A Comparative Study of Unsupervised Anomaly Detection Algorithms used in a Small and Medium-Sized Enterprise

Irina Petrariu[1], Adrian Moscaliuc[2], Cristina Elena Turcu[3], Ovidiu Gherman[4]
ASSIST Software SRL, Suceava, Romania[1, 2]
Stefan cel Mare University, Suceava, Romania[3, 4]

***Abstract*—Anomaly detection finds application in several industries and domains. The anomaly detection market is growing driven by the increasing development and dynamic adoption of emerging technologies. Depending on the type of supervision, there are three main types of anomaly detection techniques: unsupervised, semi-supervised, and supervised. Given the wide variety of available anomaly detection algorithms, how can one choose which approach is most appropriate for a particular application? The purpose of this evaluation is to compare the performance of five unsupervised anomaly detection algorithms applied to a specific dataset from a small and medium-sized software enterprise, presented in this paper. To reduce the cost and complexity of a system developed to solve the problem of anomaly detection, a solution is to use machine learning (ML) algorithms that are available in one of the open-source libraries, such as the scikit-learn library or the PyOD library. These algorithms can be easily and quickly integrated into a low-cost software application developed to meet the needs of a small and medium-sized enterprise (SME). In our experiments, we considered some unsupervised algorithms available in PyOD library. The obtained results are presented, alongside with the limitations of the research.**

Keywords*—Unsupervised anomaly detection algorithms; small and medium-sized enterprise; traceability; open-source libraries

I. Introduction

The current societal landscape has seen an increase in the quantity and complexity of information processed daily. Such increasing use is required for effective management of current industrial processes and depends on the data acquired from the process itself, data that is cleaned and converted into information that can be used to create meaningful visualizations, be fed in complex control and prediction algorithms, or even stored for future reference and use. Moreover, data reliability is paramount. Correct information must be used to obtain correct responses from the managed processes and incorrect information can lead to inefficiency, loss of precision, data, or product that in turn can negatively impact the organization's reputation or the bottom line.

In general, the data is acquired from the process via sensors, manually or through automated systems. To eliminate acquisition errors, data is sanitized and, if possible, corrected. This prevents the propagation of errors further in the system. Data that does not meet the criteria for correction may appear anomalous compared with its dataset values or regarding the median of the dataset. In any scenario, this might indicate either erroneous data or valid data signaling a potential problem with data acquisition or in the process itself. Therefore, isolating anomalous data is an important indicator of data health and a promising path in data analysis.

Anomalies are unexpected instances of deviation from a large part of the dataset. Thus, solving them will allow for improving the efficiency of the underlying process [1]. In fact, according to various studies (e.g., [2]), applications based on anomaly detection could help an enterprise detect possible issues in time, before they emerge by identifying anomalous behavior, thus minimizing the risk of data loss and streamlining business processes. Anomaly detection finds application in multiple industries and domains, including healthcare, finance, manufacturing, construction, logistics, cyber security, and many others [3], [4]. There are various specific applications of anomalies detection, such as, system health monitoring, early detection of sepsis [5], event detection, product quality, intrusion detection, energy optimization, various real-time applications, to name only a few.

The anomaly detection market is witnessing growth, thus, according to [2], "the anomaly detection market size is expected to grow from USD 2.08 Billion in 2017 to USD 4.45 Billion by 2022, at a Compound Annual Growth Rate (CAGR) of 16.4%". This growth is being driven by the increasing development and dynamic adoption of emerging technologies such as big data analytics, data mining and business intelligence, machine learning and artificial intelligence.

According to scientific literature, there are three main types of anomaly detection techniques, depending on the type of supervision: unsupervised, semi-supervised, and supervised. Essentially, the choice of anomaly detection method can be made according to the labels available in the dataset [6].

Considering the extensive variety of available anomaly detection algorithms, how can one choose which approach is most suitable for a particular application? Clearly, performance in anomaly detection is a significant factor in algorithm selection. Unfortunately, there is no one approach that is best in every context and for all domains. Depending on the specifics, one algorithm may be superior to the others for a given user or dataset. Selecting an appropriate algorithm for a specific application is still a difficult design choice [7]. This is even more important in traceability systems that must ensure that the

product's lifecycle is correct, where the detection of an anomaly in the process can have a major impact on the production pipeline. However, these workflows can vary from company to company, which means that the use of supervised learning would imply higher costs to develop a custom model, whereas unsupervised learning and more precisely, anomaly detection would allow building a model that does not require human intervention, does not need prior knowledge about the process and, at the same time, can be very dynamic in terms of feature selection.

Reviewing the literature on machine learning-based anomaly detection algorithms reveals the diversity of algorithms evaluation and comparison approaches used by researchers. Consequently, the authors of [7] draw attention to the inconsistency in splitting between training and test datasets, in the selection of performance metrics and in the threshold used to indicate anomalies. Moreover, they point out the ambiguity in the definition of the positive class (i.e., the class of interest) utilized to evaluate the various models. Because of these inconsistencies, the authors find it difficult to understand the experimental evaluations presented in different papers [7].

To reduce the cost and complexity of a system developed to solve the problem of anomaly detection, a solution is to use machine learning (ML) algorithms that are available in one of the open-source libraries, such as the scikit-learn library [8], [9] or the PyOD library [10]. These algorithms can be easily and quickly integrated into a low-cost software application developed to meet the needs of a small and medium-sized enterprise (SME). Once the relevant features considered for anomaly detection are selected and pre-processed, the integrated algorithms can be applied within the specific software application.

In this paper, we will examine the viability of implementing anomaly detection in a traceability system by applying unsupervised anomaly detection algorithms. In this regard, we will study and compare the performance of some of the unsupervised anomaly detection algorithms applied on a dataset provided by the information technology (IT) department of a small and medium-sized software enterprise. We considered some un-supervised algorithms available in one of the popular open-source libraries, namely PyOD library. This library provides several benefits over comparable existing libraries. For instance, it contains more than 20 algorithms, it "implements combination methods for merging the results of multiple detectors and outlier ensembles which are an emerging set of models", and "all models are covered by unit testing with cross platform continuous integration, code coverage and code maintainability checks" [10]. These benefits have led to its widespread adoption in academic and commercial applications [10]. According to [11], [12], the GitHub repository has more than 10,000 monthly visitors, and more than 6,000 monthly downloads for PyPOD.

The remainder of this paper is structured as follows. Section II provides an overview of machine learning-based solutions for traceability domain, methods and algorithms in anomaly detection and the evaluated anomaly detection algorithms. The methodology we relied on to conduct the presented research is also discussed. Section III describes the experimental process and results obtained on a real dataset. Section IV is dedicated to presenting insights on the performance of the algorithms. The limitations and directions for future research are presented in Section V. The final section provides the concluding remarks of the paper.

II. Materials and Methods

In this section, we review prior work in terms of machine learning in traceability, and methods and algorithms in anomaly detection.

A. Machine Learning in Traceability

By employing traceability systems, products can have better quality, or the workflow can be improved. Thus, this concept has been applied in a variety of domains, ranging from managing the food supply chain [13], [14] to the automotive industry [15]. Moreover, given the high volume of data, ML algorithms could be used to analyze and provide relevant information that can be used in the decision-making process.

Given that food safety is a critical concern, traceability has been used in this industry in order to follow the process taken by perishables to ensure safety and quality. For example, De Nadai Fernandes et al. [16] employed three supervised ML algorithms to determine the source of bovine meat in Brazil, where there was a loss of information during the slaughtering or marketing processes. On the other hand, Alfian et al. [13] used Radio Frequency IDentification (RFID) tags and Internet of Things (IoT) sensors to collect information regarding the environment and track when produce would pass through a space, for example, a warehouse door. They also employed supervised learning to identify the direction of the product and, thus, determine whether the products were safely stored. To prevent food safety incidents, Wang et al. [14] developed a traceability system that ensured quality at each stage of the production pipeline by developing supervised ML algorithms to determine the quality at each stage and establish the final quality, whereas Shahbazi and Byun [17] utilized ML and blockchain to detect counterfeits and ensure the validity of the expiration dates.

Sharma et al. [18] reviewed the use of ML algorithms in the agricultural supply chain and observed that these algorithms were implemented at each step of the production process to improve efficiency. For instance, in the preproduction stage, ML algorithms were used to predict the harvest, soil properties and irrigation management. In the production step, they were used to predict the weather, protect the harvest, detect weeds, manage animals, and overview the harvest quality. The processing step consisted of algorithms used to predict the demand and plan the production, while in the distribution phase they were used to improve transportation, analyze the consumers, and manage the inventory.

Other domains have also included ML algorithms to analyze performance, for instance, in the automotive industry [15], or to determine validity in software maintenance for traceability link recovery [19]. One important observation is the focus on quality control and preventing issues that could occur. However, most of the discussed systems used supervised machine learning algorithms that need labeled data as input, which means that the process must be firmly

established, and any change means having to reformat and revalidate the training dataset. In this context, we looked at unsupervised anomaly detection algorithms that could be applied to a more dynamic process to alert the user of any abnormalities in the system by using an unlabeled dataset.

The next sections provide brief descriptions of the algorithms used for anomaly detection.

B. Methods and Algorithms in Anomaly Detection

Detection of anomalous instances in various datasets is of great importance in many processes. Outlier observations, records, recorded values, states, and devices can either affect the workflow of processes or can induce bias in computed values or scores. Removing the outliers is a valid and known technique, but the detection of the aforementioned anomalous values is not an easy process – especially when the data is not identified and tagged [6].

Anomaly detection techniques are employed in various working domains, but especially in supply chain management, where the capabilities of various techniques allow to manage complicated processes, using predictive algorithms and other use cases. For example, using blockchain technologies in supply chain systems allows for novel methods to manage all aspects of the process. However, to have a robust data model and verification mechanism to ensure the integrity of the processes it is required to implement mechanisms to correct anomaly signals in the acquired data required in verification processes for the business logic involving transactions (both resource intensive and critical for the wellbeing of the platform) [20], [21] between entities in the platform, for a better quality in supply chain management (not only from a performance point of view, but also from a security perspective).

In many applications that require anomaly data detection, ML algorithms are used to highlight the relevant data for future correction, removal, or analysis [22]. Many times, the detection algorithms are paired with traditional detection systems, usually rule-based, for better performance. In this regard, there are various application domains where these types of algorithms are used [6]: network security for intrusion detection (used for behavior analysis in enterprise settings for known and novel threats), surveillance for suspicious moves and actions (via visual and audio capture systems) [23], [24], detection of fraudulent transactions in banking industry (including transactions involving digital goods) [25], [26], energy optimization in smart buildings [27], medical smart equipment (capable of identification and analysis of anomalies to assist in medical diagnosis) [28], [29], and, generally, in use-cases where anomalous states can appear infrequently enough in the operating processes to be properly treated, but pose enough dangers to warrant such a system.

Even in processes that are tied to physical mediums, like, for example, nuclear radiation detection [30], telemetry data for spacecraft operations [31], traffic patterns analysis [32], sensor arrays and IoT systems [33], unmanned ground and aerial vehicles detection [34], edge computing systems and novel large-scale IT systems [35] or network quality of service assurance by using a Greedy algorithm [36] or even genetic algorithms [37], such ML algorithms can be used for identification of anomalies. Moreover, similar algorithms are used in domains like supply chain management, where genetic rule-based and graph-based detection methods are employed to verify business transactions regarding their validity [20].

Anomaly detection domain can be classified in three types [6], based on the approach regarding labelling of the dataset content: supervised anomaly detection where the training data and the test data are fully labelled (in practice this is less used given that labelling the data is not always feasible or even possible), semi-supervised anomaly detection where the training is done on non-anomalous datasets (the anomalies being detected when they deviate from the "correct" model) and unsupervised anomaly detection that does not require labels to classify data (most flexible approach), the distinction being made on the internal properties of the dataset.

As stated in the introduction section, this paper has considered the evaluation of several unsupervised algorithms for anomaly detection in the context of traceability. According to [6], unsupervised anomaly detection algorithms can be roughly classified into the following main categories as illustrated in Fig.1 (1) Nearest-neighbor based techniques, (2) Clustering-based methods, (3) Statistical algorithms, (4) Subspace techniques, (5) Classifier-based algorithms.

In this paper, the evaluated algorithms are part of the first three groups.

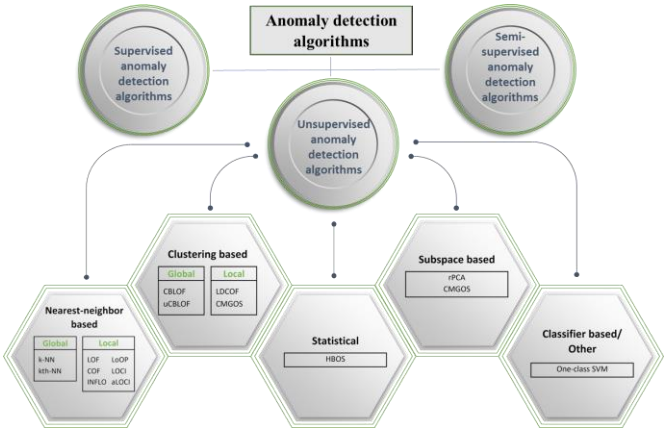

Fig. 1. Taxonomy of Unsupervised Anomaly Detection Algorithms (Adapted after [6]).

C. The Evaluated Anomaly Detection Algorithms

In the following, it briefly presents the five anomaly detection algorithms that were evaluated. The k-nearest neighbor (k-NN or KNN) algorithm consists of finding the nearest neighbors and calculating the anomaly score based either on the distance to the nearest neighbor [38] or the mean distance to the k nearest neighbors [39]. One of the drawbacks of this algorithm is that it detects only global anomalies, an issue that was tackled in the Local Outlier Factor (LOF) algorithm [40], which was the first algorithm to detect local anomalies. The Cluster-Based Local Outlier Factors (CBLOF) algorithm [41] uses the grouping to determine the dense areas from the data and uses a heuristic to classify the groups,

whereas Histogram-based Outlier Score (HBOS) [42] is based on statistics and assumes that there are no dependencies between the features of the model. One challenge faced by clustering algorithms is choosing the number of groups, which was addressed by Local Correlation Integral (LOCI) [43] that uses a maximization approach.

The implementation of these algorithms in various software libraries, such as the PyOD library [10], facilitates their use in the development of software applications for anomaly detection.

D. Performance Criteria

In this paper, we have examined the possibility of employing anomaly detection in a traceability system by applying five unsupervised anomaly detection algorithms and compared their performance.

The algorithms evaluated in the considered scenario must be analyzed from a performance perspective. Given that not necessarily all algorithms can be implemented in the traceability platform (for performance reasons), usually the best algorithm will be employed in the final product, for best performance/accuracy ratio. Alternatively, two or three algorithms can be employed in certain circumstances, where their performance can compensate for their weaknesses in certain datasets configurations or certain limited cases. Thus, it is important to establish the performance of the potential solution in the given circumstances, including when adjusting the settings in the classifier model.

In this regard, a criterion that is often used to test the performance of algorithms in machine learning is the ROC curve (Receiver Operating Characteristics) [44]. ROC metric will help establish the performance of the model (higher the value, better the outcome) by plotting the rate of true positives compared (higher is better) with the rate of false positives (lower is better) and thus establishing a threshold for the performance of the model in classifying the input data [44]; this approach is important when deciding between various algorithms or when adjusting operating parameters of a given algorithm.

Another important criterion is the accuracy score. In performance metrics [45], the accuracy score is the measure by which the classifier will offer correct predictions compared with the total number of predictions made. Obviously, a greater accuracy is a highly desirable behavior of the model.

Finally, precision @ rank n represents the precision of the model up to n^{th} prediction from the total number of predictions [46]. This metric helps usually in choosing a better model for a given type of problem, given that the goal is to obtain a good fit for our algorithm.

$$\text{Precision@k} = \frac{\text{true}_{\text{positives}}@k}{\text{true}_{\text{positives}}@k + \text{false}_{\text{positives}}@k} \quad (1)$$

Training time is also an important factor in choosing a certain algorithm. Given that the time spent training the models can be an expensive proposition (for example in big datasets or when the datasets change in time, requiring re-training of the model), choosing an algorithm that is faster on training is better, as long as the performance metrics are not suffering.

As highlighted in [7], different splits of the training and test data can be used while comparing the performances of various algorithms. Thus, decisions have to be made regarding the splitting of training and test sets. For anomaly detection, it must additionally be decided whether one of these two sets will contain normal data, abnormal data, or both. Regardless of the decision taken, it must be used in a consistent manner when evaluating different algorithms [7].

In the next section, we present the considered use case.

E. Use Case

In order to perform the proposed comparison of some anomaly detection algorithms, we have considered the use case of the traceability of different equipment types in an IT department Fig. 2 presents the main steps of our process of evaluation.

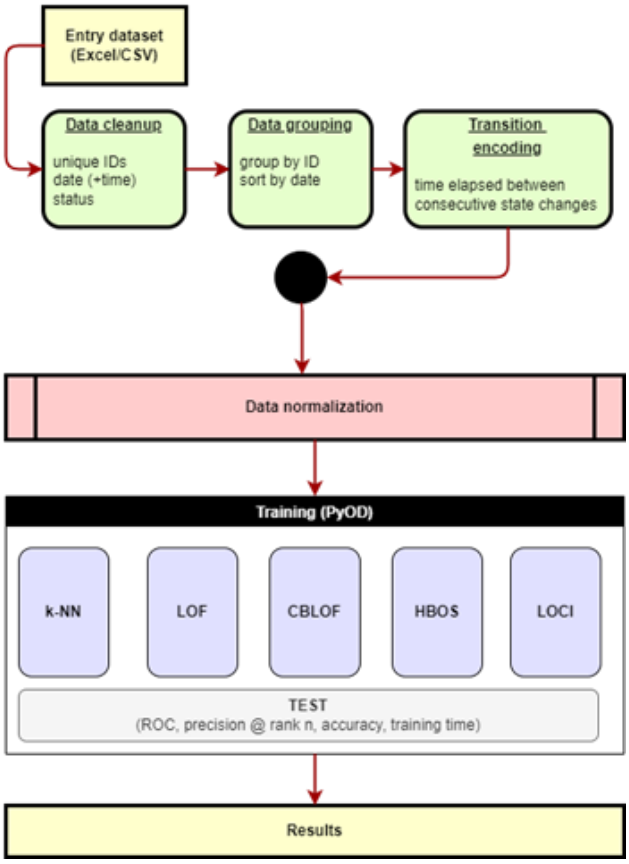

Fig. 2. The Workflow of the Experiment.

Next, we describe the data pre-processing and applied methods.

To perform the proposed anomaly detection comparison, we were provided with a set of data in the form of an Excel spreadsheet or a CSV (comma separated values) file that contained information regarding the management of equipment in an IT department. The spreadsheet was composed of records of items that were registered and then assigned to employees. There were cases when items malfunctioned, so they were sent back to the IT department for repairs and assigned back to an employee or had to be scrapped. The file consists of multiple

data columns, such as unique ID, date, equipment name, equipment type, employee name, status, etc. The dataset contained a number of 347 transitions for 130 items with 130 records for the acquired status, 177 instances of assigning an item to an employee for use, 22 cases of sending the item to the IT department for repairs, and 18 recorded for items being scrapped. It was considered that the data provided was valid and could be used for training a machine learning model, and that artificial anomalies could be added to the set of data for validation. It should be noted that this process can be applied to other types of items, regardless of their statuses as this process is generic and does not require any previous setup.

The first step of the pre-processing phase is extracting the relevant information, in this case from the Excel or CSV file, but other sources, such as databases, can be used. The significant attributes are the item unique identifier, and the date and status of the item when that record was made. Depending on how the data is stored, the status of the item may have a variety of formats, ranging from integers to strings. For example, we considered the coding of statuses presented in Table I.

Based on the previous coding of the statuses, Table II, illustrates the case of a monitor with the ID 132606, which was purchased (status with unique ID 1) on January 27th and given in use (status with unique ID 3) to an employee the next day, on January 28th. On June 25 of the same year, the item identified with ID 132606 is sent for repairs.

The date, and time if available, must have the same format and should be converted to a format that would allow for simple time difference calculations. Since the provided spreadsheet contained only the registration date, it was converted to the number of days since January 1st, 1900, in accordance with the ISO 8601:2000 YYMMDD format.

Secondly, to define the item transitions, all data must be grouped according to the unique identifier and sorted by date inside that group. If time is known, it should also be considered in the ordering, and this would be of greater significance in situations in which items transition multiple statuses on the same day. In our case, after going through these steps, the data for an item should reveal how an item was purchased and then assigned to employees with some cases where it was sent for repairs or was scrapped. A transition is represented based on the time elapsed between the current state and the previous state in the form of a number of days, or a number of minutes or milliseconds (depending on the granularity of the data), if the data contained time. The transition token is composed of the concatenation of the current and previous status identifier, which was defined in the previous step. A sample of the data is shown in Table III.

In this example, if item 132606 was purchased on the 27th of January and assigned (status with the three unique ID) to an employee on the 28th of January that same year, then this transition will be characterized by a one-day time interval, resulting from the difference between the two dates, and the "13" token, which represents the concatenation of the unique identifiers assigned to the status attributes. These will be the two features that will be used to train the ML model, which were selected as relevant as a result of an analysis of what information is critical in a traceability system with the purpose of creating a trained model that could point out the incorrect transitions. Therefore, mistakes could be made while changing the status of an item, for instance, moving a monitor from acquired to scrapped would be invalid, whereas if, for example, the monitor is left in the in-repair state for a prolonged period reveals a different type of issue such as the lack of available employees to check the monitor. When an item is added to the system and is assigned its first status, the elapsed time should be set to zero and the token should consist of the doubling of the unique identifier status, in our case, the token for the first transition of item 132606 is "11". If an item starts with a different status, then the ML algorithm should be able to signal it as an anomaly.

TABLE I. THE CODING OF THE CONSIDERED STATUSES

Status	Code
Purchased	1
Scrapped	2
In use	3
Maintenance	4

TABLE II. SAMPLE OF THE ENCODED DATA RECEIVED FROM IT DEPT.

Unique item ID	Status	Date
132606	1	January 26, 2021
132606	3	January 28, 2021
132606	4	June 25, 2021
132606	3	July 02, 2021
134338	1	April 29, 2021
134338	3	May 14, 2021
134338	2	July 30, 2021

TABLE III. EXAMPLE OF ITEM TRANSITIONS

Unique item ID	Transition token	Timelapse (days)
132606	11	0
132606	13	1
132606	34	148
132606	43	7
134338	11	0
134338	13	15
134338	32	77

The third step before training the models is normalizing the data, which in this case was performed using the preprocessing scale function from the scikit-learn [9] Python package. Normalization needs to be applied to each feature, because having overly broad scales could cause issues when training the ML model. This is avoided through the fact that the new values are smaller and at the same time they maintain the general distribution and data ratios, which means that the relevant information is preserved. Before applying this step, we also added a number of four artificial anomalies consisting of invalid transitions, new statuses, and prolonged periods of time, which should be signaled by the ML model. The added artificial anomalies had to be run through the same scaling process as the valid set of data to avoid inaccuracies. We added such a small number of anomalies, because, as per their definition, anomalies are events that occur rarely.

To analyze the performance of the applied algorithms the data has been formatted and split into a file that contains all the valid transitions, a file that contains 80% of the data and four artificial anomalies that can be used for training the model, and a file that contains the rest of the 20% of the data and other four artificial anomalies that can be used as the test data, which will be utilized to validate whether the model is overfitted. The split between the test and training data is done randomly by shuffling the item IDs and then selecting 20% of the IDs and their corresponding transitions for the test data. The rest of the items are assigned to the training dataset. In addition, when the data is split, it should be normalized separately to avoid any data leakage. Given that the IDs are separated randomly, this means that the data will be split differently when the process is run again on the same set of data.

III. Results

Next, the results of the training and tests processes for the ML algorithms having as input the datasets described in the previous section of this paper are presented. To evaluate and compare the anomaly detection methods we use the standard metrics of precision, ROC and accuracy. Also, it was considered the training time for each algorithm. The experimental evaluation is conducted on a laptop with i7-8550U CPU @1.80 GHz x 8, 16 GB of RAM, running Ubuntu 16.04 LTS. The code is written in Python 3.

In the experiments conducted, the split of data was performed considering normal and anomalous data, both in the training and in the testing phase. For the first analysis, we used the training dataset that had 80% of the whole data, and the test train dataset composed of 20% of the provided information. Each set contained the same four artificial anomalies that were used for validation. The first set of data was used to train five separate anomaly detection algorithms available in the PyOD Python package [10] by first initializing them and then applying the fit function that received the training set of data. This resulted in a list of prediction labels and outlier scores of the training data, which we used to compute the accuracy of the model based on the training dataset. To evaluate the model, the predict and *decision_function* functions were used, which received the test dataset as input, whose results were used to calculate the accuracy of the test predictions. Accuracy was calculated based on the assumption that the received dataset contained only valid records and that there were four artificial anomalies. To calculate the ROC and the Precision @ rank n, we used the *evaluate_print* function provided by the PyOD library. It was also recorded the time needed to train each model. Table IV contains the results of these metrics for the analyzed five algorithms. The evaluation results are shown in Fig. 3.

Given that the provided dataset was small, during the second analysis the complete dataset was used as the training data and some predictions were made based on four cases: normal record for item just being added to the system (N1), normal record for item being assigned to employee (N2) that was taken from the initial dataset, anomaly with non-existent states (A1), and anomaly with item assigned to employee but with an anomalous time (A2). Before sending these datasets to the algorithms for predictions they were processed by using the same normalization parameters used for the training set. Table V shows the results of training the five algorithms with the full set of data and using the previously described cases to validate the predictions. The evaluation results are shown in Fig. 4.

Nevertheless, in a normal setting, the prediction would be made based on a model that was trained using the full set of data, which could occur right when there is a request to predict whether a transition is anomalous or not, if the training is fast enough, or periodically, in which case it would contain only the records that were registered until the time of training.

TABLE IV. Results of Algorithm based on Training and Test Dataset

Metric/ algorithm	**KNN**		**LOF**		**CBLOF**		**HBOS**		**LOCI**	
	Train	***Test***	***Train***	***Test***	***Train***	***Test***	***Train***	***Test***	***Train***	***Test***
ROC	0.9982	0.9786	0.9188	0.5000	0.9657	0.9893	0.9251	0.8429	0.6922	0.9643
Precision @ rank n	0.7500	0.7500	0.5000	0.0000	0.7500	1.0000	0.2500	1.0000	0.0000	0.5000
Accuracy (%)	91.45	72.97	90.74	52.70	91.10	82.43	96.44	98.64	88.61	87.83
Time (seconds)	0.0023		0.0037		0.0699		0.0019		44.8992	

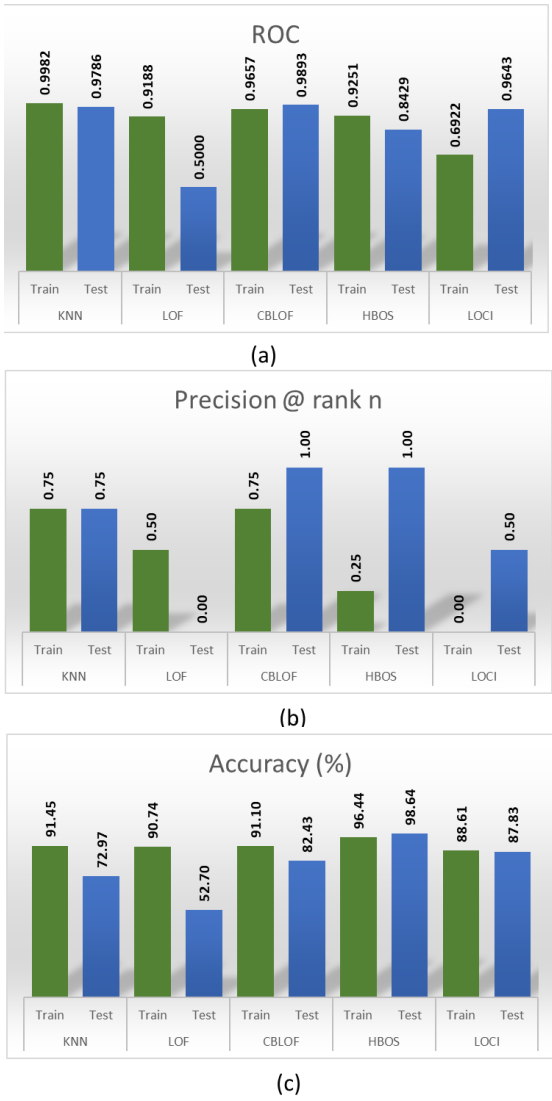

Fig. 3. Experiment Results for the First Analysis: (a) ROC Curve; (b) Precision @ Rank n; (c) Accuracy (Expressed as Percentages).

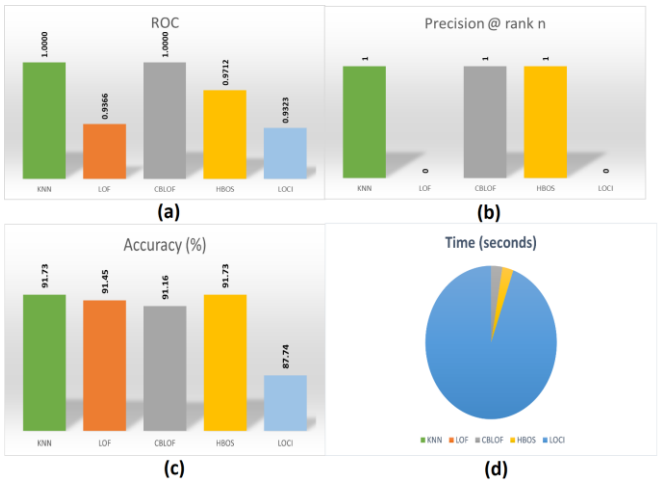

Fig. 4. Experiment Results for the Second Analysis: (a) ROC Curve; (b) Precision @ Rank n; (c) Accuracy (Expressed as Percentages); (d) Execution Time (in Seconds).

TABLE V. RESULTS OF THE ALGORITHMS TRAINED ON THE FULL DATASET AND VALIDATED BASED ON FOUR PREDICTIONS

Metric/ algorithm	**KNN**	**LOF**	**CBLOF**	**HBOS**	**LOCI**
ROC	1.0000	0.9366	1.0000	0.9712	0.9323
Precision @ rank n	1.0000	0.0000	1.0000	1.0000	0.0000
Accuracy (%)	91.73	91.45	91.16	91.73	87.74
Time (seconds)	0.0042	0.0044	2.5900	2.5399	88.51
N1 prediction	Correct	Correct	Correct	Correct	Correct
N2 prediction	Correct	Correct	Correct	Wrong	Correct
A1 prediction	Correct	Correct	Correct	Correct	Wrong
A2 prediction	Correct	Correct	Correct	Correct	Wrong

IV. DISCUSSION

In this section we discuss and analyze the results of running the five machine learning algorithms, mainly k-NN (k-nearest-neighbor) - a nearest-neighbor-based unsupervised algorithm focused on detection of global anomalies (global relative to the dataset) with low computational impact [47], LOF (Local Outlier Factor) - a nearest-neighbor-based algorithm able to detect local anomalies alongside global ones [40], [48], LOCI (Local Correlation Integral) - a nearest-neighbor-based local algorithm with increased precision over k-NN but also with increased computational complexity [43], [45], CBLOF (cluster-based local outlier factor) - a clustering-based global algorithm [49] and HBOS (histogram-based outlier score), a very fast statistical algorithm almost an order of magnitude faster than k-NN [50].

By examining the results for the ROC scores from Table IV, it can be observed that most of the algorithms had good outcomes for the training values apart from LOCI with 0.6922, with the best being KNN (0.9982) followed by CBLOF (0.9657). On the other hand, the ROC values for the test sets show us that not all the models generalized well, for instance, the test ROC for LOF was 0.5, whereas HBOS had a lower score than the other algorithms with 0.8429. The best results were achieved again by KNN (0.9786) and CBLOF (0.9893), followed by LOCI (0.9643).

In terms of the precision @ rank n score, overall, CBLOF had the best results (0.75 for training and 1.0 for test) followed by KNN (0.75 for both the training and test datasets). Although HBOS had a low value of 0.25 for the training set, it had an exceptionally good score of 1.0 for the test set, whereas the other algorithms had low scores in general with 0.5 and 0.0 for train and test, respectively, for LOF, and 0.0 and 0.5 for train and test, respectively, for LOCI.

The accuracy score results revealed that HBOS performed the best with 96.44% for training and 98.64% for test with CBLOF being second with 91.10% for training and 82.43% for test. Even though KNN had a good result for the training dataset, 91.45%, it scored lower for the test dataset, 72.97%. Albeit having lower accuracy ratings, LOCI had similar values for both the train and test sets with 88.61% and 87.83%, respectively, whereas LOF had a considerable difference

between the test and train scores (90.74% and 52.70% respectively), which denotes that the model did not generalize.

Although the training time would not be a major factor to consider if the training is performed daily when there is no heavy traffic in the system, it can be still noted that LOCI had a significantly higher training time compared to the other algorithms. Its training lasted almost 45 seconds even if the dataset was relatively small while all the other models had times lower than a second, making LOCI definitively not a suitable candidate for such a system.

The second round of experiments displayed in Table V, which are closer to a real scenario, revealed some interesting results. Firstly, when using the full dataset for training, all algorithms had high scores for the ROC values with KNN and CBLOF having 1.0 followed by LOF and LOCI that had remarkably similar results, 0.9366 and 0.9323, respectively. On the other hand, the precision @ rank n was either exceptionally good with 1.0 for KNN, HBOS and CBLOF or bad with 0.0 for LOF and LOCI.

The accuracy scores were also higher in general, with KNN and HBOS having the same outcome of 91.73% followed by LOF at 91.45%, CBLOF at 91.16% and LOCI at 87.74%, which had a very similar result to the first experiment denoting a consistent pattern in the ability of this model to predict the anomalies for this dataset. In terms of the training time, KNN and LOF had the lowest results with times under a second. However, CBLOF and HBOS had significantly higher times (around 2.5 seconds), which can indicate a more rapid increase in time given that the training from the first experiment was performed on 80% of the data. LOCI still had the highest time with 88.51 seconds.

Regarding the four predictions, two normal cases and two anomalies, KNN, LOF and CBLOF correctly predicted all four cases, HBOS wrongly detected N2 as an anomaly, whereas LOCI was not able to detect the two anomalies. Although HBOS had similar results to KNN and CBLOF for the other evaluation conditions, it did not perform as well in terms of the test prediction. Thus, overall, the best results were achieved by KNN followed closely by CBLOF apart for the training time.

V. Limitations

The size of the used dataset is a limiting factor in our work, however, even with such a small size we were able to demonstrate that anomaly detection can be applied to traceability with good results. Nonetheless, having a larger dataset could offer more insight, an analysis that could be undertaken in future work, where we could also include more algorithms in the comparison. The difficulty in creating a database for anomaly detection lies in the fact that the results will emulate the logic that was used in generating the data, thus, it is important to have access to a real dataset.

In future research, data splitting will be performed considering the normal data for training and all anomalous samples in the test set.

In terms of the discussed logic, adding a new process or status will automatically result in an anomaly detection. To solve this problem, the described logic could be adapted to add a threshold, for example, there must be a minimum number of products that went through a status in order to send that transition to the anomaly detection algorithm. Another option would be to check if a generated group has lower elements than a set threshold. However, this could also mean that transitions that could be anomalous are not sent or are not taken into account by model. Handling this issue could be further investigated in future work.

In this analysis, only one process was considered. In order to improve the performance of the model, anomaly detection should be conducted for each type of process if the company workflow has various processes with different statuses. Obviously, for training a model there must be a minimum number of transitions for each category, which might be determined by user input or by involving a user in model validation. Once the model was able to properly detect anomalies, the threshold could be automatically set for future categories. Nevertheless, this could open the door to semi-supervised learning, which will be investigated in future work.

VI. Conclusions

In recent years, there has been an increase in the number of artificial intelligence-based solutions for problems in various fields. Anomaly detection is an issue for which there are currently several approaches. One of the most widespread methods involves the use of ML techniques.

In this paper, the performance of five unsupervised anomaly detection algorithms regarding a traceability dataset that contains information on the management of devices from an IT department was analyzed and compared. The models used with anomaly detection algorithms based on machine learning do not require labeled data. Given the importance of reproducibility in research, we presented all the information regarding the implementation that allow double-checking the results and verifying whether they are reliable. By analyzing the precision, accuracy, ROC, and time we determine which algorithms tend to perform better or worse on the presented use case. We have demonstrated experimentally that these algorithms can be successfully applied to determine whether a new transition is an anomaly with an accuracy of up to 91.73%.

Acknowledgment

This research was funded by the project “119722/Centru pentru transferul de cunoștințe către întreprinderi din domeniul ICT—CENTRIC, Contract subsidiar 15568/01.09.2020, Smart Tracking Platform (STP)”, contract no. 5/AXA 1/1.2.3/G/13.06.2018, cod SMIS 2014+ 119722 (ID P_40_305).

References

[1] Y. Wang et al., “Iterative anomaly detection,” in 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Jul. 2017, pp. 586–589. doi: 10.1109/IGARSS.2017.8127021.

[2] “Anomaly detection market by Solution (Network and user behavior anomaly detection), technology (Big data analytics, data mining and business intelligence, machine learning and artificial intelligence), deployment, service, vertical - Global forecast to 2022,” MarketsandMarkets, Market report. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/anomaly-detection-market-138133262.html.

[3] M. Bahri, F. Salutari, A. Putina, and M. Sozio, "AutoML: State of the art with a focus on anomaly detection, challenges, and research directions," Int. J. Data Sci. Anal., vol. 14, no. 2, pp. 113–126, Aug. 2022, doi: 10.1007/s41060-022-00309-0.

[4] I. K. Nti, A. F. Adekoya, B. A. Weyori, and O. Nyarko-Boateng, "Applications of artificial intelligence in engineering and manufacturing: a systematic review," J. Intell. Manuf., vol. 33, no. 6, pp. 1581–1601, Aug. 2022, doi: 10.1007/s10845-021-01771-6.

[5] L. Begic Fazlic et al., "A novel hybrid methodology for anomaly detection in time series," Int. J. Comput. Intell. Syst., vol. 15, no. 1, p. 50, Jul. 2022, doi: 10.1007/s44196-022-00100-w.

[6] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," PloS One, vol. 11, no. 4, 2016, doi: https://doi.org/10.1371/journal.pone.0152173.

[7] M. Alvarez, J.-C. Verdier, D. K. Nkashama, M. Frappier, P.-M. Tardif, and F. Kabanza, "A revealing large-scale evaluation of unsupervised anomaly detection algorithms," ArXiv Prepr. ArXiv220409825, 2022.

[8] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, Jan. 2012.

[9] D. Cournapeau, "scikit-learn," scikit-learn, 2022. https://scikit-learn.org/stable/about.html (accessed Jul. 24, 2022).

[10] Y. Zhao, Z. Nasrullah, and Z. Li, "PyOD: A Python toolbox for scalable outlier detection," J. Mach. Learn. Res., vol. 20, no. 96, pp. 1–7, 2019, [Online]. Available: http://jmlr.org/papers/v20/19-011.html.

[11] Y. Zhao and M. K. Hryniewicki, "DCSO: Dynamic combination of detector scores for outlier ensembles," ArXiv Prepr. ArXiv191110418, 2019, doi: https://doi.org/10.48550/arXiv.1911.10418.

[12] Y. Zhao, Z. Nasrullah, M. K. Hryniewicki, and Z. Li, "LSCP: Locally selective combination in parallel outlier ensembles," in Proceedings of the 2019 SIAM International Conference on Data Mining, 2019, pp. 585–593.

[13] G. Alfian et al., "Improving efficiency of RFID-based traceability system for perishable food by utilizing IoT sensors and machine learning model," Food Control, vol. 110, p. 107016, Apr. 2020, doi: 10.1016/j.foodcont.2019.107016.

[14] J. Wang, H. Yue, and Z. Zhou, "An improved traceability system for food quality assurance and evaluation based on fuzzy classification and neural network," Food Control, vol. 79, pp. 363–370, Sep. 2017, doi: 10.1016/j.foodcont.2017.04.013.

[15] M. Syafrudin, G. Alfian, N. L. Fitriyani, and J. Rhee, "Performance analysis of IoT-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing," Sensors, vol. 18, no. 9, Art. no. 9, Sep. 2018, doi: 10.3390/s18092946.

[16] E. A. De Nadai Fernandes, G. A. Sarriés, M. A. Bacchi, Y. T. Mazola, C. L. Gonzaga, and S. R. V. Sarriés, "Trace elements and machine learning for Brazilian beef traceability," Food Chem., vol. 333, p. 127462, Dec. 2020, doi: 10.1016/j.foodchem.2020.127462.

[17] Z. Shahbazi and Y.-C. Byun, "A procedure for tracing supply chains for perishable food based on blockchain, machine learning and fuzzy logic," Electronics, vol. 10, no. 1, Art. no. 1, Jan. 2021, doi: 10.3390/electronics10010041.

[18] R. Sharma, S. S. Kamble, A. Gunasekaran, V. Kumar, and A. Kumar, "A systematic literature review on machine learning applications for sustainable agriculture supply chain performance," Comput. Oper. Res., vol. 119, p. 104926, Jul. 2020, doi: 10.1016/j.cor.2020.104926.

[19] C. Mills and S. Haiduc, "A machine learning approach for determining the validity of traceability links," in 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), May 2017, pp. 121–123. doi: 10.1109/ICSE-C.2017.86.

[20] B. Oh, T. J. Jun, W. Yoon, Y. Lee, S. Kim, and D. Kim, "Enhancing trust of supply chain using blockchain platform with robust data model and verification mechanisms," in 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Oct. 2019, pp. 3504–3511. doi: 10.1109/SMC.2019.8913871.

[21] M. Khalfaoui, R. Molva, and L. Gomez, "Secure alert tracking in supply chain," in 2013 International Conference on Security and Cryptography (SECRYPT), Jul. 2013, pp. 1–11.

[22] S. B. Wankhede, "Anomaly detection using machine learning techniques," in 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), Mar. 2019, pp. 1–3. doi: 10.1109/I2CT45611.2019.9033532.

[23] L. Kratz and K. Nishino, "Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models," in 2009 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2009, pp. 1446–1453. doi: 10.1109/CVPR.2009.5206771.

[24] P. Khaire and P. Kumar, "A semi-supervised deep learning based video anomaly detection framework using RGB-D for surveillance of real-world critical environments," Forensic Sci. Int. Digit. Investig., vol. 40, p. 301346, Mar. 2022, doi: 10.1016/j.fsidi.2022.301346.

[25] V. Chang, L. M. T. Doan, A. Di Stefano, Z. Sun, and G. Fortino, "Digital payment fraud detection methods in digital ages and Industry 4.0," Comput. Electr. Eng., vol. 100, p. 107734, May 2022, doi: 10.1016/j.compeleceng.2022.107734.

[26] J. Vanhoeyveld, D. Martens, and B. Peeters, "Value-added tax fraud detection with scalable anomaly detection techniques," Appl. Soft Comput., vol. 86, p. 105895, Jan. 2020, doi: 10.1016/j.asoc.2019.105895.

[27] Y. Himeur, K. Ghanem, A. Alsalemi, F. Bensaali, and A. Amira, "Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives," Appl. Energy, vol. 287, p. 116601, Apr. 2021, doi: 10.1016/j.apenergy.2021.116601.

[28] N. Melnykova, R. Kulievych, Y. Vycluk, K. Melnykova, and V. Melnykov, "Anomalies detecting in medical metrics using machine learning tools," Procedia Comput. Sci., vol. 198, pp. 718–723, Jan. 2022, doi: 10.1016/j.procs.2021.12.312.

[29] J. Lin, E. Keogh, A. Fu, and H. Van Herle, "Approximations to magic: finding unusual medical time series," in 18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05), Jun. 2005, pp. 329–334. doi: 10.1109/CBMS.2005.34.

[30] P. Zhou and S. Abbaszadeh, "Towards real-time machine learning for anomaly detection," in 2020 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), Oct. 2020, pp. 1–3. doi: 10.1109/NSS/MIC42677.2020.9507937.

[31] M. M. Fernández, Y. Yue, and R. Weber, "Telemetry anomaly detection system using machine learning to streamline mission operations," in 2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT), Sep. 2017, pp. 70–75. doi: 10.1109/SMC-IT.2017.19.

[32] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, p. 15:1-15:58, Jul. 2009, doi: 10.1145/1541880.1541882.

[33] L. Erhan et al., "Smart anomaly detection in sensor systems: A multi-perspective review," Inf. Fusion, vol. 67, pp. 64–79, Mar. 2021, doi: 10.1016/j.inffus.2020.10.001.

[34] S. Khan, C. F. Liew, T. Yairi, and R. McWilliam, "Unsupervised anomaly detection in unmanned aerial vehicles," Appl. Soft Comput., vol. 83, p. 105650, Oct. 2019, doi: 10.1016/j.asoc.2019.105650.

[35] O. M. Ezeme, Q. H. Mahmoud, and A. Azim, "A deep learning approach to distributed anomaly detection for edge computing," in 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Dec. 2019, pp. 992–999. doi: 10.1109/ICMLA.2019.00169.

[36] Ç. Ateş, S. Özdel, M. Yıldırım, and E. Anarım, "Network anomaly detection using header information with greedy algorithm," in 2019 27th Signal Processing and Communications Applications Conference (SIU), Apr. 2019, pp. 1–4. doi: 10.1109/SIU.2019.8806451.

[37] Q. Su and J. Liu, "A network anomaly detection method based on genetic algorithm," in 2017 4th International Conference on Systems and Informatics (ICSAI), Nov. 2017, pp. 1029–1034. doi: 10.1109/ICSAI.2017.8248437.

[38] M. Hassan, H. Maher and K. Gouda, "A Fast and Efficient Algorithm for Outlier Detection Over Data Streams," International Journal of Advanced Computer Science and Applications, vol. 12, no. 11, 2021.

[39] F. Angiulli and C. Pizzuti, “Fast outlier detection in high dimensional spaces,” in Principles of Data Mining and Knowledge Discovery, Berlin, Heidelberg, 2002, pp. 15–27. doi: 10.1007/3-540-45681-3_2.

[40] M. U. Rehman and D. M. Khan, "Local Neighborhood-based Outlier Detection of High Dimensional Data using different Proximity Functions," International Journal of Advanced Computer Science and Applications, vol. 11, no. 4, 2020.

[41] Z. He, X. Xu, and S. Deng, “Discovering cluster-based local outliers,” Pattern Recognit. Lett., vol. 24, no. 9, pp. 1641–1650, Jun. 2003, doi: 10.1016/S0167-8655(03)00003-5.

[42] M. Goldstein and A. Dengel, “Histogram-based Outlier Score (HBOS): A fast unsupervised anomaly detection algorithm.”, KI-2012: Poster and Demo Track, 2012.

[43] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos, “LOCI: Fast outlier detection using the local correlation integral,” in Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405), Mar. 2003, pp. 315–326. doi: 10.1109/ICDE.2003.1260802.

[44] T. Fawcett, “An introduction to ROC analysis,” Pattern Recognit. Lett., vol. 27, no. 8, pp. 861–874, Jun. 2006, doi: 10.1016/j.patrec.2005.10.010.

[45] M. E. Villa-Pérez, M. Á. Álvarez-Carmona, O. Loyola-González, M. A. Medina-Pérez, J. C. Velazco-Rossell, and K.-K. R. Choo, “Semi-supervised anomaly detection algorithms: A comparative summary and future research directions,” Knowl.-Based Syst., vol. 218, p. 106878, Apr. 2021, doi: 10.1016/j.knosys.2021.106878.

[46] N. Craswell, “Precision at n,” in Encyclopedia of Database Systems, L. Liu and M. T. Özsu, Eds. New York, NY: Springer, 2016, pp. 1–1. doi: 10.1007/978-1-4899-7993-3_484-2.

[47] A. E. Ezugwu et al., “A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects,” Eng. Appl. Artif. Intell., vol. 110, p. 104743, Apr. 2022, doi: 10.1016/j.engappai.2022.104743.

[48] E. H. Budiarto, A. Erna Permanasari, and S. Fauziati, “Unsupervised anomaly detection using K-Means, local outlier factor and one class SVM,” in 2019 5th International Conference on Science and Technology (ICST), Jul. 2019, vol. 1, pp. 1–5. doi: 10.1109/ICST47872.2019.9166366.

[49] H. Alimohammadi and S. Nancy Chen, “Performance evaluation of outlier detection techniques in production timeseries: A systematic review and meta-analysis,” Expert Syst. Appl., vol. 191, p. 116371, Apr. 2022, doi: 10.1016/j.eswa.2021.116371.

[50] M. Goldstein, “Anomaly detection in large datasets,” Phd thesis (published), Technische Universität Kaiserslautern, Germany, 2014. Accessed: Jul. 24, 2022. [Online]. Available: https://www.goldiges.de/phd/.