

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/342945892>

# Combining K-Means and XGBoost Models for Anomaly Detection Using Log Datasets

Article in *Electronics* · July 2020

DOI: 10.3390/electronics9071164

---

CITATIONS  
26

READS  
3,314

---

4 authors:



João Henriques  
University of Coimbra  
36 PUBLICATIONS 47 CITATIONS

[SEE PROFILE](#)



Filipe Caldeira  
Polytechnic Institute of Viseu  
59 PUBLICATIONS 208 CITATIONS

[SEE PROFILE](#)



Tiago J. Cruz  
University of Coimbra  
105 PUBLICATIONS 1,038 CITATIONS

[SEE PROFILE](#)



Paulo Simoes  
University of Coimbra  
191 PUBLICATIONS 1,285 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



ATENA project (<https://www.atena-h2020.eu/>) [View project](#)



5G EPICENTRE [View project](#)

Article

# Combining K-Means and XGBoost Models for Anomaly Detection Using Log Datasets

João Henriques <sup>1,†</sup>, Filipe Caldeira <sup>2,‡</sup>, Tiago Cruz <sup>3,‡</sup>, and Paulo Simões <sup>4,‡</sup>

<sup>1</sup> Department of Informatics Engineering, University of Coimbra, Coimbra 3030-290, Portugal; Informatics Department, Polytechnic of Viseu, Viseu, Portugal; jpmh@dei.uc.pt

<sup>2</sup> Department of Informatics Engineering, University of Coimbra, Coimbra 3030-290, Portugal; CISeD – Research Centre in Digital Services, Polytechnic of Viseu, Portugal; caldeira@estgv.ipv.pt

<sup>3</sup> Department of Informatics Engineering, University of Coimbra, Coimbra 3030-290, Portugal; tjcruz@dei.uc.pt

<sup>4</sup> Department of Informatics Engineering, University of Coimbra, Coimbra 3030-290, Portugal; psimoes@dei.uc.pt

† These authors contributed equally to this work.

Version July 9, 2020 submitted to Electronics

**Abstract:** Computing and networking systems traditionally record their activity in log files, which have been used for multiple purposes such as troubleshooting, accounting, post-incident analysis of security breaches, capacity planning and anomaly detection. In earlier systems those log files were processed manually by system administrators, or with the support of basic applications for filtering, compiling and pre-processing the logs for specific purposes. However, as the volume of these log files continues to grow (more logs per system, more systems per domain), it is becoming increasingly difficult to process those logs using traditional tools, especially for less straightforward purposes such as anomaly detection. On the other hand, as systems continue to become more complex, the potential of using large data-sets built of logs from heterogeneous sources for detecting anomalies without prior domain knowledge becomes higher. Anomaly detection tools for such scenarios face two challenges. First, to devise appropriate data analysis solutions for effectively detecting anomalies from large data sources, possibly without prior domain knowledge. Second, to adopt data processing platforms able to cope with the large data-sets and complex data analysis algorithms required for such purposes. In this paper we address those challenges by proposing an integrated scalable framework that aims at efficiently detecting anomalous events on large amounts of unlabeled data logs. Detection is supported by clustering and classification methods that take advantage of parallel computing techniques in environments. We validate our approach using the well known NASA Hypertext Transfer Protocol (HTTP) logs data-sets. Fourteen features were extracted in order to train a K-Means model for separating anomalous and normal events in highly coherent clusters. A second model, making use of the XGBoost system implementing a Gradient Tree Boosting algorithm, uses the previous binary clustered data for producing a set of simple interpretable rules. These rules represent the rationale for generalizing its application over massive number of unseen events in a distributed computing environment. The classified anomaly events produced by our framework can be used, for instance, as candidates for further forensic and compliance auditing analysis in security management.

**Keywords:** Anomaly detection; clustering; K-Means; Gradient Tree Boosting; XGBoost.

## 1. Introduction

Hosts and network systems typically record their detailed activity in log files with specific formats, which are valuable sources for anomaly detection systems. The growing number of hosts per

organization, and the growing complexity of infrastructures, results in increasingly massive amount of recorded logs available – requiring simpler and cheaper anomaly detection methods. While classic log management applications based on manual or preset rule-based analysis still hold value, they don't scale well with the large volumes of data that are currently available. Moreover, they are limited in terms of exploratory analysis: they fail to detect anomalies not predefined in the rules (i.e. based on prior knowledge) and/or require considerable operator's expertise to reach their full potential. This opens the way for the introduction of new approaches, less dependent on prior knowledge and human-guided workflows, able to extract knowledge from large volumes of log data in a scalable and (semi)automated way. Moreover, taking advantage of the available computational resources may also contribute to achieve performance and accuracy for identifying anomalies and to retrieve forensic and compliance auditing evidence.

Over the past years, several automated log analysis methods for anomaly detection have been proposed. However, most of those proposals are not suitable to scale for identifying unknown anomalies from the growing high-rate amount of logs being produced and their inherent complexity. In the scope of the ATENA H2020 Project [1] [2], we faced this challenge while building a Forensics and Compliance Auditing (FCA) tool able to handle all the logs produced by a typical energy utility infrastructure.

To address such challenges, we researched novel integrated anomaly detection methods employing parallel processing capabilities for improving detection accuracy and efficiency over massive amounts of log records. These methods combine the K-Means clustering algorithm [3] and the Gradient Tree Boosting classification algorithm [4] to leverage the filtering capabilities over normal events, in order to concentrate the efforts on the remaining anomaly candidates. Such an approach may greatly contribute to reduce the involved computational complexity.

The characteristics of abnormal system behaviors were obtained by extracting 14 statistical features containing numerical and categorical attributes from the logs. Then, the K-Means clustering algorithm was employed to separate anomalous from normal events into two highly coherent clusters. The previous binary clustered data serves as labeled input to produce a Gradient Tree Boosting algorithm implemented by the XGBoost system [5]. Its role is to produce a set of simple rules with the rationale for generalizing the classification of anomalies of a large number of unseen events in a distributed computing environment. K-Means, XGBoost and Dask [6] provide the tools for building scalable clustering and classification solutions to find out the candidate events for forensic and compliance auditing analysis.

The rest of this paper is organized as follows. Section 2 discusses background concepts and related work. Section 3 describes the proposed framework. Section 4 presents the validation work and discusses achieved results, and Section 5 concludes the paper.

## 2. Background and Related Work

This section starts by providing the reader with the key base concepts related with the scope of our approach. Next, we discuss related work (Section 2.2). Finally, we present the algorithms and tools we adopted in our work, namely K-Means (Section 2.3), Decision Trees (Section 2.4), Gradient Tree Boosting on XGBoost (Section 2.5) and Dask (Section 2.6).

### 2.1. Base Concepts

By definition, an anomaly is an outlying observation that appears to deviate markedly from other members [7]. Anomalies are typically classified into three types: point anomalies, contextual anomalies, and collective anomalies. A point anomaly in data significantly deviates from the average or normal distribution of the rest of the data [8]. A contextual anomaly is identified as anomalous behavior constrained to a specific context, and normal according to other contexts [8]. Collection of data instances may reveal collective anomalies while anomalous behavior may not be depicted when analyzed individually. [9]. Time series data include a significant amount of chronologically ordered

77 sequence data samples values retrieved at different instants. Their features include high-dimension,  
78 dynamicity, high levels of noise, and complexity. Consequently, in the data mining research area, time  
79 series data mining was classified as one of the ten most challenging problems [10].

80 Anomaly detection for application log data faces important challenges due to the inherent  
81 unstructured plain text contents, redundant runtime information, and the existence of a significant  
82 amount of unbalanced data. Application logs are unstructured and stored as plain text, and their  
83 format varies significantly between applications. This lack of structure presents important barriers  
84 to data analysis. Moreover, runtime information such as server IP addresses may change during  
85 execution. Additionally, application log data are designed to record all changes to an application and  
86 hence contain data that is significantly unbalanced in comparison to non-anomalous execution. The  
87 size and unbalanced nature of log data thus complicates the anomaly detection process.

## 88 2.2. Related Work

89 Various anomaly detection methods have been proposed for applying clustering algorithms to  
90 detect unknown abnormal behaviors or potential security attacks.

91 Some of those proposals have addressed the usage of log analysis as one of the input sources for  
92 anomaly detection. Chen and Li [11], for instance, proposed an improved version of the DBSCAN  
93 algorithm for detecting anomalies from audit data while updating the detection profile along its  
94 execution. Syarif et al. [12] compare five different clustering algorithms and identify those providing  
95 the highest detection accuracy. However, they also conclude those algorithms are not mature enough  
96 for practical applications. Hoglund et al. [13], as well as Lichodzijewski et al. [14], constructed a  
97 host-based anomaly detection systems which applied the self-organizing maps algorithm to evaluate  
98 if an user behaviour pattern is abnormal.

99 Often, clustering techniques such as the K-Means algorithm are used by intrusion detection  
100 systems for classifying normal or anomalous events. Münz et al. [15] applied K-Means clustering  
101 algorithm to feature datasets extracted from raw records, where training data is divided into clusters  
102 of time intervals for normal and anomalous traffic. Li and Wang [16] improved a clustering algorithm  
103 supported by a traditional means clustering algorithm, in order to achieve efficiency and accuracy  
104 when classifying data. Eslamnezhad and Varjani [17] proposed a new detection algorithm to increase  
105 the quality of the clustering method based on a MinMax K-Means algorithm, overcoming the low  
106 sensitivity to initial centers in the k-Means algorithm. Ranjan and Sahoo [18] propose a modified  
107 K-medoids clustering algorithm for intrusion detection. The algorithm takes a new approach  
108 in selecting the initial medoids, overcoming the means in anomaly intrusion detection and the  
109 dependency on initial centroids, number of clusters, and irrelevant clusters.

110 Other authors have used hybrid solutions for log analysis, combining the use of the K-Means  
111 algorithm with other techniques for improving detection performance. They realized that, despite  
112 the inherent complex structure and high computational cost, hybrid classifiers can contribute to  
113 improve accuracy. Tokanju et al. [19], for instance, take advantage of an integrated signature-based  
114 and anomaly-based approach to propose a framework based on frequent patterns. Asif-Iqbal et al.  
115 [20] correlate different logs from different sources, supported by clustering techniques, to identify and  
116 remove unneeded logs. Hajamydeen et al. [21] classify events in two different stages supported by  
117 the same clustering algorithm. Initially, it uses a filtering process to identify the abnormal events, and  
118 then apply it for detecting anomalies. Varuna and Natesan [22] introduced a new hybrid learning  
119 method integrating K-Means clustering and Naive Bayes classification. Muda et al. [23] propose  
120 K-Means clustering and Naive Bayes classifiers in a hybrid learning approach, by using the KDD  
121 Cup'99 benchmark dataset for validation. In their approach, instances are separated into potential  
122 attacks and normal clusters. Subsequently, they are further classified into more specific categories.  
123 Elbasiony et al. [24] use data-mining techniques to build a hybrid framework for identifying network  
124 misuse and detecting intrusions. They used the random forests algorithm to detect misuses, with

125 K-Means as the clustering algorithm for unsupervised anomaly detection. The hybrid approach is  
126 achieved by combining the random forests algorithm with the weighted K-Means algorithm.

127 Some research focused on detecting which outliers constitute an anomaly, when applying  
128 clustering methods [25,26]. Liao and Vemuri [26] compute the membership of data points to a given  
129 cluster, supported by the use of Euclidean distance. Breunig et al. [27] state that some detection  
130 proposals weights data point as outliers.

131 Hybrid approaches have indeed proven quite interesting. However, in general proposed solutions  
132 still take considerable amounts of time to generate models for particular datasets, aggravated by the  
133 growth patterns normally associated with log sources in production systems. This situation calls for  
134 alternative strategies, able to improve speed (as well as accuracy and efficiency) by taking advantage  
135 of innovative algorithmic approaches, together with improved parallelism.

136 Our work focuses on scalability and interpretability, since the aim is to use it in the forensics and  
137 audit compliance contexts already discussed in Section 1. The goal is to be able to sift through data to  
138 select candidates for a more detailed analysis/inspection.

139 Similar to other works, we also take a hybrid approach for identifying anomalies for log analysis.  
140 However, unlike other works, we specifically target speed, agility and interpretability. Our approach  
141 allows training and classifying out of core datasets in scenarios involving the computation of very large  
142 datasets with limited computing resources, parallelizing their processing by distributing them across  
143 the available nodes. Therefore, our approach is supported by clustering and classification algorithms  
144 that are able to scale and produce interpretable results. Our method works in two stages: first, it starts  
145 with the unlabelled dataset, implementing a binary anomalous event classifier, through the use of  
146 unsupervised learning algorithms. The second stage produces a set of simple rules by considering  
147 the previously classified data, through the use of supervised learning algorithms. It combines the  
148 K-Means algorithm for clustering anomalies and the Gradient Tree Boosting to produce a simple set of  
149 interpretable rules to be parallelized in a distributed environment on classifying a large amount of  
150 data.

151 Next, we present in more detail the already existing techniques used by our approach.

### 152 2.3. K-Means

153 K-Means remains as one of the most popular clustering methods and one of the most relevant  
154 algorithms in data mining [3]. The advantage of K-Means is its simplicity. By starting with a set of  
155 randomly chosen initial centers, one procedure assigns each input point to its nearest center and then  
156 recomputes the centers given the point assignment [28].

157 Scaling K-Means to massive data is relatively easy, due to its simple iterative nature. Given a  
158 set of cluster centers, each point can independently decide which center is closest to it and, given an  
159 assignment of points to clusters, computing the optimum center can be performed by simply averaging  
160 the points. Indeed, parallel implementations of k-Means are readily available [28].

161 From a theoretical standpoint, K-Means is not a good clustering algorithm in terms of efficiency  
162 or quality. Thus, the running time can grow exponentially in the worst case [29,30] and even though  
163 the final solution is locally optimal, it can be very far away from the global optimum (even under  
164 repeated random initializations). Nevertheless, in practice, the speed and simplicity of K-Means are  
165 attractive. Therefore, recent work has focused on improving its initialization procedure performance  
166 in terms of quality and convergence [28].

### 167 2.4. Decision Trees

168 Decision Trees is a popular supervised machine learning method that produces regression or  
169 classification models in the form of a tree structure containing decisions as nodes, resulting in a set  
170 of leaves containing the solution. Decision trees are suitable to be applied to any data without much  
171 effort, when compared with algorithms such as neural networks. Trees are built top-down from  
172 the root node and involve recursively binary splitting. The initial dataset is partitioned into smaller

173 subsets according to their features, while an associated decision tree is incrementally built. Such a  
174 splitting process is driven by a greedy algorithm evaluating the best solution at each of those steps and  
175 evaluating the maximum loss reduction from the cost function, in order to make a split on features.  
176 To regulate the complexity of a given model and increase the performance of a given tree, pruning  
177 processes are available. Notwithstanding, the performance of decision tree learning does not generally  
178 provide the best performance in terms of prediction. Some approaches exist in learning decision forests,  
179 including bagging [31], random forests [32] and boosted trees [33].

180 Tree boosting overcomes the above performance problem by the use of an additive model that  
181 iteratively builds decision trees to learn decision forests by applying a greedy algorithm (boosting) on  
182 top of a decision tree base learner [34], [33], [35]. Tree boosting is regarded as one the most effective  
183 off-the-shelf nonlinear learning methods for a wide range of application problems [34]. It is also highly  
184 effective and widely used for achieving state-of-the-art results on many machine learning challenges  
185 hosted by the machine learning competition site Kaggle [36].

186 Regularized Greedy Forest is an algorithm that can handle general loss functions to a wider range  
187 of applicability that directly learns decision forests taking advantage of the tree structure itself, while  
188 other methods employ specific loss functions, such as exponential loss function in the case of the  
189 Adaboost algorithm [34].

## 190 2.5. XGBoost

191 XGBoost is a scalable system that implements the Gradient Tree Boosting and the regularized  
192 model – to prevent overfitting – and simplifies the objective function – for parallelization of regularized  
193 greedy forest algorithm [34]. It is suitable for the development of parallel computing solutions  
194 applicable to larger datasets or faster training. Besides processors and memory, it uses disk space  
195 to handle data that does not fit into main memory. To enable out of core computation, the data is  
196 divided into multiple blocks [5]. It includes cache access patterns, data compression, and sharding. Its  
197 performance relies on a tree learning algorithm, that is able to handle sparse data, and in a weighted  
198 quantile sketch procedure. This procedure enables handling instance weights in approximate tree  
199 learning able to solve real-world scale problems using a minimal amount of resources. Besides  
200 the penalty from regularizing the objective function, two techniques prevent overfitting: shrinkage,  
201 introduced by Friedman [37], and feature subsampling retrieved from Random Forests to speed up  
202 computations. XGBoost works well in practice and has won several machine learning competitions,  
203 such as Kaggle [36], running faster than other popular solutions on a single machine and scaling in  
204 distributed or out of core settings. It can be easily interpreted, given the tools it provides for finding  
205 the important features from the XGBoost model.

## 206 2.6. Dask

207 The Dask parallel computing framework leverages the existing Python ecosystem, including  
208 relevant libraries such as "numpy" or "pandas". Dask capabilities are supported by executing graphs to  
209 be run by the scheduler component, potentially scaling execution to millions of nodes. Those features  
210 are suitable to be applied to out of core scenarios (not fitting in memory) on a single machine [6].

211 Dask is a Python specification representing the computation of directed acyclic graphs of tasks  
212 with data dependencies, to encode parallel task graph schedules. It extends the easy to adopt NumPy  
213 library for leveraging parallel computation over modern hardware. It allows to scale large datasets  
214 by using disks that extend the physical memory as out of core and parallelize and linearly speedup  
215 the code taking advantage of several cores. The main objective is to parallelize the existing Python  
216 software stack without triggering a full rewrite. A Dask cluster includes a central scheduler and several  
217 distributed workers. It starts up a XGBoost scheduler and a XGBoost worker within each of the Dask  
218 workers sharing the same physical processes and memory spaces.

219 Dask enables parallel and out of core computation by including collections such as arrays, bags,  
220 and dataframes. It couples blocked algorithms with dynamic and memory-aware task scheduling

221 to achieve a parallel and out of core popular NumPy clone [6]. Sharing distributed processes with  
222 multiple systems allow the usage of specialized services easily and avoid large monolithic frameworks.  
223 Dask is often compared with other distributed machine learning libraries, such as H2O [38] or  
224 Spark's Machine Learning Library (MLLib) [39]. XGBoost is available in Dask to provide users with  
225 a fully featured and efficient solution. The Dask parallel computing approach can handle problems  
226 more complicated than map-reduce at a lower cost and complexity, when compared to solutions such  
227 as MLLib, given that most of the problems can be resolved in a single machine. Any function is able  
228 to be parallelized by the use of delayed functions decorators. Also, Dask is substantially lightweight  
229 when compared to Spark.

### 230 3. Proposed Framework

231 Motivated by the related work, we propose an integrated method with filtering mechanisms to  
232 improve detection accuracy and efficiency in scenarios involving large amounts of logs. This method  
233 is supported by the K-Means clustering and the Gradient Tree Boosting Classification algorithms,  
234 as implemented by the XGBoost system. To overcome the limitations of existing anomaly detection  
235 methods that spend a significant amount of time building the models for the whole dataset, we built  
236 three different tools for improving detection accuracy and efficiency.

237 This section starts with a formal presentation of the algorithm of the model, followed by the  
238 discussion of the three compounding tools used for implementing the proposed approach.

#### 239 3.1. Description of the Algorithm

240 The proposed approach is formalized in Algorithm 1, which describes how to combine K-Means  
241 and XGBoost. The algorithm is implemented as a function that takes as input a set of events  $E$  and  
242 returns the identification of the anomaly  $anomalycluster$ , the classified events  $ypred^1$ , total classified  
243 events  $totalevents$ , and the total of those events classified as anomalies  $totalanomalies$ .

244 It starts by initializing the cluster  $S$  and activating the client connection  $C$  to the cluster  $S$ .  
245 Following, the distributed array  $G$  is prepared from the received events in  $E$ . The next step is to  
246 initialize the K-Means model  $Km$  for binary classification in the cluster ( $k = 2$ ) from the distributed  
247 array  $G$  to separate events in two distinct clusters in  $Y$ . Then, the XGBoost model  $X$  is initialized with  
248 the previously predicted events  $Y$  being provided as an input for training in the cluster through the  
249 use of the client connection  $C$ . The final prediction  $ypred$  is achieved from the XGBoost model  $X$ . In  
250 the next stage, each of those predictions ( $i \in ypred$ ) is classified according to the cluster they belong to  
251 in  $ypred^1$ . Such classification will be determined by evaluating the total number of events in clusters  
252  $k1$  and  $k2$ , to decide which corresponds to the anomaly cluster. To that aim, 0.5 was considered as the  
253 threshold to classify events as belonging the cluster 1 or cluster 2 ( $ypred_i > 0.5$ ).

254 After all events have been classified, the cluster including the fewer number of events ( $k1 > k2$ )  
255 will correspond to the anomaly cluster, and such decision will stored in  $anomalycluster$ .

#### 256 3.2. Tools

257 The framework encompasses three tools that may be independently combined in a cooperative  
258 way for normalizing raw data and to produce a model able to achieve evidence for forensic and  
259 compliance auditing analysis. The 'fca\_normalization' tool is used to normalize the raw data,  
260 'fca\_model' produces the model, and 'fca\_analysis' provides the pieces of evidence for forensic and  
261 compliance auditing analysis.

262 The normalization tool takes as input HTTP raw data logs and normalizes data into a new file.  
263 Optionally, the encoded features may also be specified. In the case encoding is not provided or in  
264 the case of missing feature values, the tool automatically applies an encoding label. The tool can be  
265 invoked, for example, by using the following command:

```
266 python fca_normalization  
267     -in NASA_access_log_Jul95
```

**Algorithm 1** Proposed Algorithm**INPUT:**  $E$ , Event Set

```

 $S \leftarrow \text{CLUSTER}()$ 
 $C \leftarrow \text{CLIENT}(S)$ 
 $G \leftarrow \text{DISTRIBUTEDARRAY}(E)$ 
 $k \leftarrow 2$ 
 $km \leftarrow \text{KMEANS}(C, k)$ 
 $km.\text{TRAIN}(G)$ 
 $Y \leftarrow km.\text{PREDICT}(G)$ 
 $X \leftarrow \text{XGBOOST}(X)$ 
 $X.\text{TRAIN}(Y, Y)$ 
 $ypred \leftarrow X.\text{PREDICT}(G)$ 
for all  $i \in ypred$  do
    if  $ypred_i > 0.5$  then
         $ypred^1_i \leftarrow 1$ 
         $k2 \leftarrow k2 + 1$ 
    else
         $ypred^1_i \leftarrow 0$ 
         $k1 \leftarrow k1 + 1$ 
    end if
end for
if  $k1 > k2$  then
     $anomalycluster \leftarrow 1$ 
     $totalanomalies \leftarrow k2$ 
else
     $anomalycluster \leftarrow 0$ 
     $totalanomalies \leftarrow k1$ 
end if
 $totalevents \leftarrow k1 + k2$ 

```

**OUTPUT:**  $ypred^1$ , Cluster Predictions**OUTPUT:**  $anomalycluster$ , Identification of the anomaly cluster**OUTPUT:**  $totalevents$ , Total number of events**OUTPUT:**  $totalanomalies$ , Total number o anomalies

```

268      -in_encoding in_encoding.data
269      -out logs_NASA.csv
270      -out_encoding out_encoding.data

```

271 In this example 'fca\_normalization' receives the raw HTTP log data file 'NASA\_access\_log\_Jul95',  
 272 along with the optional encoding file 'encoding.data'. The output normalized file is saved as  
 273 'logs\_NASA.csv'. Finally, the tool optionally defines the encoding table in file 'out\_encoding.data'.

274 The modeling tool takes as input the previously normalized data and builds the XGBoost  
 275 classification model by making use of the Gradient Tree Boosting algorithm after applying the K-Means  
 276 clustering algorithm. In the example invocation provided next, the input file 'logs\_NASA.csv' contains  
 277 the HTTP raw log data and the output model is saved as 'fca\_xgboost.pkl'.

```

278 python fca_model
279     -in logs_NASA.csv
280     -out fca_xgboost.pkl

```

281 The forensic and compliance auditing analysis tool takes as input the model and the normalized  
 282 events in order to identify the anomalies. In the invocation example provided next, the input model is  
 283 read from 'fca\_xgboost.pkl', and the normalized data is read from 'logs\_NASA.csv'. The final output  
 284 containing the anomaly events is saved on 'outlier\_events.csv'.

```

285 python fca_analysis
286     -in_model fca_xgboost.pkl
287     -in_data logs_NASA.csv
288     -out outlier_events.csv

```

**289** Table 1 summarizes the inputs and outputs for each tool.

**290**

Tool	Input	Output
Normalization	HTTP raw logs data, Encoding	Normalized data, Encoding
Modelling	Normalized data	Model
Analysis	Model, Normalized data	Anomaly Events

**Table 1.** Tools Inputs and Outputs

## **291** 4. Discussion and Evaluation

**292** This section addresses the validation of the proposed framework. First, we discuss feature  
**293** extraction. Next, based on the extracted features, we describe the initial application of the K-Means  
**294** clustering algorithm for dividing the dataset into two different clusters. Finally, we discuss how to  
**295** use the previous clustered data for training a scalable Gradient Tree Boosting implemented by the  
**296** XGBoost system.

**297** For sake of readability, along this section we extensively use as reference a set of well-known,  
**298** publicly available datasets [40]. These datasets consist of traces containing two month's worth of  
**299** all HTTP requests to the NASA Kennedy Space Center WWW server, involving 1,871,988 logged  
**300** events. This dataset was selected because it is probably the largest log-based dataset publicly available,  
**301** allowing us to assess our scalability claims.

### **302** 4.1. Feature Extraction and Data Exploration

**303** To capture the characteristics of the system behaviors, 14 features were extracted, containing both  
**304** numeric and categorical attributes from the raw log records. The original features in the raw HTTP  
**305** logs records are 'IP', 'Date', 'Request', 'Response' and 'length'. By making use of regular expressions  
**306** the most relevant time-related components are extracted from the 'date' feature, including the 'Day',  
**307** 'Month', 'Year', 'Hour', 'Minute' and 'Second'. From the 'Request' field the 'operation', 'page' and the  
**308** 'method' features were extracted. Then, 'Month' names are encoded. Therefore 'Year', 'Month' and  
**309** 'Day' was composed in the temporary 'date' feature in order to retrieve the day of the week ('weekday')  
**310** and 'weekend' features. Next, 'Request' other temporary features were removed from the dataset.  
**311** Finally, categorical features such as 'IP', 'page', 'operation', 'method' and 'Response' were encoded,  
**312** and the dataset was saved in a file.

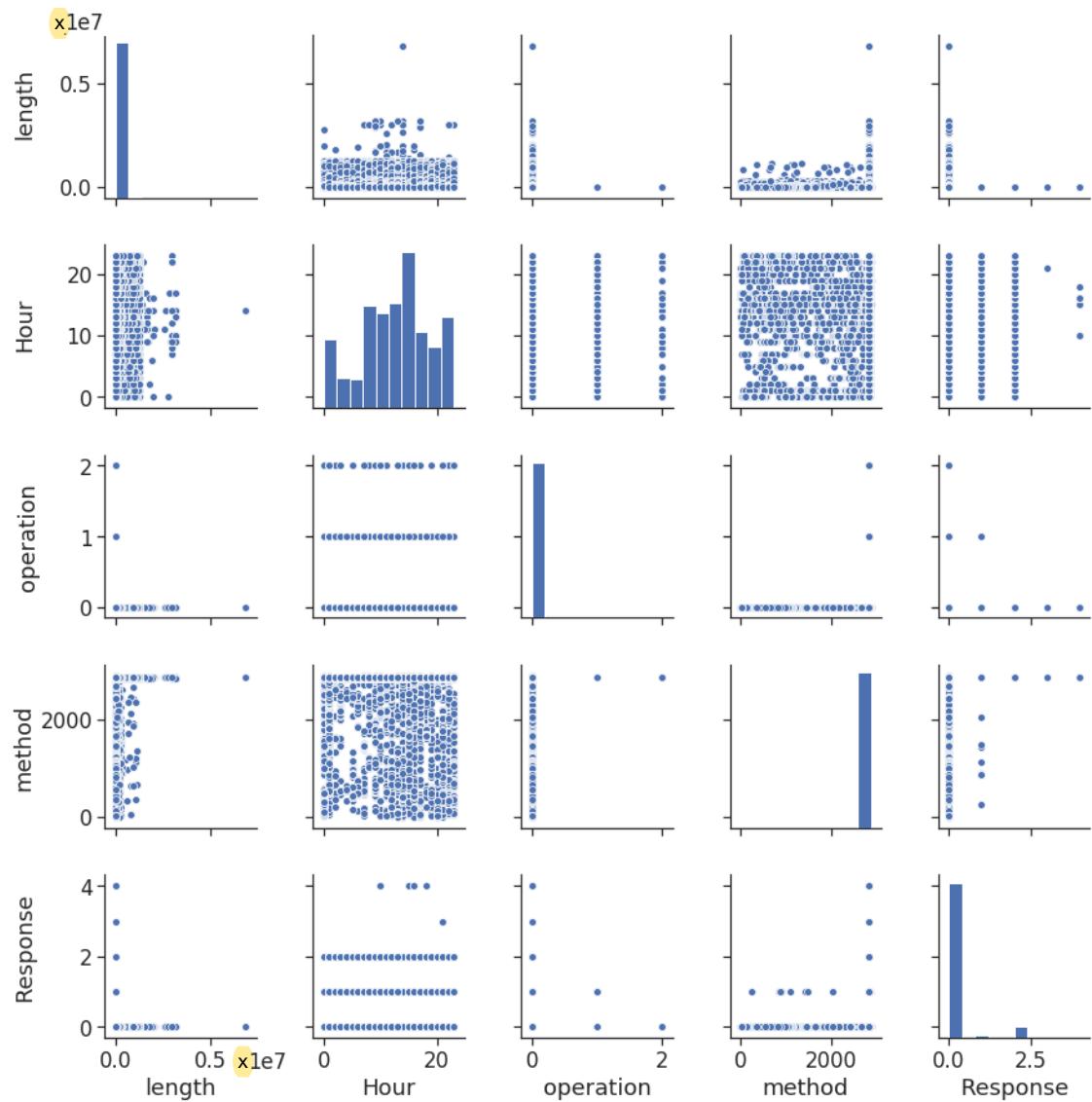
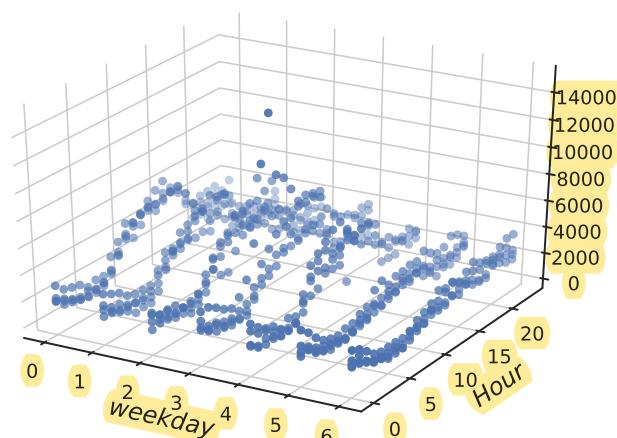
**313** By exploring the dataset we can achieve the first insights. Figure 1 depicts the covariance of the  
**314** most representative features, including 'length', 'Hour', 'operation', 'method' and 'Response'. This  
**315** figure shows interesting covariance between length and other features.

**316** Figure 2 provides a three-dimensional analysis of the number of events occurred along the day  
**317** (from 0 to 24 hours) and along each weekday (0 to 6), where days 5 and 6 correspond to Saturday and  
**318** Sunday respectively.

### **319** 4.2. Clustering

**320** Based on the extracted features, we employ the K-Means clustering algorithm for grouping log  
**321** events into two different clusters. The larger cluster gathers the normal events, while the smaller  
**322** holds the deviations from normal behavior. Therefore, the latter cluster should correspond to the set  
**323** grouping the anomaly events. In addition, sparse clusters are possibly caused by anomalous activities,  
**324** which can be labeled as anomaly candidates for further analysis.

**325** Our framework model takes advantage of the initialization k-means|| algorithm (largely inspired  
**326** by K-Means++) to obtain a nearly optimal solution after a logarithmic number of steps. In practice, a  
**327** constant number of passes suffices [28].

**Figure 1.** Features covariance**Figure 2.** HTTP events over Time

328 After training this model with 90% of the total number of records and using just the remaining  
 329 10% for testing, the model produces a normal cluster containing 185,897 events while the anomaly  
 330 cluster includes 1301 events, corresponding to 0,06% of total number of events taking part of the  
 331 normal cluster.

332 The computed centroids for the two clusters, separating the normal and anomaly events, are the  
 333 following:

```
334 [[4.41534608e+04, 0.0000000e+00, 8.12115012e+05, 1.14495884e+01,  

  335 0.0000000e+00, 0.0000000e+00, 1.26692042e+01, 2.96762857e+01,  

  336 2.94179871e+01, 0.0000000e+00, 1.75010380e+04, 2.84859910e+03,  

  337 2.82322741e+00, 2.23245109e-01]  

  338  

  339 [4.27877328e+04, 1.63161125e-01, 1.53047043e+04, 1.24323538e+01,  

  340 0.0000000e+00, 0.0000000e+00, 1.26856431e+01, 2.95910303e+01,  

  341 2.94991093e+01, 2.22648078e-03, 1.47567972e+04, 2.84883391e+03,  

  342 2.68136168e+00, 1.93607622e-01]]
```

#### 343 4.3. Classification

344 Classification results from the application of the Gradient Tree Boosting algorithm implemented  
 345 by the XGBoost system, which is the second and final stage of our model. The resulting tree can be  
 346 linearized into decision rules, where the outcome is the content of the leaf node, and the conditions  
 347 along the path form a conjunction in the if clause.

348 The results of this stage were validated by comparing if the number of events classified as  
 349 anomalies is equal to the number of events belonging to the anomaly cluster. This condition was  
 350 verified for 1301 events. The predict function for XGBoost outputs probabilities by default and not  
 351 actual class labels. To calculate accuracy we converted them to 0 and 1 labels where 0.5 probability  
 352 corresponds to the threshold. XGBoost is able to correctly classify all the test data according to the  
 353 K-Means clustering algorithm. Figure 3 depicts the importance of the XGBoost features, according to  
 354 the F-Score metric.

355 This classification model produces a set of rules providing the rationale for generalizing to unseen  
 356 events, as shown in Figure 4. The leaf values depicted in the figure are converted into probabilities by  
 357 applying the logistic function.

358 Figure 5 depicts the 'length' and 'page' covariance, which are the two most important features  
 359 computed by the final model. The events tagged as anomalous are highlighted in red color.

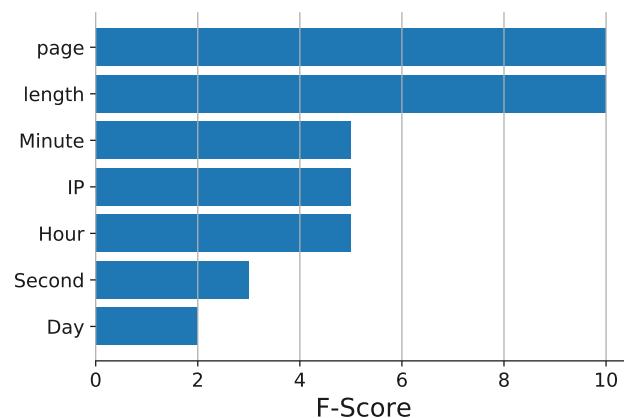
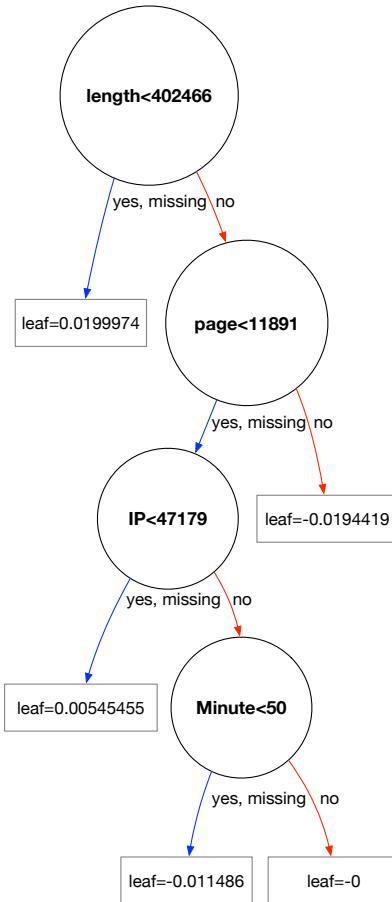
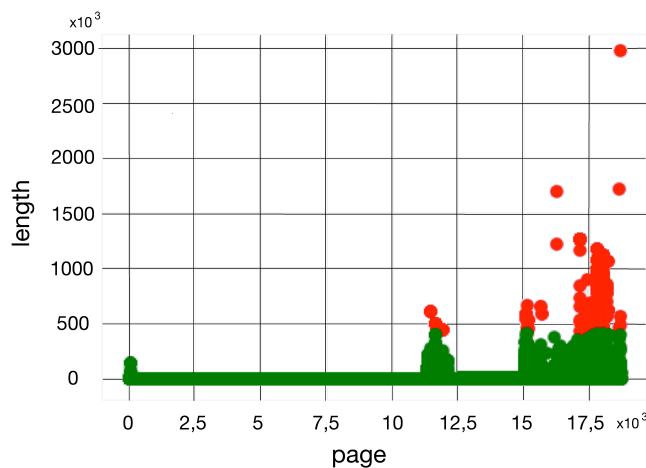


Figure 3. Features Importance

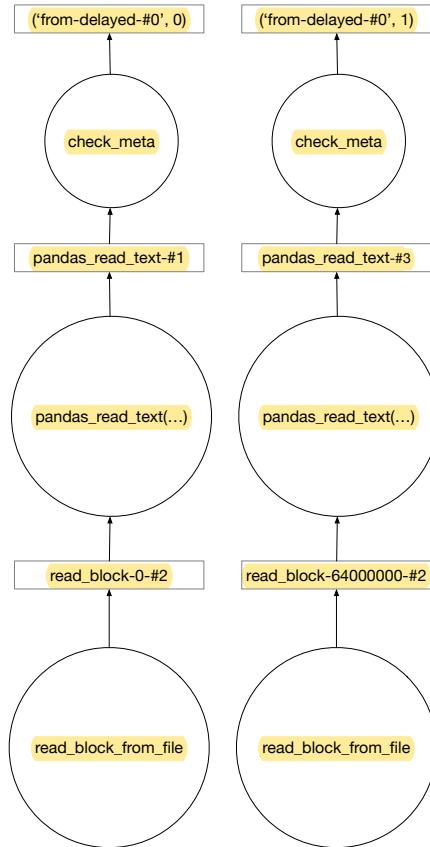
**Figure 4.** Decision Tree**Figure 5.** Page and Length Covariance

#### **360 4.4. Parallelization**

**361** The proposed framework makes use of the K-Means algorithm and the XGBoost system, which are  
**362** designed to scale in a distributed environment supported by available parallel computing capabilities.  
**363** Such an approach comes in line with a Big Data scenario.

**364** Our approach is supported by the use of parallel computing capabilities available in the Python  
**365** 'Dask' library. More specifically, the 'dataframe' component is able to manage out of core datasets along  
**366** the execution pipeline, since the features are extracted until the clustering and classification models are  
**367** implemented. Figure 6 provides an example of the kind of graphs Dask is able to produce when reading

**368** and splitting a dataset. Dask libraries 'dask\_ml' and 'dask\_xgboost' provide the implementation of  
**369** popular machine learning algorithms such as K-Means and XGBoost, which support the framework  
**370** models.



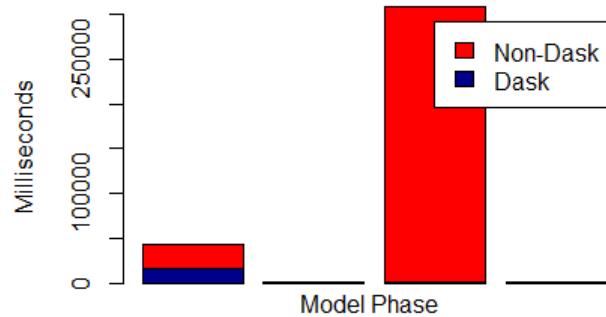
**Figure 6.** Parallelized Dask Graphs

**371** Our experiment involved a simple cluster formed by just two workers in a single one node with  
**372** two cores while the total available memory was 13.66 GB.

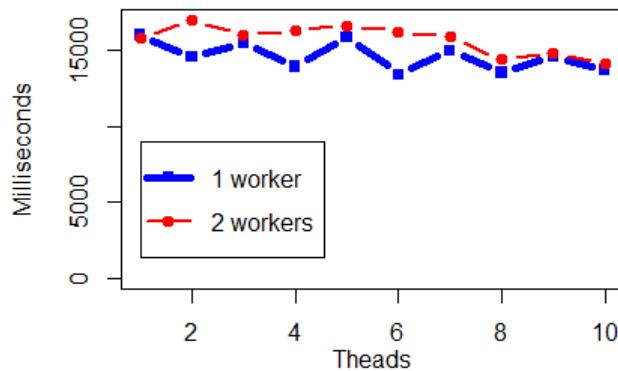
**373** To study the framework model's ability to scale, in order to cope with large datasets in a reasonable  
**374** time, two experiments were performed using the parsed NASA HTTP logs dataset. Due to constrained  
**375** laboratory resources, those experiments were limited to the use of two cores in a single node. As  
**376** a setup configuration, the Dask chunk size was set to 50,000 events. The model's ability to scale  
**377** was assessed by comparing its performance under different configurations. To determine the model  
**378** performance, running time (in milliseconds) was considered, throughout the training and predict steps  
**379** for both K-Means and XGBoost stages, in accordance with the model topology.

**380** A first experiment aimed at determining the parallel approach performance, compared with  
**381** the sequential approach – considering non-Dask sklearn as the sequential approach and Dask as the  
**382** parallel approach. As setup configuration, Dask framework included a single worker and two threads.  
**383** The running time was measured along the four steps previously defined for the two stages. Those  
**384** sequential steps include train (1) and predict (2) for the K-Means stage, followed by the train (3) and  
**385** predict (4) for XGBoost stage. The running time for each framework, along those running steps, is  
**386** provided in Figure 7. Achieved results show that the Dask framework outperforms non-Dask sklearn  
**387** framework, especially in the case of the training step.

**388** A second experiment evaluated the parallelization capability of the Dask framework under  
**389** different configurations, such as the number of workers and threads per worker, by measuring  
**390** the aggregated running time along the topological steps. Figure 8 compares the performance for  
**391** one and two running workers, while increasing the number of threads per worker from one to ten.

**Figure 7.** Sequential (Non-Dask) vs Parallel (Dask) Comparison

<sup>392</sup> Measurements show that one worker outperforms two workers. Increasing the number of workers did  
<sup>393</sup> not improve performance, while increasing the number of threads contributes to improve performance  
<sup>394</sup> until a given threshold is reached. Finally, it was also possible to depict higher performance running  
<sup>395</sup> over a even number of threads in comparison to the odd ones – due to the less optimal parallelization  
<sup>396</sup> gains that occur when splitting an odd number of threads by two cores.

**Figure 8.** Dask Parallel Comparison

#### <sup>397</sup> 4.5. Discussion

<sup>398</sup> The presented framework method relies in two stages. The clustering model is the output of the  
<sup>399</sup> first stage and serves as the input for the classification stage. Therefore, this approach allows starting  
<sup>400</sup> from initial unlabelled data for obtaining the interpretable meaningful rules with the rationale for  
<sup>401</sup> classifying unseen events. Those rules are simple to understand, interpret and visualize, requiring  
<sup>402</sup> relatively little effort in data preparation. Additionally, the described algorithms can easily handle  
<sup>403</sup> heterogeneous data containing different features produced by different sources. Although the initial  
<sup>404</sup> nature of our problem is not a classification problem, this approach may be adapted to different  
<sup>405</sup> scenarios where labeled data is not available. This way, it becomes possible to convert an unsupervised  
<sup>406</sup> into a supervised learning scenario and take advantage of the use of classification algorithms.

<sup>407</sup> The decision to select the K-Means algorithm and XGBoost system, both supported by the Dask  
<sup>408</sup> library for parallel processing, was driven by requirements in terms of ability to scale, interpretability,  
<sup>409</sup> when working with limited resources. This decision enabled the application of this framework to  
<sup>410</sup> larger datasets in order to highlight the anomalous events. Given the inherent nature of the problem  
<sup>411</sup> being addressed through the use of the unsupervised learning approach, it is not trivial to evaluate the  
<sup>412</sup> framework model's accuracy in the scope of this paper. An alternative option would be to compare  
<sup>413</sup> the achieved results with those provided in existing literature. However, to the best of our knowledge,  
<sup>414</sup> there are no anomaly detection research works addressing the NASA HTTP logs.

<sup>415</sup> The obtained results highlight the obviously normal events in highly coherent clusters, with a  
<sup>416</sup> minor subset of events being classified as anomalies, for further forensic and compliance auditing

analysis. The model interpretability is indirectly validated by the produced decision rule set already provided in Figure 4, which implicitly shows how the model identifies classes. Figure 7 also shows the performance of the parallel approach, compared with the sequential approach, and Figure 8 highlights the parallelization capabilities of the Dask library in processing out of core datasets.

Designing the framework with independent tools makes it possible to reuse them over different scenarios. For example, the same modeling tool can be combined with a different normalization tool for processing a different data source. Additionally, these framework tools are able to be applied to the context of the aforementioned ATENA project in order to identify anomaly events from massive logs. This approach is suitable to be independently applied to different datasets in a first stage, allowing to correlate them as heterogeneous sources in a second stage.

Achieved results demonstrate the capability of the proposed method in terms of finding a set of interpretable rules, which able to be parallelized and applied in scale.

## 5. Conclusions and Future Work

In this paper we propose a framework that takes a parallel computing approach for identifying anomaly events in massive log files. In a first stage, our method uses the K-Means algorithm to separate anomalies from normal events. In a second stage, a Gradient Tree Boosting classification model, implemented using the XGBoost system, produces the interpretable meaningful rationale rule set for generalizing its application to a massive number of unseen events. This approach is suitable for application in the context of out of core datasets in use cases where log sources are so massive that it becomes impossible to use more traditional approaches.

The proposed method was presented and the achieved results demonstrated its applicability on producing simple and interpretable rules to highlight anomalies in application logs data to scale in a distributed environment. Such an approach makes it suitable to be applied in the fields of forensics and audit compliance.

Regarding future work, we plan to explore the application of collective anomaly detection over time series summarized data logs and the application of Bayesian networks as the classification model component and evaluate its capability of producing scalable and interpretable models. We also plan to explore the map-reduce model as the way to achieve higher parallelism performance on data preparation.

**Funding:** This work is partially funded by National Funds through the FCT - Foundation for Science and Technology, I.P., and the European Social Fund, through the Regional Operational Program Centro 2020, within the scope of the projects UIDB/05583/2020 and CISUC UID/CEC/00326/2020. Furthermore, we would like to thank the Research Centre in Digital Services (CISeD) and the Polytechnic of Viseu for their support.

**Conflicts of Interest:** The authors declared that we have no conflicts of interest to this work.

451

- 452 1. Adamsky, F.; Aubigny, M.; Battisti, F.; Carli, M.; Cimorelli, F.; Cruz, T.; Giorgio, A.D.; Foglietta, C.; Galli, A.; Giuseppi, A.; Liberati, F.; Neri, A.; Panzieri, S.; Pascucci, F.; Proenca, J.; Pucci, P.; Rosa, L.; Soua, R. Integrated protection of industrial control systems from cyber-attacks: the ATENA approach. *International Journal of Critical Infrastructure Protection* **2018**, *21*, 72–82. doi:<https://doi.org/10.1016/j.ijcip.2018.04.004>.
- 453 2. Rosa, L.; Proença, J.; Henriques, J.; Graveto, V.; Cruz, T.; Simões, P.; Caldeira, F.; Monteiro, E. An Evolved Security Architecture for Distributed Industrial Automation and Control Systems. European Conference on Cyber Warfare and Security. Academic Conferences International Limited, 2017, pp. 380–390.
- 454 3. Wu, X.; Kumar, V.; Quinlan, J.R.; Ghosh, J.; Yang, Q.; Motoda, H.; McLachlan, G.J.; Ng, A.; Liu, B.; Philip, S.Y.; others. Top 10 algorithms in data mining. *Knowledge and information systems* **2008**, *14*, 1–37.
- 455 4. Friedman, J.H. Greedy function approximation: a gradient boosting machine. *Annals of statistics* **2001**, pp. 1189–1232.
- 456 5. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

- 465 6. Rocklin, M. Dask: Parallel computation with blocked algorithms and task scheduling. Proceedings of the  
466 14th Python in Science Conference. Citeseer, 2015, number 130-136.
- 467 7. Grubbs, F.E. Procedures for detecting outlying observations in samples. *Technometrics* **1969**, *11*, 1–21.
- 468 8. Gogoi, P.; Bhattacharyya, D.; Borah, B.; Kalita, J.K. A survey of outlier detection methods in network  
469 anomaly identification. *The Computer Journal* **2011**, *54*, 570–588.
- 470 9. Zheng, Y.; Zhang, H.; Yu, Y. Detecting collective anomalies from multiple spatio-temporal datasets  
471 across different domains. Proceedings of the 23rd SIGSPATIAL international conference on advances in  
472 geographic information systems, 2015, pp. 1–10.
- 473 10. Yang, Q.; Wu, X. 10 challenging problems in data mining research. *International Journal of Information  
474 Technology & Decision Making* **2006**, *5*, 597–604.
- 475 11. Chen, Z.; Li, Y.F. Anomaly detection based on enhanced DBScan algorithm. *Procedia Engineering* **2011**,  
476 *15*, 178–182.
- 477 12. Syarif, I.; Prugel-Bennett, A.; Wills, G. Unsupervised clustering approach for network anomaly detection.  
478 International Conference on Networked Digital Technologies. Springer, 2012, pp. 135–145.
- 479 13. Hoglund, A.J.; Hatonen, K.; Sorvari, A.S. A computer host-based user anomaly detection system using  
480 the self-organizing map. *Neural Networks*, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS  
481 International Joint Conference on. IEEE, 2000, Vol. 5, pp. 411–416.
- 482 14. Lichodzijewski, P.; Zincir-Heywood, A.N.; Heywood, M.I. Host-based intrusion detection using  
483 self-organizing maps. *Neural Networks*, 2002. IJCNN'02. Proceedings of the 2002 International Joint  
484 Conference on. IEEE, 2002, Vol. 2, pp. 1714–1719.
- 485 15. Münz, G.; Li, S.; Carle, G. Traffic anomaly detection using k-means clustering. GI/ITG Workshop MMBnet,  
486 2007, pp. 13–14.
- 487 16. Tian, L.; Jianwen, W. Research on network intrusion detection system based on improved k-means  
488 clustering algorithm. *Computer Science-Technology and Applications*, 2009. IFCSTA'09. International  
489 Forum on. IEEE, 2009, Vol. 1, pp. 76–79.
- 490 17. Eslamnezhad, M.; Varjani, A.Y. Intrusion detection based on MinMax K-means clustering.  
491 Telecommunications (IST), 2014 7th International Symposium on. IEEE, 2014, pp. 804–808.
- 492 18. Ranjan, R.; Sahoo, G. A new clustering approach for anomaly intrusion detection. *arXiv preprint  
493 arXiv:1404.2772* **2014**.
- 494 19. Makanju, A.; Zincir-Heywood, A.N.; Milios, E.E. Investigating event log analysis with minimum apriori  
495 information. Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on.  
496 IEEE, 2013, pp. 962–968.
- 497 20. Asif-Iqbal, H.; Udzir, N.I.; Mahmud, R.; Ghani, A.A.A. Filtering events using clustering in heterogeneous  
498 security logs. *Information Technology Journal* **2011**, *10*, 798–806.
- 499 21. Hajamydeen, A.I.; Udzir, N.I.; Mahmud, R.; GHANI, A.A.A. An unsupervised heterogeneous log-based  
500 framework for anomaly detection. *Turkish Journal of Electrical Engineering & Computer Sciences* **2016**,  
501 *24*, 1117–1134.
- 502 22. Varuna, S.; Natesan, P. An integration of k-means clustering and naïve bayes classifier for Intrusion  
503 Detection. Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference  
504 on. IEEE, 2015, pp. 1–5.
- 505 23. Muda, Z.; Yassin, W.; Sulaiman, M.; Udzir, N. K-means clustering and naive bayes classification for  
506 intrusion detection. *Journal of IT in Asia* **2016**, *4*, 13–25.
- 507 24. Elbasiony, R.M.; Sallam, E.A.; Eltobely, T.E.; Fahmy, M.M. A hybrid network intrusion detection framework  
508 based on random forests and weighted k-means. *Ain Shams Engineering Journal* **2013**, *4*, 753–762.
- 509 25. Sequeira, K.; Zaki, M. ADMIT: anomaly-based data mining for intrusions. Proceedings of the eighth ACM  
510 SIGKDD international conference on Knowledge discovery and data mining. ACM, 2002, pp. 386–395.
- 511 26. Liao, Y.; Vemuri, V.R. Use of k-nearest neighbor classifier for intrusion detection. *Computers & security*  
512 **2002**, *21*, 439–448.
- 513 27. Breunig, M.M.; Kriegel, H.P.; Ng, R.T.; Sander, J. LOF: identifying density-based local outliers. ACM  
514 sigmod record. ACM, 2000, Vol. 29, pp. 93–104.
- 515 28. Bahmani, B.; Moseley, B.; Vattani, A.; Kumar, R.; Vassilvitskii, S. Scalable k-means++. *Proceedings of the  
516 VLDB Endowment* **2012**, *5*, 622–633.

- 517 29. Vattani, A. K-means requires exponentially many iterations even in the plane. *Discrete & Computational  
518 Geometry* **2011**, *45*, 596–616.
- 519 30. Arthur, D.; Vassilvitskii, S. How slow is the k-means method? Proceedings of the twenty-second annual  
520 symposium on Computational geometry. ACM, 2006, pp. 144–153.
- 521 31. Breiman, L. Bagging predictors. *Machine learning* **1996**, *24*, 123–140.
- 522 32. Breiman, L. Random forests. *Machine learning* **2001**, *45*, 5–32.
- 523 33. Friedman, J.; Hastie, T.; Tibshirani, R.; others. Additive logistic regression: a statistical view of boosting  
524 (with discussion and a rejoinder by the authors. *The annals of statistics* **2000**, *28*, 337–407.
- 525 34. Johnson, R.; Zhang, T. Learning nonlinear functions using regularized greedy forest. *IEEE transactions on  
526 pattern analysis and machine intelligence* **2014**, *36*, 942–954.
- 527 35. He, X.; Pan, J.; Jin, O.; Xu, T.; Liu, B.; Xu, T.; Shi, Y.; Atallah, A.; Herbrich, R.; Bowers, S.; others. Practical  
528 lessons from predicting clicks on ads at facebook. Proceedings of the Eighth International Workshop on  
529 Data Mining for Online Advertising. ACM, 2014, pp. 1–9.
- 530 36. Kaggle. Kaggle. [www.Kaggle.com](http://www.Kaggle.com), 2018. visited on 2018-07-16.
- 531 37. Friedman, J.H. Stochastic gradient boosting. *Computational Statistics & Data Analysis* **2002**, *38*, 367–378.
- 532 38. H2O.ai. H2O Framework for Machine Learning. <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html>, 2018. visited on 2020-02-15.
- 533 39. Meng, X.; Bradley, J.; Yavuz, B.; Sparks, E.; Venkataraman, S.; Liu, D.; Freeman, J.; Tsai, D.; Amde, M.;  
534 Owen, S.; et al.. MLlib: Machine Learning in Apache Spark. *J. Mach. Learn. Res.* **2016**, *17*, 1235–1241.
- 535 40. NASA. NASA HTTP. <http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html>, 2018. visited on 2018-07-01.

537 © 2020 by the authors. Submitted to *Electronics* for possible open access publication  
538 under the terms and conditions of the Creative Commons Attribution (CC BY) license  
539 (<http://creativecommons.org/licenses/by/4.0/>).