




UNSUPERVISED MACHINE LEARNING METHODS FOR REAL-TIME ANOMALY DETECTION IN ENDPOINTS

 K.N. Asgarov^{1*},  Y.N. Imamverdiyev¹,  M.M. Abutalibov²

¹Azerbaijan Technical University, Baku, Azerbaijan

²Qarabug MMC, Baku, Azerbaijan

Abstract. Among all the contemporary digital ecosystems, endpoint devices, including computers, servers, and IoT gadgets, represent at once the most important and the most vulnerable part of the infrastructure of any organization. Traditional security solutions very often cannot identify subtle and sophisticated threats that reveal their presence only through deviations from normal behavior. This paper presents the application of unsupervised machine learning methods-namely, Isolation Forest, One-Class Support Vector Machines, Gaussian Mixture Models and Local Outlier Factor-for real-time anomaly detection in endpoint devices. The efficacy, for this purpose, is shown in this paper using comprehensive telemetry data such as CPU usage, memory consumption, and network traffic, whether anomalies due to security breaches or system malfunction can be detected using these techniques.

Keywords: Anomaly detection, endpoint security, unsupervised machine learning.

AMS Subject Classification: 68T05, 68M25, 68W40.

Corresponding author: Kamran N. Asgarov, Department of Engineering Mathematics and Artificial Intelligence, Azerbaijan Technical University, PhD Candidate, Baku, Azerbaijan, Tel.: +994 50 541 54 19, e-mail: kamran.asgarov.n@student.aztu.edu.az

Received: 20 October 2024; Revised: 28 November 2024; Accepted: 5 December 2024;

Published: 25 December 2024.

1 Introduction

Nowadays, information systems are becoming more complex, and thus the level of related security risks, including monitoring of network events and endpoint health, is correspondingly growing. An endpoint is a remote computing device that communicates back and forth with a network to which it is connected. Some examples of endpoints are desktops, laptops, smartphones, servers, workstations and Internet-of-things (IoT) devices - like security cameras, smart thermostats and smart home assistants. For a long time, monitoring of endpoint health has been done using signature-based antivirus systems. However, these methods have begun to lose their effectiveness (Patcha & Park, 2007). Firstly, they become helpless against zero-day threats because there is no known signature; secondly, systems based on these methods require regular updates to maintain an up-to-date database of malware signatures; and most importantly it is very difficult for such systems to recognize the signature of a malware if it has been modified using obfuscation techniques.

How to cite (APA): Asgarov, K.N., Imamverdiyev, Y.N., & Abutalibov M.M. (2024). Unsupervised machine learning methods for real-time anomaly detection in endpoints. *Journal of Modern Technology and Engineering*, 9(3), 141-155 <https://doi.org/10.62476/jmte93141>

Malwares are constantly evolving and creating new threats. It is important to have security solutions that can recognize these new threats without the need of regular updates. One of the most effective methods that can be helpful to solve this problem is machine learning methods. These methods can monitor the endpoint and many of its systems, including network activity, or the activity of different hardware devices and learn the behaviour of the endpoint.

An anomaly is anything that deviates from what is standard or expected (Hawkins, 1980). The process of finding such unexpected and non-standard behaviours in an endpoint we call anomaly detection. There are many anomaly detection algorithms, such as: statistical methods, rule-based methods, supervised and unsupervised machine learning methods, deep learning methods etc. However not all these methods can be useful in endpoint anomaly detection, the main criterias for the effectiveness of an algorithm are its performance, memory and CPU consumption, detection accuracy and possibility of continuous learning.

Some studies have explored machine learning techniques for intrusion detection systems for IoT endpoint devices. For instance, Sai Kiran et al. (2020) developed machine learning classifier such as Support Vector Machines and AdaBoost to identify attacks in IoT network. Another study by Vinayakumar et al. (2019) highlighted the potential of models such as Convolutional Neural Networks, Autoencoders and Recurrent Neural Networks to improve scalability and reliability of cyber threat detectors.

Many of these methods are dependent on supervised learning, which requires labeled data for training. In real world scenarios, endpoint telemetry data is not labelled, which makes it difficult to use those methods for real-time anomaly detection. Unsupervised machine learning is an optimal approach for anomaly detection due to the fact that it does not require labeled data for training.

The objective of this paper is to provide a comprehensive review and analysis of four unsupervised machine learning methods for real-time anomaly detection in endpoints: Isolation Forest, One-Class Support Vector Machines (SVM), Gaussian Mixture Models (GMM), and Local Outlier Factor (LOF). By evaluating these methods on real endpoint telemetry data, we aim to determine their suitability for detecting anomalies in endpoint devices and to provide insights into their performance characteristics.

2 Background

This section delves into key definitions, concepts and terminologies that form the basis of our study.

Definition 1 (Endpoint). An **endpoint** is a computing device or node that is connected to a network and communicates back and forth with the network to which it is connected.

Definition 2 (Anomaly). Given a dataset $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, an **anomaly** $\mathbf{x}_i \in \mathcal{X}$ is a data point that significantly differs from the majority of the data according to a defined metric or model.

In endpoint security, anomalies may indicate unusual system behaviour caused by malicious activities, hardware failures, software bugs or any other unexpected events. Anomalies aren't necessarily malicious, but they can also be related to endpoint behaviour not seen before.

Definition 3 (Anomaly Detection). **Anomaly detection** is the process of identifying anomaly data points in a dataset.

Definition 4 (Real-Time Anomaly Detection). **Real-time anomaly detection** is the process of identifying anomaly data points in a dataset as as they occur, with minimal delay.

In the context of endpoint security, this entails the continuous monitoring of the status of the endpoint, coupled with the analysis of new data for any anomalies that may be identified.

Definition 5 (Endpoint Telemetry). **Endpoint Telemetry** is data that is continuously collected from the endpoint. This data can include logs, metrics and other types of information that reflects the state of the endpoint.

In this paper, the most basic endpoint metrics were taken as telemetry. At the same time, these basic indicators are the most important ones on monitoring endpoints:

- **CPU usage** (x_1): The percentage of CPU resources utilized by the endpoint.
- **Memory usage** (x_2): The amount of memory (RAM) consumed by the endpoint.
- **Video memory usage** (x_3): The memory used by the graphics processing unit (GPU).
- **Disk read/write rate** (x_4, x_5): The rate of data read from or written to the disk.
- **Network traffic** (x_6, x_7): The volume of data transmitted over the network.
- **Process count** (x_8): The number of active processes running on the endpoint.
- **Average process CPU usage** (x_9): The average CPU usage of active processes.
- **Average process memory usage** (x_{10}): The average memory usage of active processes.
- **Total active connections** (x_{11}): The number of active network connections.
- **Count of unique IPs** (x_{12}): The count of unique IP addresses connected to the endpoint.

These indicators form a multidimensional data point $\mathbf{x} = [x_1, x_2, \dots, x_{12}]^\top \in \mathbb{R}^{12}$ for each time interval. The data is stored in a structured format and preprocessed to handle missing values and normalise scales (Dempster et al., 1977).

2.1 Unsupervised Machine Learning for Real-Time Anomaly Detection

In real world scenarios, endpoint telemetry data is not labelled, making it difficult to use supervised machine learning for anomaly detection. Unsupervised machine learning is an optimal approach for anomaly detection due to the fact that it does not require labeled data for training. Unsupervised methods analyse the entire dataset and identify patterns in it, then based on these patterns determines the data points that least fit the pattern and labels these data points as anomalies.

Definition 6 (Unsupervised Learning). **Unsupervised machine learning** is a type of machine learning that learns from data without human supervision. Unlike supervised learning unsupervised machine learning models are given unlabeled data and allowed to discover patterns and insights without any explicit guidance or instruction.

Unsupervised machine learning methods are also optimal for real-time anomaly detection due to their working principle (Tan et al., 2011). After continuous training of a model for a certain period of time (e.g 1 month), the model establishes a baseline pattern of normal behaviour (Bifet & Gavaldà, 2009). Subsequently, as new data is occurred, the model can compare it against its learned patterns, efficiently identifying deviations that suggest potential anomalies.

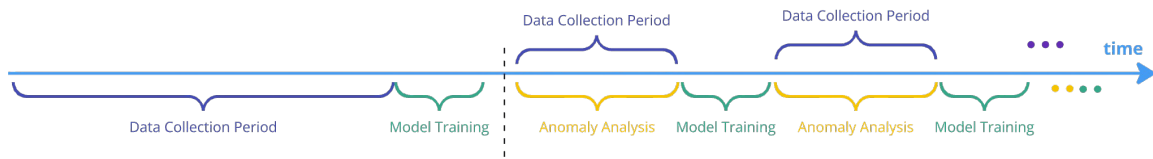


Figure 1: Timeline of anomaly detection methodology.

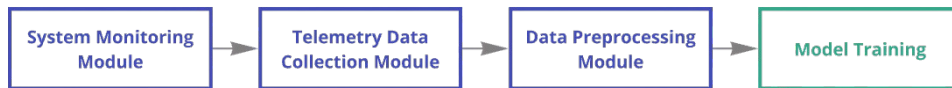


Figure 2: Training period.

2.2 Continuous Anomaly Detection Methodology

This paper puts forth a methodology for the continuous detection of anomalies, as illustrated in Figure 1. The methodology comprises two principal phases, delineated by a dashed line.

The first phase consists of two periods: data collection period, which includes the collection of endpoint telemetry data from the system monitoring module followed by preprocessing of collected data, and the model training period (Figure 2).

Following initial training phase, the model has already developed patterns based on the data collection period data. These patterns enable the model to recognise anomalies in preprocessed data points as new data occur in telemetry collection module. Each new data point is not only analysed by the anomaly detection module, but also stored for the upcoming retraining phase of the model (Figure 3).

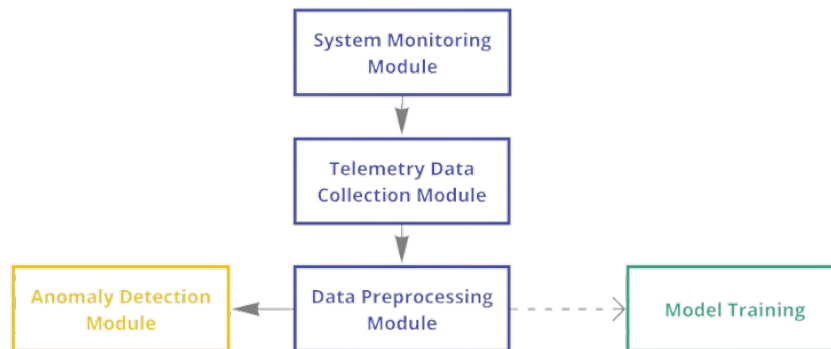


Figure 3: Detection period.

It is important to note that the data collection time in the first period of the paper’s anomaly detection methodology can be called a hyperparameter of the whole methodology, as it can be customised depending on the endpoint’s requirements. In dynamic endpoints such as laptops and personal computers, which may have a lot of different behaviours, this time can be long (4-7 weeks), because for successful analysis it is necessary to collect data on all possible behaviours of the endpoint before starting the analysis for anomalies. In the case of less dynamic endpoints, such as those represented by the IoT devices, the data collection period can be reduced to a shorter timeframe (3-12 days). This is due to the fact that such endpoints exhibit a limited range of behavioural patterns, which can be more efficiently captured within a shorter data collection window.

Another hyperparameter of the methodology is the data collection time in the anomaly

detection period. All stored data points after a certain period of time are passed to the model for retraining. This time should be shorter than the initial time for the main training phase of the model. Furthermore, it is advisable to consider the level of dynamism of the endpoint on training model.

Model training time in Figure 1 shown large than it will be in real world application. This was done for the purpose of good and clear illustration. In real applications this time will be longer in the first period when the model is trained on a large amount of data from first main data collection period, and shorter when it is retrained on a smaller amount of data. It is important to note that this time strongly depends on the training model. Training time will different significantly on different models. Therefore, it is very important to chose the right models for the training.

3 Methods

In this section of the paper, different unsupervised machine learning methods are discussed and their suitability for the endpoint anomaly detection is analysed. The methods described in this section were tested using all given features on real data that was collected from an endpoint within one hour. During this period, telemetry from the endpoint was collected at 10-seconds intervals, then it was preprocessed and stored for model training. Telemetry dataset is consists of a total of 354 data points. The first 283 data points were selected as train data, and the remaining 71 for the test. Dataset can be found on author’s GitHub repository (Asgarov, 2024).

To test the effectiveness of the described methods, it is necessary to add a certain number of anomalies to the test data. But anomalies in itself is a data points that significantly differs from other data points by definition, for example, it can be a strong increase or decrease in the metrics of a data point, or a strong change in the proportions of the data point metrics. In the test data of this paper, 53 data points were simulated by increasing some of the metrics of data points by a certain percentage. Detailed values of increase percentage also can be found on author’s GitHub repository (Asgarov, 2024).

Therefore due to the simulation over the data we have clear labels for each data point in the dataset. Consequently, during the evaluation of the methods, we can use the evaluation methods that are usually used for supervised machine learning methods – confusion matrix and classification report.

Definition 7 (Confusion Matrix). A **confusion matrix** is a table that summarizes the performance of a classification model by showing the counts of actual versus predicted classifications. For a binary classification problem, the confusion matrix is a 2×2 matrix with the following structure:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix},$$

where:

- **True Positive (TP):** The number of data points that are correctly predicted as the positive class.
- **False Positive (FP):** The number of data points that are incorrectly predicted as the positive class (but are actually negative).
- **False Negative (FN):** The number of data points that are incorrectly predicted as the negative class (but are actually positive).
- **True Negative (TN):** The number of data points that are correctly predicted as the negative class.

While the confusion matrix provides a detailed breakdown of the model's predictions compared to the actual classifications, it can be further analyzed to derive more insightful performance metrics. To quantify the model's effectiveness in terms of precision, recall, and the balance between them, a classification report is used.

Definition 8 (Classification Report). A **classification report** is a summary that evaluates the performance of a classification model by presenting key metrics for each class in the dataset. For binary or multi-class classification problems, the report includes the following metrics for each class:

- **Precision:** The ratio of true positives to the sum of true positives and false positives. It measures the accuracy of positive predictions for a class.
- **Recall** (Sensitivity): The ratio of true positives to the sum of true positives and false negatives. It assesses the model's ability to find all relevant instances of a class.
- **F1-Score:** The harmonic mean of precision and recall. It provides a balance between precision and recall, especially useful when dealing with imbalanced datasets.
- **Support:** The number of actual occurrences of each class in the dataset.

3.1 Isolation Forest

Isolation Forest is an ensemble anomaly detection algorithm that isolates anomalies instead of profiling normal data points (Liu et al., 2000). By recursively partitioning the data using random feature selection and split values, it efficiently identifies anomalies as data points that require fewer splits to isolate. We formally define the Isolation Forest as follows.

Definition 9 (Isolation Forest). An Isolation Forest consists of t isolation trees built from subsamples of the data. Each isolation tree T is constructed by recursively partitioning a dataset X using randomly selected features and split values until each data point is isolated. The anomaly score for an \mathbf{x} is calculated based on the average path length $E[h(\mathbf{x})]$ across all trees:

$$s(\mathbf{x}, n) = 2^{-\frac{E[h(\mathbf{x})]}{c(n)}}, \quad (1)$$

where $c(n)$ is the average path length of searches in a Binary Search Tree (Cormen et al., 2001):

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n},$$

and $H(i)$ is the harmonic number defined as $H(i) = \ln(i) + \gamma$ with $\gamma \approx 0.57721\dots$ being the Euler-Mascheroni constant (Abramowitz et al., 1965).

The anomaly score $s(\mathbf{x}, n)$ quantifies the degree of abnormality of \mathbf{x} , with higher scores indicating higher likelihoods of being anomalies.

3.1.1 Algorithm Details

Isolation Forest operates under the principle that anomalies are "few and different" and thus more susceptible to isolation through random partitioning. The algorithm comprises two main processes: building the isolation trees and calculating anomaly scores.

Each isolation tree is constructed by recursively partitioning the data. Starting with a subsample X' of size ψ randomly selected from the dataset X , the tree is built as follows. At each node, a feature q is randomly selected from the set of features. A split value p is then randomly chosen between the minimum and maximum values of feature q in the data at that

node. The data is partitioned into two child nodes based on whether the feature value is less than or greater than or equal to p . This process continues recursively for each child node until one of the termination conditions is met: either the node contains only one data point, or the tree reaches a predefined maximum depth.

The anomaly score for an \mathbf{x} is calculated using the average path length $E[h(\mathbf{x})]$ across all trees. The path length $h_j(\mathbf{x})$ is the number of edges traversed from the root node to the terminating node in tree T_j . The average path length is computed as $E[h(\mathbf{x})] = \frac{1}{t} \sum_{j=1}^t h_j(\mathbf{x})$. The anomaly score is then determined using the formula (1).

3.1.2 Experimental Results

Method was tested using python library scikit-learn (Pedregosa et al., 2011) with the data discussed at the beginning of the section 3. Jupyter notebooks of the this method and all other methods can be found on author’s GitHub repository (Asgarov, 2024). Tests were done using different values of hyperparameters ψ which is a subsample size used to construct each isolation tree described in section 3.1.1 and t which represent the number of isolation trees in the forest described in definition 9. Method results for different values of hyperparameters are shown in Table 1.

Table 1: Classification metrics for Isolation Forest with different values of t and ψ .

t	ψ	Accuracy	Precision	Recall	F1-Score
100	100	0.9577	1.0000	0.8333	0.9091
50	100	0.9577	1.0000	0.8333	0.9091
100	75	0.9437	1.0000	0.7778	0.8750
100	50	0.9296	1.0000	0.7222	0.8387
50	75	0.9296	1.0000	0.7222	0.8387
50	50	0.9155	1.0000	0.6667	0.8000
50	20	0.9014	1.0000	0.6111	0.7586
25	40	0.8873	1.0000	0.5556	0.7143
50	5	0.8310	0.6500	0.7222	0.6842
25	10	0.8732	1.0000	0.5000	0.6667

Best result of this method was F1-Score of 0.91. To visualise this result a pair plot was constructed (Figure 4). The pair plot highlights how specific pairs of features interact. For instance, there is a notable separation between normal and anomalous data when comparing CPU usage with RAM usage, suggesting that anomalies affect both CPU and RAM usage in a distinguishable way.

The confusion matrix of the method’s best results is as follows:

$$\begin{bmatrix} 53 & 0 \\ 3 & 15 \end{bmatrix}.$$

The confusion matrix shows that the model correctly identified 53 anomalies (True Negatives) and 15 normal points (True Positives). It produced 3 false negatives (missed anomalies) and no false positives (normal points classified as anomalies). This demonstrates excellent precision for normal data and a high specificity for anomalies, although the model has a slightly lower recall for anomalies due to the missed cases. Overall, the Isolation Forest method shows strong performance with minimal misclassifications.

3.2 One-Class Support Vector Machines

The One-Class Support Vector Machine (One-Class SVM) is an unsupervised learning algorithm derived from the traditional Support Vector Machine framework. While traditional SVMs are

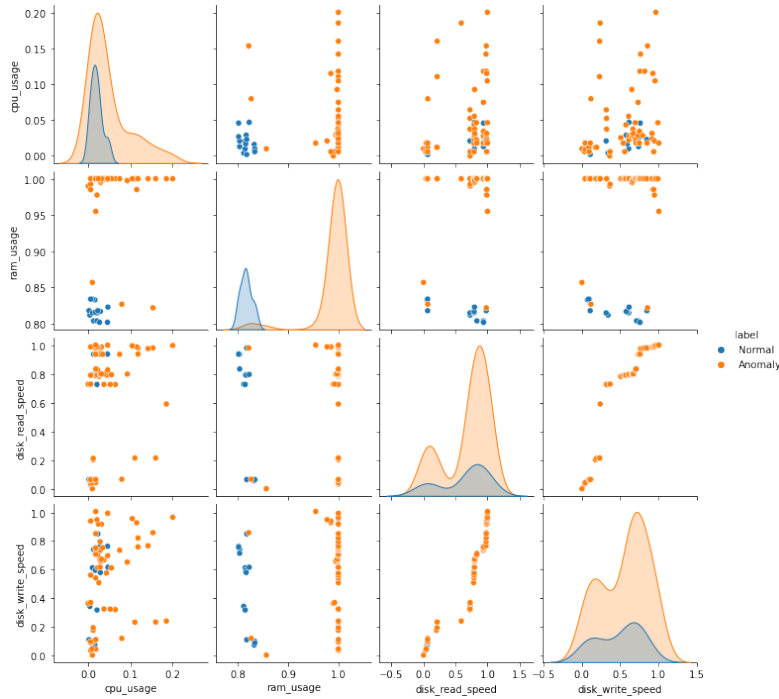


Figure 4: Isolation Forest Pair Plot Visualisation.

supervised learning methods used for classification and regression tasks with labeled data, One-Class SVM is specifically designed for anomaly detection in unlabeled datasets (Schölkopf et al., 2001). It aims to learn a decision function that identifies regions in the input space where the probability density of the data resides, thereby enabling the detection of outliers or anomalies that fall outside this region.

Definition 10 (One-Class SVM). Given a set of training data $\{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^D$, the One-Class SVM seeks a hyperplane in the feature space \mathcal{F} , induced by the mapping $\phi : \mathbb{R}^D \rightarrow \mathcal{F}$, that best separates the data from the origin with maximum margin. This is formulated as the following optimization problem:

$$\min_{\mathbf{w}, \rho, \xi_i} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu N} \sum_{i=1}^N \xi_i - \rho \quad \text{subject to} \quad (\mathbf{w}^\top \phi(\mathbf{x}_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, N,$$

where $\mathbf{w} \in \mathcal{F}$ is the normal vector to the hyperplane, $\rho \in \mathbb{R}$ is the offset from the origin, $\xi_i \geq 0$ are slack variables allowing for some errors, and $\nu \in (0, 1]$ is a parameter controlling the trade-off between the fraction of outliers and the margin of the hyperplane.

The One-Class SVM constructs a decision function $f(\mathbf{x})$ that indicates whether a new data point \mathbf{x} is similar to the training data (normal) or not (anomalous). The decision function is given by:

$$f(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^N \alpha_i K(\mathbf{x}_i, \mathbf{x}) - \rho \right),$$

where $K(\mathbf{x}_i, \mathbf{x})$ is the kernel function, and α_i are the solution to the dual optimization problem.

3.2.1 Algorithm Details

The One-Class SVM algorithm involves solving an optimization problem to find the hyperplane that best separates the data from the origin in the feature space. By introducing Lagrange

multipliers α_i , the problem can be transformed into its dual form:

$$\max_{\alpha_i} -\frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \text{ subject to } 0 \leq \alpha_i \leq \frac{1}{\nu N}, \quad \sum_{i=1}^N \alpha_i = 1.$$

The kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ computes the inner product in the feature space, allowing the algorithm to handle non-linear separations without explicitly mapping the data to a higher-dimensional space.

3.2.2 Experimental Results

Method was tested using python library scikit-learn with the data discussed at the beginning of the section 3. Tests were done using different kernel functions and different values of ν . Method results for different values of hyperparameters are shown in Table 1.

Table 2: Classification metrics for One-Class SVM with different kernel functions and ν values.

Kernel Function	ν	Accuracy	Precision	Recall	F1-Score
$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-0.100\ \mathbf{x}_i - \mathbf{x}_j\ ^2)$	0.001	0.9859	1.0000	0.9444	0.9714
$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-0.010\ \mathbf{x}_i - \mathbf{x}_j\ ^2)$	0.010	0.9859	1.0000	0.9444	0.9714
$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-0.500\ \mathbf{x}_i - \mathbf{x}_j\ ^2)$	0.100	0.9718	1.0000	0.8889	0.9412
$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$	0.001	0.2535	0.2535	1.0000	0.4045
$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(0.100 \mathbf{x}_i^\top \mathbf{x}_j)$	0.001	0.2535	0.2535	1.0000	0.4045
$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$	0.010	0.2394	0.2429	0.9444	0.3864
$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(0.001 \mathbf{x}_i^\top \mathbf{x}_j)$	0.050	0.2113	0.2206	0.8333	0.3488
$K(\mathbf{x}_i, \mathbf{x}_j) = (0.500 \mathbf{x}_i^\top \mathbf{x}_j + 1)^3$	0.050	0.1831	0.1970	0.7222	0.3095
$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$	0.050	0.1831	0.1970	0.7222	0.3095
$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(0.500 \mathbf{x}_i^\top \mathbf{x}_j)$	0.050	0.1831	0.1970	0.7222	0.3095

Best result of this method was F1-Score of 0.91. To visualise this result a pair plot was constructed (Figure 5).

The confusion matrix of the method's best results is as follows:

$$\begin{bmatrix} 53 & 0 \\ 1 & 17 \end{bmatrix}$$

The confusion matrix for this method shows 53 anomalies correctly identified (True Negatives), 17 normal points correctly classified (True Positives), 1 false negatives (missed anomalies) and 0 false positive (normal point classified as an anomaly). Compared to the Isolation Forest method, One-Class SVM demonstrates a slight improvement in precision with fewer false negatives.

3.3 Gaussian Mixture Models

Gaussian Mixture Models (GMMs) are a probabilistic approach for representing the presence of subpopulations within an overall population without requiring that each data point belong exclusively to a single subpopulation. In the context of anomaly detection, GMMs are utilized to model the normal behavior of a system by capturing the underlying distribution of the data (Dempster et al., 1977). Observations that do not conform to this learned distribution are flagged as anomalies.

Definition 11 (Gaussian Mixture Model). A Gaussian Mixture Model represents the probability density function (PDF) of a dataset as a weighted sum of K Gaussian (normal) component densities. The PDF of the GMM is given by:

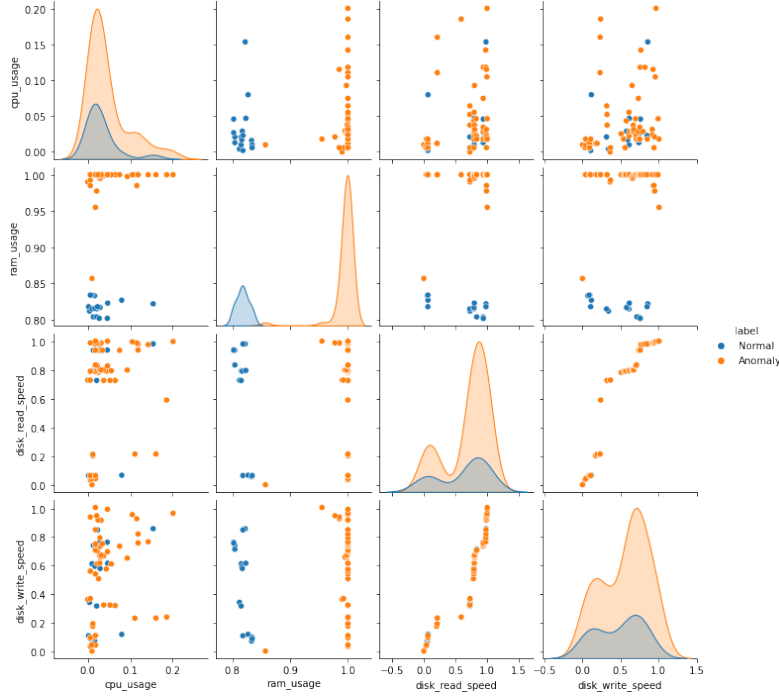


Figure 5: One-Class SVM Pair Plot Visualisation.

$$p(\mathbf{x} | \Theta) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

where:

- $\mathbf{x} \in \mathbb{R}^D$ is a D -dimensional data point,
- $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ represents the set of parameters of the model,
 - π_k is the mixing coefficient for component k , such that $\sum_{k=1}^K \pi_k = 1$ and $0 \leq \pi_k \leq 1$,
 - $\boldsymbol{\mu}_k \in \mathbb{R}^D$ is the mean vector of component k ,
 - $\boldsymbol{\Sigma}_k \in \mathbb{R}^{D \times D}$ is the covariance matrix of component k ,
- $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the multivariate Gaussian density function:

$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right).$$

The goal is to estimate the parameters Θ that maximize the likelihood of the observed data $\{\mathbf{x}_i\}_{i=1}^N$.

3.3.1 Parameter Estimation with Expectation-Maximization

The Expectation-Maximization (EM) algorithm is employed to iteratively estimate the parameters of the GMM. The EM algorithm consists of two main steps: the Expectation step (E-step) and the Maximization step (M-step).

In the **Initialization** phase, initial guesses are made for the parameters $\Theta^{(0)} = \{\pi_k^{(0)}, \boldsymbol{\mu}_k^{(0)}, \boldsymbol{\Sigma}_k^{(0)}\}$ for $k = 1, \dots, K$

$$\gamma_{ik}^{(t)} = \frac{\pi_k^{(t-1)} \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k^{(t-1)}, \boldsymbol{\Sigma}_k^{(t-1)})}{\sum_{j=1}^K \pi_j^{(t-1)} \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_j^{(t-1)}, \boldsymbol{\Sigma}_j^{(t-1)})}$$

In the **M-Step**, the parameters are updated using the responsibilities computed in the E-step:

$$\pi_k^{(t)} = \frac{N_k}{N}, \quad \boldsymbol{\mu}_k^{(t)} = \frac{1}{N_k} \sum_{i=1}^N \gamma_{ik}^{(t)} \mathbf{x}_i, \quad \boldsymbol{\Sigma}_k^{(t)} = \frac{1}{N_k} \sum_{i=1}^N \gamma_{ik}^{(t)} (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t)}) (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t)})^\top$$

where $N_k = \sum_{i=1}^N \gamma_{ik}^{(t)}$ is the effective number of data points assigned to component k .

The **Convergence** criterion is based on the log-likelihood of the data given the model parameters:

$$\mathcal{L}(\Theta) = \sum_{i=1}^N \ln \left(\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

The EM algorithm iteratively performs the E-step and M-step until the change in log-likelihood $\mathcal{L}(\Theta)$ between iterations falls below a predefined threshold, indicating convergence.

After training, the GMM models the probability density function of the normal data. For a new data point \mathbf{x}_{new} , the likelihood $p(\mathbf{x}_{\text{new}} | \Theta)$ is computed using the GMM. An anomaly score is defined as the negative log-likelihood:

$$S(\mathbf{x}_{\text{new}}) = -\ln p(\mathbf{x}_{\text{new}} | \Theta)$$

A threshold τ is set to classify observations as normal or anomalous. If $S(\mathbf{x}_{\text{new}}) > \tau$, the observation is considered anomalous. The threshold τ can be determined based on the desired false positive rate or by analyzing the distribution of anomaly scores on a validation dataset.

3.3.2 Experimental Results

Method was tested using python library scikit-learn with the data discussed at the beginning of the section 3. Tests were done using different covariance types and different values of τ and K . Method results for different values of hyperparameters are shown in Table 3.

Table 3: Classification metrics for Gaussian Mixture Model with different hyperparameters.

K	Covariance Type	τ	Accuracy	Precision	Recall	F1-Score
1	full	0.5	1.0000	1.0000	1.0000	1.0000
4	diag	0.5	1.0000	1.0000	1.0000	1.0000
4	tied	2.0	1.0000	1.0000	1.0000	1.0000
1	spherical	2.5	0.9577	1.0000	0.8333	0.9091
1	spherical	2.0	0.9296	1.0000	0.7222	0.8387
1	spherical	1.5	0.9014	1.0000	0.6111	0.7586
1	spherical	1.0	0.8873	1.0000	0.5556	0.7143
1	spherical	0.5	0.8873	1.0000	0.5556	0.7143

The confusion matrix of the method's best results is as follows:

$$\begin{bmatrix} 53 & 0 \\ 0 & 18 \end{bmatrix}$$

The confusion matrix for this method shows 53 anomalies correctly identified (True Negatives) and 18 normal points correctly classified (True Positives), with no false positives (normal

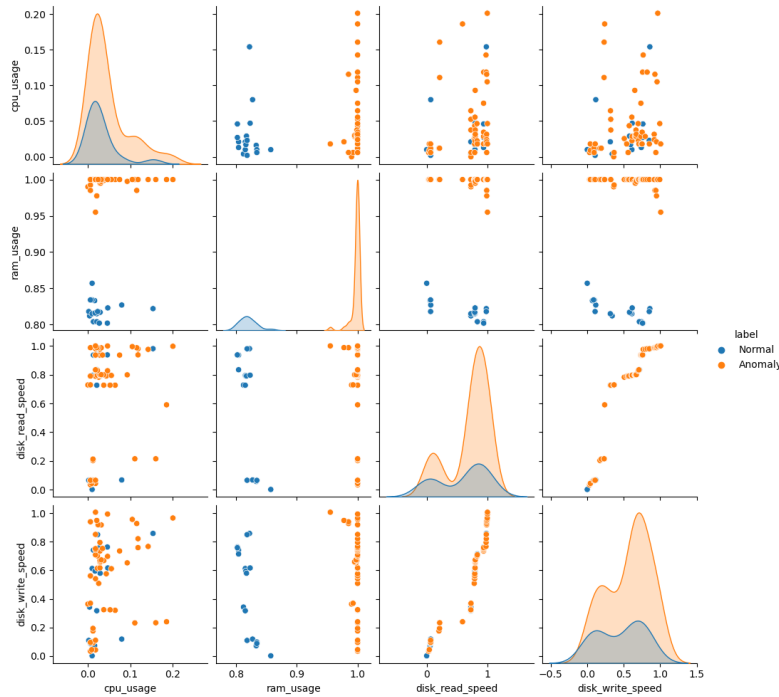


Figure 6: Gaussian Mixture Models Pair Plot Visualisation.

points classified as anomalies) and no false negatives (missed anomalies). Compared to the Isolation Forest and One-Class SVM methods, the Gaussian Mixture Model (GMM) successfully detected all anomalies and achieved perfect classification without any misclassifications, indicating both strong recall and precision.

3.4 Local Outlier Factor

The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection method that identifies anomalies based on the concept of local density (Breunig et al., 2000). It measures the degree to which a data point is isolated from its surrounding neighborhood. In the context of endpoint anomaly detection, LOF can detect observations that have a significantly lower density than their neighbors, indicating abnormal behavior.

Definition 12 (Local Outlier Factor). Given a dataset $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \subset \mathbb{R}^D$ and a positive integer parameter k specifying the number of nearest neighbors, the Local Outlier Factor of a data point \mathbf{x} , denoted as $\text{LOF}_k(\mathbf{x})$, quantifies the extent to which \mathbf{x} is an outlier. It is defined based on the following concepts:

1. **k -distance of \mathbf{x}** : The k -distance of \mathbf{x} , denoted as $k\text{-distance}(\mathbf{x})$, is the distance between \mathbf{x} and its k -th nearest neighbor.
2. **k -distance neighborhood of \mathbf{x}** : Denoted as $N_k(\mathbf{x})$, it includes all points $\mathbf{y} \in \mathcal{X}$ such that the distance $d(\mathbf{x}, \mathbf{y}) \leq k\text{-distance}(\mathbf{x})$.
3. **Reachability distance**: The reachability distance between \mathbf{x} and \mathbf{y} is defined as:

$$\text{reach-dist}_k(\mathbf{x}, \mathbf{y}) = \max\{k\text{-distance}(\mathbf{y}), d(\mathbf{x}, \mathbf{y})\}$$

4. **Local reachability density**: The local reachability density of \mathbf{x} is given by:

$$\text{lrd}_k(\mathbf{x}) = \left(\frac{\sum_{\mathbf{y} \in N_k(\mathbf{x})} \text{reach-dist}_k(\mathbf{x}, \mathbf{y})}{|N_k(\mathbf{x})|} \right)^{-1}$$

5. **Local Outlier Factor:** The LOF of \mathbf{x} is defined as:

$$\text{LOF}_k(\mathbf{x}) = \frac{\sum_{\mathbf{y} \in N_k(\mathbf{x})} \frac{\text{lr}_k(\mathbf{y})}{\text{ld}_k(\mathbf{x})}}{|N_k(\mathbf{x})|}$$

A higher value of $\text{LOF}_k(\mathbf{x})$ indicates that the point \mathbf{x} has a significantly lower density than its neighbors, suggesting that it is an outlier.

3.4.1 Algorithm Details

Implementing LOF involves several steps. First, compute the pairwise distances between data points to identify the k -nearest neighbors for each point. Then, calculate the reachability distances for each point based on its k -nearest neighbors. Next, compute the local reachability density for each point using the reachability distances. Subsequently, calculate the LOF score for each point by comparing its local reachability density to those of its neighbors. Finally, classify points as normal or anomalous based on the LOF scores and the chosen threshold τ .

Careful attention must be paid to computational efficiency, especially when dealing with large datasets. Techniques such as indexing structures (e.g., k -d trees) or approximate nearest neighbor algorithms can be used to reduce computation time.

Selecting the parameter k is a critical aspect of applying the LOF algorithm effectively. One approach is to experiment with multiple values of k and evaluate the algorithm's performance on validation data. Considering the average density of the dataset and the expected size of local neighborhoods can inform the choice of k . In practice, values of k ranging from 10 to 30 are commonly used, but the optimal value depends on the specific characteristics of the data.

3.4.2 Experimental Results

Method was tested using python library scikit-learn with the data discussed at the beginning of the section 3. To visualise results a pair plot was constructed (Figure 7).

Method was tested using python library scikit-learn with the data discussed at the beginning of the section 3. Tests were done using different values of τ and k . Method results for different values of hyperparameters are shown in Table 3.

Table 4: Classification metrics for Local Outlier Factor with different hyperparameters.

k	τ	Accuracy	Precision	Recall	F1-Score
30	0.15	1.0000	1.0000	1.0000	1.0000
20	0.15	0.9859	1.0000	0.9444	0.9714
30	0.30	0.9718	1.0000	0.8889	0.9412
15	0.15	0.9718	1.0000	0.8889	0.9412
15	0.30	0.9577	1.0000	0.8333	0.9091
30	0.40	0.9437	1.0000	0.7778	0.8750
25	0.40	0.9296	1.0000	0.7222	0.8387
10	0.30	0.9296	1.0000	0.7222	0.8387
20	0.40	0.9155	1.0000	0.6667	0.8000
15	0.40	0.9014	1.0000	0.6111	0.7586

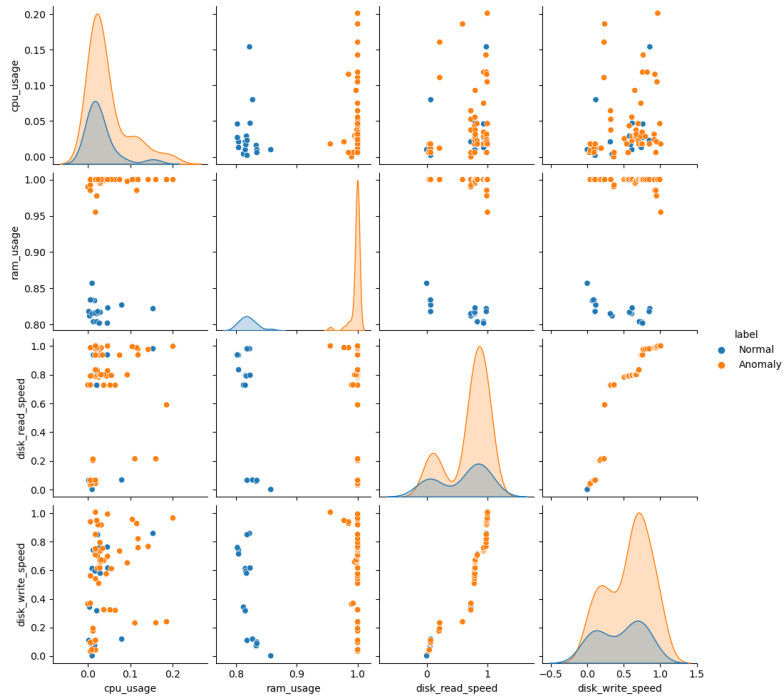


Figure 7: Local Outlier Factor Pair Plot Visualisation.

The confusion matrix of the method’s best results is as follows:

$$\begin{bmatrix} 53 & 0 \\ 0 & 18 \end{bmatrix}$$

The Local Outlier Factor (LOF) method achieved perfect classification, correctly identifying all 53 anomalies (True Negatives) and all 18 normal data points (True Positives), resulting in no false positives or false negatives.

4 Conclusion

This paper investigated unsupervised machine learning techniques for real-time anomaly detection in endpoints using a novel continuous anomaly detection methodology. Paper provided a detailed description of the methodology and its core working principles. Four unsupervised machine learning methods were considered as part of the detection framework: Isolation Forest, One-Class Support Vector Machines (SVM), Gaussian Mixture Models (GMM), and Local Outlier Factor (LOF).

All methods were trained using real endpoint telemetry data collected over a one-hour period. Part of this telemetry was modified to simulate anomalies and used as test data. Pair plot visualizations were utilized to illustrate the separation between anomalies and normal data across different features. Furthermore, confusion matrices and classification reports were calculated for each method to evaluate their performance comprehensively.

Experimental results identified that both **Gaussian Mixture Model (GMM)** and **Local Outlier Factor (LOF)** methods achieved the highest overall accuracy, precision, recall and f1-score. Those methods detected anomalies with no false positives or false negatives, making them ideal choice for real-time anomaly detection monitoring systems. **Isolation Forest** and **One-Class SVM** method also showed good performance with low false positive rate and remain viable options.

References

- Abramowitz, M., Stegun, I.A., & Miller, D. (1965). Handbook of Mathematical Functions With Formulas, Graphs and Mathematical Tables (National Bureau of Standards Applied Mathematics Series No. 55). In *Journal of Applied Mechanics* (pp. 239-239). <https://doi.org/10.1115/1.3625776>
- Asgarov, K.N. (2024). Unsupervised ML anomaly detection. *GitHub Repository* <https://github.com/tivole/unsupervised-ml-anomaly-detection>
- Breunig, M.M., Kriegel, H.P., Ng, R.T., & Sander, J. (2000). LOF: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (pp. 93-104). <https://doi.org/10.1145/342009.335388>
- Bifet, A., Gavaldà, R. (2009). Adaptive learning from evolving data streams. In *Advances in Intelligent Data Analysis VIII (IDA 2009)* (Vol. 5772, pp. 249-260). Springer. https://doi.org/10.1007/978-3-642-03915-7_22
- Cormen, T.H., Leiserson, C.E., Rivest, R.L., & Stein, C. (2001). *Introduction To Algorithms*. India: MIT Press.
- Dempster, A.P., Laird, N.M., & Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1), 1-22. <https://doi.org/10.1111/j.2517-6161.1977.tb01600.x>
- Hawkins, D.M. (1980). *Identification of outliers*. Chapman and Hall.
- Liu, F.T., Ting, K.M., & Zhou, Z.H. (2008). Isolation forest. In *Proceedings of the 2008 IEEE International Conference on Data Mining* (pp. 413-422). IEEE. <https://doi.org/10.1109/ICDM.2008.17>.
- Patcha, A., Park, J.M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12), 3448-3470. <https://doi.org/10.1016/j.comnet.2007.02.001>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830. <https://www.jmlr.org/papers/v12/pedregosa11a.html>
- Sai Kiran K.V.V.N.L., Kamakshi Devisetty. R.N., Pavan Kalyan, N., Mukundini, K., & Karthi, R. (2020). Building a Intrusion Detection System for IoT Environment using Machine Learning Techniques. In *Procedia Computer Science* (Vol. 171, pp. 2372-2379). <https://doi.org/10.1016/j.procs.2020.04.257>
- Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., & Williamson, R.C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7), 1443-1471. <https://doi.org/10.1162/089976601750264965>
- Tan, S.C., Ting, K.M., & Liu, F.T. (2011). Fast anomaly detection for streaming data. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011)* (pp. 1511-1516).
- Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A. & Venkatraman, S. (2019) Deep Learning Approach for Intelligent Intrusion Detection System, In *IEEE Access*, (vol. 7, pp. 41525-41550) <https://doi.org/10.1109/ACCESS.2019.2895334>