

[11] J. Camacho, C. McDonald, R. Peterson, X. Zhou, and D. Kotz, "Longitudinal analysis of a campus wi-fi network," *Computer Networks*, vol. 170, p. 107103, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128619308187

[12] A. Furno, M. Fiore, and R. Stanica, "Joint spatial and temporal classification of mobile traffic demands," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[13] A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia, and P. Casas, "A survey on big data for network traffic monitoring and analysis," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 800–813, Sep. 2019.

[14] D. Kotz and K. Essien, "Analysis of a campus-wide wireless network," *Wireless Networks*, vol. 11, no. 1–2, pp. 115–133, Jan. 2005.

[15] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," *Computer Networks*, vol. 52, no. 14, pp. 2690–2712, Oct. 2008.

[16] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol (SNMP)," Internet Requests for Comments, RFC Editor, RFC 1157, May 1990. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1157.txt

[17] J. Camacho, A. Pérez-Villegas, R. A. Rodríguez-Gómez, and E. Jiménez, "Multivariate exploratory data analysis (MEDA) toolbox for Matlab," *Chemometrics and Intelligent Laboratory Systems*, vol. 143, no. 0, pp. 49–57, 2015.

[18] "GitHub repository for the MEDA Toolbox," https://github.com/josecamachop/MEDA-Toolbox, accessed: 2018-09-30.

[19] "GitHub repository for the FCParser," https://github.com/josecamachop/FCParser, accessed: 2018-09-30.

[20] "Open grid scheduler," http://gridscheduler.sourceforge.net, accessed: 2018-09-30.

[21] J. Jackson, *A User's Guide to Principal Components*. England: Wiley-Interscience, 2003.

[22] J. Camacho, A. Pérez-Villegas, P. García-Teodoro, and G. Maciá-Fernández, "PCA-based multivariate statistical network monitoring for anomaly detection," *Computers & Security*, vol. 59, pp. 118–137, June 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404816300116

[23] J. V. Kresta, J. F. Macgregor, and T. E. Marlin, "Multivariate statistical monitoring of process operating performance," *The Canadian Journal of Chemical Engineering*, vol. 69, no. 1, pp. 35–47, Feb. 1991.

[24] P. Nomikos and J. F. MacGregor, "Monitoring batch processes using multiway principal component analysis," *AIChE Journal*, vol. 40, no. 8, pp. 1361–1375, 1994.

[25] A. Ferrer, "Multivariate statistical process control based on principal component analysis (MSPC-PCA): Some reflections and a case study in an autobody assembly process," *Quality Engineering*, vol. 19, no. 4, pp. 311–325, 2007.

[26] E. Saccenti and J. Camacho, "Determining the number of components in principal components analysis: A comparison of statistical, crossvalidation and approximated methods," *Chemometrics and Intelligent Laboratory Systems*, vol. 149, Part A, pp. 99–116, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169743915002579

[27] K. Gabriel, "The biplot graphic display of matrices with application to principal component analysis," *Biometrika*, vol. 58, pp. 453–467, 1971.

[28] T. Kourti, P. Nomikos, and J. F. MacGregor, "Analysis, monitoring and fault diagnosis of batch processes using multiblock and multiway PLS," *Journal of Process Control*, vol. 5, no. 4, pp. 277–284, 1995.

[29] J. A. Westerhuis, S. P. Gurden, and A. K. Smilde, "Generalized contribution plots in multivariate statistical process monitoring," *Chemometrics and Intelligent Laboratory Systems*, vol. 51, pp. 95–114, 2000.

[30] J. Camacho, "Observation-based missing data methods for exploratory data analysis to unveil the connection between observations and variables in latent subspace models," *Journal of Chemometrics*, vol. 25, no. 11, pp. 592–600, 2011.

[31] C. F. Alcala and S. J. Qin, "Reconstruction-based contribution for process monitoring," *Automatica*, vol. 45, no. 7, pp. 1593–1600, 2009.

[32] M. Fuentes-García, G. Maciá-Fernández, and J. Camacho, "Evaluation of diagnosis methods in PCA-based multivariate statistical process control," *Chemometrics and Intelligent Laboratory Systems*, vol. 172, pp. 194–210, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169743917302046

[33] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. ACM, 2017, pp. 1313–1328. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134027