# Anomaly Detection in Cyber Security using Machine Learning (2025)

Andrew M. Stephenson, Ethan Busby, Mason Wolf

*Abstract*– This document provides an overview for the concept of anomaly detection in cyber security using machine learning. It will first introduce the topic by providing a basic definition. This is followed by describing the types of anomalies present in such cases, along with data sources of where to find them. Preprocessing and cleaning the data is also discussed, as it is a necessary step to prevent false detections. Machine learning models, as well as ways to interpret results, will also be discussed, as this is the main focus of the paper. Wrapping up will be delving into the current findings and trends in the modern day and age, followed by the potential research and directions in the future.

## I. Introduction

Anomaly detection is an imperative process in several fields, none more so than cyber security. Since cyber security is a prevalent field in today's age, where cyber attacks are more common than ever before, anomaly detection and innovation to prevent attacks is a necessity. Anomaly detection is used to strengthen security by recognizing unusual patterns and data that indicate potential security breaches or unauthorized access. There are a multitude of benefits from this, including Fraud prevention, DDoS attack prevention, better customer experience, enhanced network security, etc.

With advancements in technology, artificial intelligence and machine learning have significantly improved both the speed and accuracy of anomaly detection. These systems do this by going past the lengths of human capabilities to recognize spikes in data, analyze vast amounts of data, and detect logins from unique places instantaneously. Additionally, the development of a variety of machine learning models allows for greater flexibility in applying anomaly detection across different environments. This adaptability of machine learning models helps ensure organizations can find the most effective approach to keep their information secure from cyber attacks.

This paper will explore everything related to anomaly detection in cyber security, including types of anomalies, data sources for said anomalies, data cleansing methods, machine learning models, interpreting results, current trends, and future directions and research.

## II. Types of Anomalies

Before diving into understanding different machine learning models and how they can be used in anomaly detection, it is first essential to understand the different types of anomalies in cyber security. By doing this, an individual can use their knowledge about anomalies to train machine learning models efficiently and accurately. If someone does not understand anomalies and how to detect them themselves, then they will not be able to train artificial intelligence to do it on its own. There are many variations of different anomalies, but they can be separated into three main groups which are point anomalies, contextual anomalies, and collective anomalies [1].

### A. Point Anomalies

The simplest and easiest anomalies to detect are point anomalies, also known as outliers. In other words, this is where there is some data that is significantly different from the rest of the dataset. Point anomalies can be broken down into two subcategories [2].

The first subcategory is univariate outliers. This outlier refers to where there are data points that differ from the rest of a singular dataset. An example of this would be spikes in network traffic, as shown in figure 1 below. The traffic of the network example is constant apart from the one spike, so it is deemed an outlier. In this example, the spike could indicate malicious activity, so individuals have to be wary when these occur.

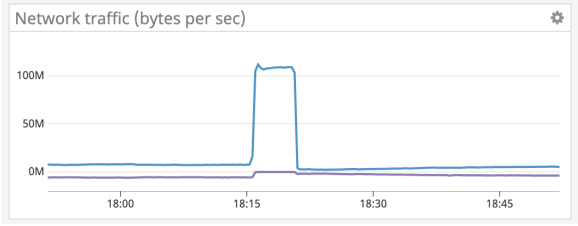


Fig. 1. Example of network traffic spike [3].

The other subcategory of point anomalies is multivariate outliers. Unlike univariate outliers, these deal with looking at multiple variables and seeing how the combination of them deviate from the norm [2]. An example here would be someone logging into a secure network on a new device but also failing to login correctly multiple times. Now these instances separately may not cause much concern, but when paired together, it looks much more unusual.

Various statistical and visualization techniques can be very helpful when it comes to detecting point anomalies. Calculating z-scores can help show how far different data points variate from the mean while interquartile ranges can define boundaries of acceptable data points. Different graphs, such as scatter plots and boxplots, can provide visual representation that allows for outliers to be easily seen when it may not be so easy to see that anomaly in just a dataset.

### B. Contextual Anomalies

Unlike point anomalies, contextual anomalies deal with data points that are abnormal within certain contexts. Essentially, data that looks normal at some point may not look normal at another. To detect this, it ultimately comes down to looking at the relationships in datasets and then seeing when these relationships deviate from normal behavior [4].

One simple example of a contextual anomaly is unusual login behavior. During normal working hours for an employee, it is expected for them to login; however, if there was an instance where this employee logged in, say 2 AM, then this could be flagged as potential suspicious activity. Without the context of time, it may seem like a normal operation, but when that time of day is factored in, something as simple as logging in may raise red flags.

As mentioned previously, looking at the relationships in datasets can help detect contextual anomalies, so using historical data can be of great use. Machine learning models are the best method for understanding past data and using it to find abnormalities, but this paper will discuss more specific models later on.

### C. Collective Anomalies

Similar to contextual anomalies, collective anomalies deal with data points that are abnormal within certain contexts; however, the context with collective anomalies is when a group of data points show abnormal behavior when considered all together. These can be challenging to notice because each individual data point may seem perfectly normal when looked at on its own [5].

An example of a collective anomaly that may be very difficult for a person to notice, especially without the help of AI, is insider threats using small actions. Say there is an employee of some tech company that makes very small, practically unnoticeable changes to secure company information. Each instance of this employee making a change can very easily go unnoticed, but overtime, the collection of these changes may indicate some sort of cyber attack or sabotage.

As shown by the example, machine learning models need to be trained to detect very specific patterns in order to pick up on small details. One potential model that can be used is the Long Short-Term Memory Recurrent Neural Network (LSTM RNN). To introduce briefly, this model is trained on normal historical data. It then uses this to make live predictions about what the data should look like in the present or some set time period in advance. If the data deviates too much from the prediction, then the model can identify it as an anomaly, and then necessary actions can be used to deal with it. This model will be described more in depth later on in this paper [5].

## III. Data Sources for Anomaly Detection

Effective anomaly detection in cybersecurity relies on collecting and analyzing data from various sources to recognize abnormal system behavior. These sources provide important insight into various aspects of computing like network activity, system operations, and user behavior. This allows machine learning models and other tools to recognize a variety of threats against a system whether it be a malware or insider attack. The most commonly used data sources for this task are perimeter network data, internal network data, and distributed and comprehensive network data. By utilizing these data sources together, organizations can detect and respond to anomalies and threats.


Fig 2. An overview of data sources used for security analytics[4].

### A. Perimeter Network Data

Perimeter network data is made up of logs and network traffic information collected at the boundary of an organization's network. The boundary serves as the entry and exit point of a network where internal systems connect to external networks, like the internet. By monitoring this perimeter, organizations can detect various threats before they can move deeper into internal systems.

Network flow events are similar to logs as they record information pertaining to source and destination IP address, ports, protocols, timestamps, and byte counts for every communication session. This data flow is important as it can assist in spotting abnormal traffic patterns, abnormal connections and spikes in data transfer volume. Network flow events are also used to establish a level of normal activity within an organization, allowing deviations to be recognized as possible threats. Unlike full network packet captures, network flow events provide a high level summary of network activity allowing scalable monitoring across large networks [6].

Proxy server logs received from web gateways provide detailed records of a user's web requests. This includes destination URLs, hostnames, HTTP methods and request statuses. By analyzing proxy server logs, an organization can identify employees attempting to access malicious websites and track unusual web usage patterns like high volumes of encrypted traffic or repeated visiting of blocked sites. Proxy logs are primarily used for the detection of insider threats or compromised accounts.

Firewall logs record connection attempts at the perimeter, capturing IPs, protocols, timestamps, and firewall actions. Unusual behavior recorded in a firewall log is typically representative of port scanning and brute-force attacks. Also, unexpected outbound traffic recorded to firewall logs could signal possible malware communicating with outside command-and-control servers. In conjunction with other perimeter data, firewall logs can detect coordinated attacks and generate real time alerts. Advanced firewalls with deep packet inspection can analyze network traffic to distinguish between benign and malicious activity [6].

Full packet captures records all network communications in complete detail (whereas network flow events store summaries). This allows deep visibility into network behavior which is important for post-breach investigations. Full packet capture is sometimes used in a similar notion as network flow data, but since it captures the full packet instead of just the meta data, it is not as efficient for real time threat detection. It requires more processing power and data storage making it largely impractical for large scale organizations with massive amounts of data coming in every second.

Perimeter network data serves as an organization’s first line of defense against cyber threats. By analyzing key network boundary data, organizations can detect suspicious activity and other threats before they gain access to internal infrastructure.

### B. Internal Network Data

Internal network data provides insights into activities inside an organization’s network, where threats have already bypassed the perimeter boundary and its defenses. Via monitoring internal traffic, cybersecurity teams can spot malicious behaviors occurring behind the firewall.

Network flow events and full packet capture not only monitor the boundary but also monitor internal network exchanges. These data sources focus on internal communication between networks and hosts. An internal network flow record can recognize whether a compromised host is abnormally contacting other internal machines, or unusual server activity transmitting data across the network. These activities could be a result of an attacker gaining access and exploiting an organization's network.

Internal network flow data can also be effective at detecting convert exfiltration, where an attacker with internal access sends data over an uncommon protocol or during unusual hours. Full packet capture can then be used to see detailed information on suspicious communications and verify the threat. It can see small things missed in the flow records like

querying a database, unauthorized API calls, or signs of privilege escalation.

Internal networks also gather large amounts of data by recording user and device interactions with core infrastructure. These instances come in the form of DNS and DHCP logs along with email metadata.

DNS logs capture domain name resolution requests from internal hosts and the resulting response. By analyzing patterns in DNS queries, organizations can identify signs of possible malware communication and other cyber threats.

DHCP logs record IP address allocations across the network, allowing organizations to identify anomalies like potentially rogue servers used for man-in-the-middle attacks or IP exhaustion attacks.

Another important internal data source is email metadata. This includes sender, receiver, subject, message, and attachment information. It can help an organization protect against phishing campaigns or compromised accounts. Also, users interacting with external recipients they have never contacted before could be a signal for a social engineering attempt or account takeover.

By using these various internal network data sources, organizations can detect attacks and establish a quick and correct response ensuring strong security across networks.

C. Distributed and Comprehensive Network Data

This category is representative of data collected from individual endpoints like workstations, laptops, and servers. It includes logs that provide detailed views of system behaviors. This data is important for detecting malware that is found on hosts and within individual systems themselves [8].

The largest source in this category comes from system logs. System logs track everything significant that happens on a given device. This includes loading drivers, service start/stop events, system errors, logins, lockouts, crashes, and warnings. These logs are essential for identifying insider threats or compromised accounts, but it is necessary for continuous monitoring as bad actors are able to modify or clear logs to evade detection.

Similar to system logs, organizations can deploy monitoring tools that capture detailed activity not significant enough for system logs. This can include process execution logs, application activity, file operations, network connections, and authentication events [8].

By integrating perimeter network data, internal network data, and distributed endpoint data, organizations can detect and defend against cyber threats by identifying anomalies and comparing them to established baseline activity.

## IV. Preprocessing and Data Cleansing Methods

Filtering data is an important step when it comes to preparing it for machine learning models. This varies on the data type and source, but has some standardized techniques. There are several aspects when it comes to processing data, such as looking for the seemingly desirable values and excluding ones that are insignificant. On top of this, depending on the data source, some logs of data may be incomplete. This means those instances must be discarded, or the missing values replaced in several different ways.

A. Direct Data Cleansing

There are several common data cleaning steps specifically for anomaly detection in cyber security. The first step directly after obtaining data from the sources include direct cleansing the data. Some of these methods include removing redundant logs, normalizing packet headers, and filtering benign traffic.

Removing redundant logs helps reduce the overall storage for the data, since most of the sources will have a multitude of duplicates. It also helps make sure certain patterns are not represented, which ensures real anomalies will not be obscured [10].

Filtering benign traffic also reduces overall storage and processing speed, as it eliminates trusted sources, regular system updates, and communication between internal servers [10]. How this is done is shown in the figure below.

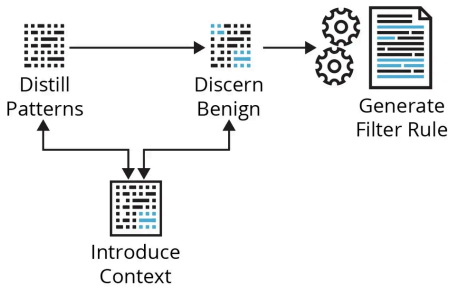


Fig 3. Filtering Benign Traffic [13]

The normalization of packet headers is different, as it helps standardize the many fields of network traffic data, which include IP addresses, port numbers, and timestamps. This helps the data all be in the same format, which means it can be compared much easier [10].

## B. Feature Extraction

Now that the data has been directly cut down and transformed into something that can be interpreted, the use of a feature extraction technique is the next step. Motif-based sequence representation is a key preprocessing method that transforms raw system call sequences (SCS) into structured patterns for anomaly detection. Each system call is first translated into a unique symbol and ranked based on how likely it is to be involved with malicious activity..

To extract meaningful patterns, motifs (frequently occurring subsequences) are identified using two techniques. One of these is auto-match, which detects repeated patterns within a single sequence, and another is cross-match, which finds common motifs across multiple sequences. These motifs are stored in a database and ranked based on their likelihood to be malicious [11]. The final step involves representing sequences in a two-dimensional space. Motifs are plotted based on their positions within the sequence. Transforming this data so it has more structure helps in detecting anomalies by highlighting variations in system behavior that might indicate cyber threats [11].

Apart from motif-based sequence representation, N-gram analysis can be used too. N-gram analysis is a powerful feature extraction method that breaks down sequences of network traffic, system logs, or command executions into contiguous patterns of N items [12]. Shown below is an example.

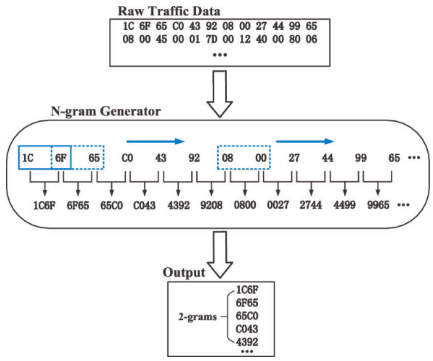


Fig 4. Turning network traffic into N-grams [14]

Each sequence is segmented into overlapping n-grams, capturing local dependencies and recurring patterns within the data. To identify the most relevant features for anomaly detection, n-grams are extracted from the dataset. A ranking process is applied, similar to motif-based sequence representation, where common n-grams in attack traffic but rare in normal traffic are prioritized as potential cyber threats. These selected n-grams are then encoded into feature vectors, enabling machine learning models to recognize patterns associated with anomalies. By structuring raw sequential data into measurable and comparable patterns, n-gram analysis enhances the ability to detect deviations from normal behavior, improving cybersecurity defenses [12].

## C. Final Data Preprocessing Methods

With all of this accomplished, there are several final data cleansing methods before it is finally ready to be incorporated into machine learning models. These depend on the different scenarios the data could be at, which includes high-dimensionality, redundant/irrelevant features, inconsistent scaling, and textual patterns.

Dimensionality reduction techniques like Principal Component Analysis (PCA) or autoencoders help remove redundant information while preserving key patterns, making anomaly detection more efficient.

Next, feature selection methods such as mutual information selection or the chi-square test can identify the most relevant features, ensuring that only the most important data points are used for classification.

To maintain consistency across features, scaling and normalization techniques like Min-Max scaling or Z-score standardization are applied, preventing models from being biased by varying feature magnitudes.

Finally, to enhance interpretability and optimize model performance, final transformations such as TF-IDF (Term Frequency-Inverse Document Frequency) for text-based data or Word2Vec embeddings for sequence-based data can be used to change raw features into structured numerical formats [10]. These steps all refine the data, ensuring it is well prepared for the next step of effective anomaly detection.

## V. Machine Learning Models for Anomaly Detection

There are a variety of machine learning models that can be used in anomaly detection, but each one falls under one of three broader categories. These categories include supervised, unsupervised, and reinforcement learning. By understanding these categories, it can help a developer narrow down what specific models they can use for their specific needs.

Firstly, there are supervised learning models, and this is where a machine learning algorithm learns patterns from labeled historical data. This allows the model to create a mapping equation based on inputs and outputs that are then used to make predictions of future data. Based on the definition, these algorithms are primarily used when the user knows what kind of output they should expect [15].

On the flip side, unsupervised learning models are training on information without labeled data. They instead group input data into classes that have similar features and use those classes to predict future outcomes. Since these models deal with unlabeled data, they are used to find hidden patterns that may not be able to be seen at first glance [15].

Lastly, reinforcement learning deals with an agent receiving feedback based on past outcomes and using that feedback to make better informed decisions in the future. The agent typically performs different actions within an environment and either receives rewards or penalties based on certain choices. The goal by the end of the training is to maximize the total profit [15].

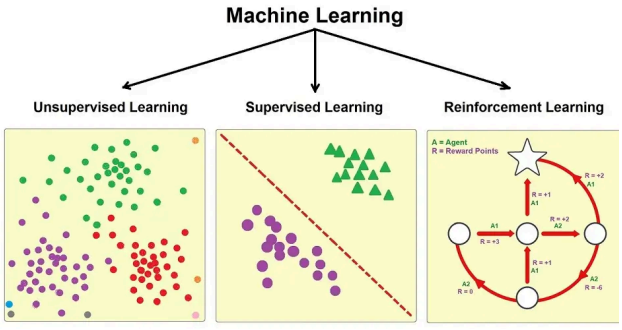


Fig. 5. Visual representation of three models [16].

### A. Long Short Term Memory Recurrent Neural Network (LSTM RNN)

A recurrent neural network (RNN) is a type of supervised deep learning model that processes past sequential data to understand and make decisions about current input. Think of it as reading a book, a person wants to remember the words before the current word to understand the meaning of that current word or possibly predict what it might be. RNN's have a kind of memory that allows them to remember previous data in the sequence, but the issue that they have is that they do not have the ability to remember that much, which is why the Long Short Term Memory Recurrent Neural Network was developed [17].

Like the name suggests, the LSTM has the ability to keep track of much more past data than a normal RNN, so it is built to avoid long term dependency problems. It does this by incorporating feedback connections that allow the model to process entire sequences of data rather than just individual data points. This functionality makes the LSTM effective in learning and predicting patterns in sequential data, such as time series.

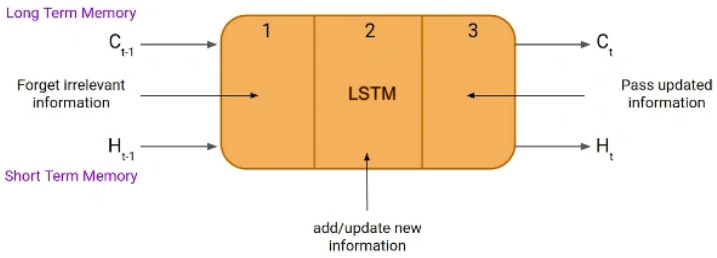


Fig 6. Logic behind LSTM [18].

The final topic to understand before looking at how the LSTM RNN can be used to detect anomalies is the logic of the LSTM. The model is made up of three distinct gates that all deal with the understanding and processing of information.

The first gate is the forget gate. Here, the model decides whether the previous sequential information is relevant, and if it is not, then it is completely forgotten. Information that was further back in the sequence can be an example of something that can be forgotten since the more recent information may be more important to making predictions.

Next is the input gate. The purpose of this gate is to quantify the importance of all new information. For instance, someone's name would be flagged as something to be remembered while filler words, like the, can be forgotten.

The third and final gate is the output gate. This is the stage that deals with predicting future events. The

gate takes into account all past and present information to create a possible prediction from what comes next.

One final thing to mention about all these gates is that they all follow some formula to quantify the relevance and possibility of an event. Each formula takes into account the input at the current timestamp, the weight matrix of the input, the hidden state at the previous timestamp, and the weight matrix associated with the hidden state. Each formula then has a sigmoid function applied to it to calculate some value between 0 and 1. This value is what tells the user how relevant past information is, how important the input is, and the token with the highest value being the prediction for the future [19].

In the paper, "Collective Anomaly Detection based on Long Short Term Memory Recurrent Neural Network" [5], the authors suggest a method to use the LSTM RNN to detect collective anomalies. By the definition of collective anomalies, they can not be noticed when individual points are considered, so using a time series model, such as the LSTM RNN, provides an effective way to consider data points together. In order to identify collector anomalies specifically, the authors introduce the following essential terms that can provide measurements to help the model decide if there is a collective anomaly or not [5].

**Relative Error (RE):** the Relative Error between two real values $x$ and $y$ is given by Eq. 1:

$$RE\ (x, y) = \frac{|x - y|}{x} \quad (1)$$

**Relative Error Threshold (RET):** Relative Error value above a predetermined threshold indicates an anomaly. This threshold, $RET$ , is determined by using labeled normal and attacks from a validation set.

**Minimum Attack Time (MAT):** The minimum amount of recent time steps that is used to define a collective attack.

**Danger Coefficient (DC):** The density of anomalous points within the last $MAT$ time steps. Let $N$ be the number of anomalous points over the last $MAT$ time steps, $DC$ is defined as in Eq. 2.

$$DC = \frac{N}{MAT} \quad (2)$$

NB: $0 < DC < 1$

**The Averaged Relative Error (ARE):** The Average Relative Error over a $MAT$ is given by Eq. 3:

$$ARE = \sum_{i=1}^{MAT} RE_i \quad (3)$$

Fig 7. Important definitions and formulas for determining collective anomalies [5].

It is important to note that the DC and ARE are the key values for determining if the latest time steps are identified as a collective anomaly. Also, the proposed method uses a validation set to compare the predicted value and real future value to compute RE values. Based on the observation of RE values, the developers can determine a proper RET.

After determining the RET, the next step is to adapt the LSTM RNN model to actually be able to detect collective anomalies rather than just point ones. The paper suggests using a circular array containing the MAT latest error values to represent possible anomalies in the latest time steps. The array element allows the model to store past time series data efficiently so that it can be used whenever the model needs to look to the past. When looking at the array to analyze possible anomalies, the model can find one if both the DC and ARE are greater than calculated thresholds alpha and beta respectively [5].

These thresholds come from a preliminary training and validation phase to tune the LSTM RNN. In the training phase, the model is only fed normal data so that it can learn what is considered valid input by the developers. In the validation phase, the model is fed labeled normal data as well as anomaly data. From this, the model can see how the anomaly differs from normal data, and the developers can determine proper alpha and beta detection thresholds. Once the model is properly tuned to detect collective anomalies, researchers can feed testing data into the model.

In the paper, the authors perform their own experiment to see how to model performs with a particular dataset. They found that the LSTM RNN with the use of a circular array accurately identified collective anomalies at different thresholds. However, the more accurate it got, the more false positives the model produced. This means individuals need to ensure that this model is trained properly before entering testing data. All in all though, the Long Short Term Memory Recurrent Neural Network is an excellent choice in identifying anomalies, but it is especially effective at finding collective ones since it focuses on time series data.

Since this supervised machine learning model focuses on time series data, it can be extremely useful in cybersecurity with things like network traffic, user activity, and system logs. All of these use cases contain some sort of time logs, so the LSTM RNN can effectively be trained on normal patterns and throw a flag whenever there is a significant deviation from the norm.

### B. K-Means Clustering Algorithm

K-means clustering is a frequently used unsupervised machine learning algorithm that partitions a dataset into a K number of clusters based on similar features. Each data point is assigned to the cluster whose centroid (the mean of the points) is nearest. Euclidean distance is typically used for this, as it is simply the distance between two points. The goal is to minimize the cluster sum of squares within, which groups similar data points together. Though conceptually simple and efficient to compute, the traditional K-means algorithm has limitations, particularly with how sensitive it is when first placing centroids, as well as its vulnerability to local minima.

In network anomaly detection, the need to differentiate between normal and potentially malicious traffic is critical. This algorithm has been increasingly used in this field due to its ability to uncover hidden patterns without requiring the data to be specifically labeled. However, to increase its adaptability and performance, researchers have developed hybrid systems that integrate K-means with other algorithms like Support Vector Machines (SVMs), Particle Swarm Optimization (PSO), and decision trees.

One example is where Lima et al. [21] proposed an anomaly detection system that combines K-means with PSO. This system uses baselines generated by the Baseline for Automatic Backbone Management (BLGBA) tool, which models normal network traffic patterns from previous data. The real-time traffic is compared against these baselines using K-means to generate clusters, and PSO is used to optimize cluster centroids. This overcomes K-means limitations in local optima avoidance. Traffic that significantly deviates from the centroid is flagged as anomalous. The structure of this system is illustrated in Figure 8, which shows how traffic data is collected, clustered using PSO-enhanced K-means, and then evaluated for anomalies.

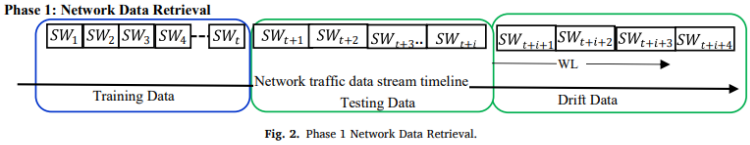


**Fig. 2.** Phase 1 Network Data Retrieval.

Fig 8. Collection and evaluation of data [22]

The system was tested in a real-world network setup at the State University of Londrina. It performed well, reaching a high detection rate of 82.92% while also keeping the false alarm rate low at just 2.85%. These results show that using K-means with PSO can be an effective method for detecting unusual network behavior [21].

In fast-changing network environments, it's common for traffic patterns to shift over time—a problem known as concept drift. To deal with this, Jain et al. [22] developed a hybrid model that uses K-means clustering along with a Support Vector Machine (SVM) to adjust to these changes in streaming data. Their method divides the data in several steps. First, the method divides the incoming network traffic into segments, preparing it for the algorithm. Once this is done, the actual K-means algorithm helps organize this data into groups. This reduces the amount of information the system needs to process. These groups are then used to train and update the SVM model. The system checks for drift using two methods. One of these is based on how the error rate changes, while another looks at shifts in the data itself. Figure 9 shows how this is organized, including how the system collects the data, then detects any changes, and trains the model again if needed.

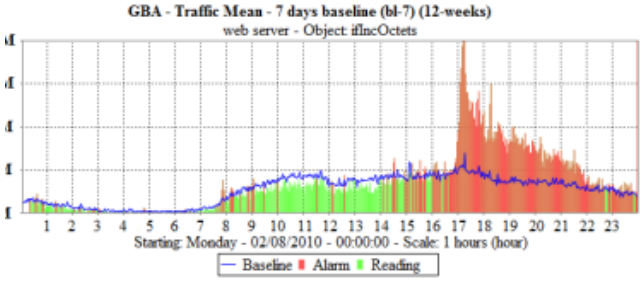


Fig 9. Organization and collection of data [21]

This helps the model stay accurate, even though network traffic can be very volatile. Tests using the datasets showed that the system reached over 91% accuracy and was able to detect concept drift effectively. K-means plays an important role in this setup by helping reduce the amount of data the system needs to process and by spotting changes in how the data is structured [22].

Another way to improve the performance of

K-means algorithms is by using decision tree algorithms along with them. Muniyandi et al. [20] introduced a method where K-means is first used to divide the data into the specific number of clusters, and then a separate decision tree is trained on each individual one. This two-step process helps fine-tune the decision-making inside each group and solves two major problems: the "forced assignment" issue (which happens when K is set too low) and the "class dominance" issue (when one label takes over a cluster and hides other patterns).

When tested on DARPA 1999 network traffic data, this combined method showed higher detection accuracy and fewer false alarms than using K-means or C4.5 on their own. By giving each part of the system a specific role—clustering or classifying—it becomes more flexible and able to handle different types of data more effectively [20].

Even with its limitations, K-means continues to be a key tool in detecting unusual network behavior. On its own, it can struggle with how clusters are shaped and where centroids start. But when it's paired with other techniques like PSO, SVM, or decision trees, its strengths become much more useful. As seen in all three studies, K-means works best as part of a larger system that combines different methods to better handle constantly changing network data. Whether it's detecting traffic spikes, adjusting to long-term changes, or improving decision rules, K-means is both reliable and flexible, keeping many networks secure.

### C. Isolation Forest

Isolation Forest (IF) is an unsupervised anomaly detection algorithm recently introduced in 2008. Unlike other algorithms that profile normal behavior, IF's core principle is, as the name suggests, isolating anomalies. Other methods model the distribution of normal data, IF uses random partitions to isolate outliers [23].

The algorithm constructs a large group of isolation trees (iTrees), which act essentially as random binary decision trees. Each iTree is recursively built via selecting a random feature and random split value until every point is isolated or maximum tree depth is reached. The length of the path from the root of the tree to the leaf nodes becomes a representation of the tree's normality, where shorter paths are indicative of potential anomalies [24].

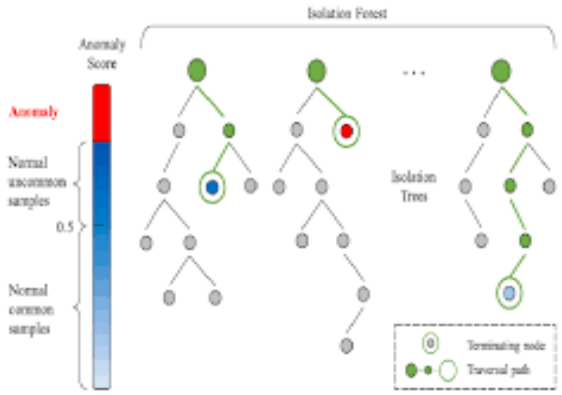


Fig 10. Diagram of iTree and Isolation Forest [25].

Cybersecurity analysts can then calculate the average path length over the "forest" of trees to find an anomaly score. This score is then standardized between 0 and 1. A tree that is significantly shorter than the rest and thus indicative of an anomaly has a score close to 1. A normal length tree has a score close to 0. From there, analysts can manually inspect and look for potential anomalies or threats [23].

Isolation forest has proven to be effective for a variety of cybersecurity anomaly detection tasks. Its ability to operate without predetermined "examples" and create its own baseline of normal is especially valuable, as this can oftentimes be difficult to determine.

IF is commonly used in network intrusion detection systems, where it can identify unusual network traffic patterns or connection attempts. For example, IF can flag bursts of malicious login attempts or suspicious authentication patterns indicating a possible brute-force attack or compromised account.

IF has also been used to detect unusual system behavior. For instance, rare sequences of system calls or unusual processes happening in the background that could signal possible malware. For network monitoring, it also detects abnormal traffic flows by pinpointing outlier flow records. This specific case highlights IF's strength of not needing predefined attack signatures like other algorithms. It is able to effectively detect subtle deviations. This also means it is well prepared to take on emerging threats.

The Isolation Forest algorithm offers several advantages for anomaly detection. As already stated, it is completely data driven and does not require any labeled training data or prior knowledge of attack

patterns. It can figure out what “normal” looks like by itself. This attribute is essential as attacks are constantly evolving and predetermined attack data to train on is scarce [24].

IF is also very efficient and scalable. Its tree-based isolation mechanism is able to run in linear time relative to the size of the given dataset and it handles high-dimensional feature spaces well. These attributes mean that IF is easily able to scale to large corporate datasets and effectively train and detect.

Another benefit is IF is able to drown out the noise quite effectively. Since trees are made up of many data points, a small number of noisy points will have minimal influence on the length of the tree, and thus the model's view of normal behavior. Only real anomalies are able to make a meaningful impact on the outcome of the tree. Going along with this, since IF works to isolate anomalies instead of modeling normality, it is able to recognize data points that are vastly different. This is important as new patterns in attacks emerge.

Despite its strengths, Isolation Forest does have some drawbacks. One consideration is the need to estimate the expected anomaly proportion when configuring the model. For example, if 1 is indicative of an anomaly, where do we draw the line? Is it .8 or higher, .95, etc. Deciding this cutoff can be challenging and requires having an idea of what percentage of the data is abnormal, because you can’t check every tree or piece of data. You don’t want to flag too much or too little data.

Another drawback is IF can sometimes struggle with subtle anomalies. If a piece of data is only a slight outlier or an entire tree is made up of only slight outliers, it may not have a large enough impact on the length from root to leaf resulting in no detection. This can create blindspots where data that is a threat or an anomaly gets the green flag and remains undetected [24].

To summarize, Isolation Forest is a powerful tool for anomaly detection in cybersecurity. Though it does have its drawbacks, with tuning and context understanding, IF serves as a strong method for identifying unknown threats in security data at scale.

## VI. Interpreting Results

When evaluating the performance of a machine learning algorithm, it is essential to not just understand that it works, but to understand how well it works. In order to do this, developers can use different classification metrics to determine how often the model is correct, how wrong it is, and how severe mistakes may be.

The confusion matrix is the first step in the process of understanding results. The matrix summarizes the performance of an algorithm by separating each recorded value into one of the following categories: true positive, true negative, false positive, and false negative. From here, four important values can be calculated to describe the performance.

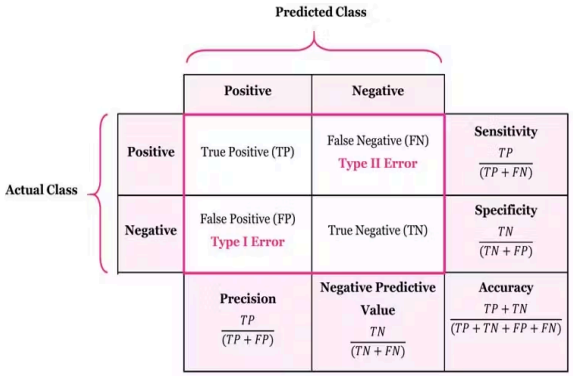


Fig 11. Confusion matrix visualization [27].

The first, and most simple of the four metrics is accuracy. This simply describes how correct the model is overall. It can be calculated by dividing all accurate predictions by every single prediction. While accuracy is very useful in identifying correctness, it can be extremity misleading, especially in anomaly detection where anomalies are rare. For example, if anomalies make up about 1% of traffic, a model can look 99% accurate if it always predicts normal, but it is actually very useless information since the anomalies only make up 1% of the dataset. Due to this, the following metrics can be more indicative of the correctness of models in anomaly detection [26].

Precision describes the number of predictions that are actually positive. It is found by dividing true positives by all predictions that were flagged as positive. In other wards, it can tell users how many false alarms were raised. The higher the precision score, the fewer false positives. Precision should be used when false positives cause disruption. An example of this in anomaly detection would be flagging legitimate logins. If the model is not precise, it may flag legitimate users and lock them out of their system [26].

Recall, also known as true positive rate, can be commonly confused with precision, but it deals with identifying correctly predicted positives against all use cases that were actually positive. This is done by dividing the true positives by every use case that is actually positive. False negatives can have serious consequences if they go undetected, so it can be very crucial to make sure the recall score is high. Developers would want to use recall scores if it is critical to not miss any threats, such as an intruder breaking into a system. If the intruder is actually identified as a false negative, then they could have access to protected information [26].

The final metric is F1 score, and it measures the combination of precision and recall. It is calculated by dividing precision multiplied by recall twice by precision plus recall. This metric should be used when false positives and false negatives are equally as bad. An example would be examining system logs to detect possible strange behavior by employees. Too many false positives will result in overwhelming the security team by having to check many legitimate users, but too many false negatives may result in missing actual threats.The goal is for the model to be reliable without causing too many false alarms, and that is what the F1 score can convey [28].

With the use of all of these metrics, developers can efficiently test the correctness of their model and see where there are potential shortcomings.

## VII. Alternative Algorithms and Emerging Approaches

The methods mentioned earlier are widely used in cybersecurity for spotting unusual activity. But as needs change and data becomes more complex, new and improved techniques have started to show up. These try to boost accuracy, make results easier to understand, and speed things up.

LSTM networks have been a go-to for working with time-based data since they're good at spotting long-term patterns. Still, they can be slow and hard to train. Transformers offer a newer option. Unlike LSTMs, they use self-attention to look at all parts of a sequence at once instead of one step at a time. Models like the Temporal Fusion Transformer (TFT) and Time Series Transformer (TST) have done well in finding anomalies by picking up both quick changes and long-term trends more effectively [29].

K-means clustering is simple and works well with large datasets, but it struggles when clusters aren't clearly separated or when there's noise. Newer methods like DBSCAN and HDBSCAN are better at this [29]. They can find clusters of different shapes and spot outliers more consistently, making them useful when you do not have data without labels.

Isolation Forests are also great for data with multiple dimensions. They work by splitting the data again and again to find what doesn't fit. But they can have trouble with very complex patterns. Newer versions like the Deep Isolation Forest use deep learning to handle more complicated data [29]. Autoencoders and VAEs are also being used more frequently, as they learn what normal data looks like and flag anything that doesn't fit as an anomaly.

So while LSTM, K-means, and Isolation Forests are still important, newer tools like transformers, density-based clustering, and deep learning models offer more powerful and flexible options. Going forward, the best results may come from combining these methods to handle the messy and fast-changing data found in real-world systems.

## VIII. Conclusion

Anomaly detection plays a critical role in modern cybersecurity by providing the necessary tools and logic to identify malicious behavior. Through the use of machine learning algorithms, such as long short term recurrent neural networks, k-means clustering, and isolation forests, organizations can not only detect the obvious threats, but they can detect very subtle, unseen patterns. Each model has their own strength and weakness, such it is up to the user to identify their needs and train the model appropriately.

This paper explored types of anomalies, data sources from which they originate, preprocessing techniques required to transform data. It also went into depth about various machine learning algorithms and discussed important metrics, such as accuracy, precision, recall, and F1 score, that can be used to evaluate the performance of each algorithm.

As cybersecurity threats continue to become more complex, so do the tools needed to identify them. Moving forward, research should continue to focus on developing adaptive, accurate algorithms that combine traditional models with new innovations. This way, cybersecurity analysts can build systems that are accurate, efficient, and resilient to any kind of threat.

**References**


[1] CrowdStrike, "Anomaly Detection: How Next-Gen SIEM Improves Threat Detection," *CrowdStrike*, [Online]. Available: https://www.crowdstrike.com/en-us/cybersecurity-101/next-gen-siem/anomaly-detection/. [Accessed: 01-Mar-2025].

[2] Anomaly, "What is a Data Anomaly? Common Types and How to Identify Them," *Anomalo Blog*, [Online]. Available: https://www.anomalo.com/blog/data-anomaly-what-is-it-common-types-and-how-to-identify-them/. [Accessed: 01-Mar-2025].

[3] ServerFault, "Trying to Identify Bandwidth Usage Spike on Linux Debian VM," *ServerFault*, [Online]. Available: https://serverfault.com/questions/1009867/trying-to-identify-bandwidth-usage-spike-on-linux-debian-vm. [Accessed: 02-Mar-2025].

[4] M. J. Zaki, "Data mining and machine learning in big data: A survey," *Journal of Big Data*, vol. 1, no. 1, p. 1, 2014. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-014-0011-y. [Accessed: 02-Mar-2025].

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *arXiv preprint arXiv:1703.09752*, 2017. [Online]. Available: https://arxiv.org/pdf/1703.09752. [Accessed: 02-Mar-2025].

[6] Flō Networks, "Network perimeter security: The first line of defense against cyber threats," [Online]. Available: https://flo.net/network-perimeter-security-the-first-line-of-defense-against-cyber-threats/. [Accessed: Mar. 3, 2025].

[7] Securiwiser, "What are the top sources for collecting network data?," [Online]. Available: https://www.securiwiser.com/blog/what-are-the-top-sources-for-collecting-network-data/. [Accessed: Mar. 3, 2025].

[8] C. A. Alexander and L. Wang, "Cybersecurity data sources and practices," *Journal of Computer Networks*, vol. 12, no. 1, pp. 1–6, 2024. [Online]. Available: https://pubs.sciepub.com/jcn/12/1/1/. [Accessed: Mar. 3, 2025].

[9]Devo, "Understanding Security Analytics," *Devo Threat Hunting Guide*, [Online]. Available: https://www.devo.com/threat-hunting-guide/understanding-security-analytics/

[10] Encord, "Data Cleaning vs. Data Preprocessing: Key Differences & Best Practices," *Encord Blog*, [Online]. Available: https://encord.com/blog/data-cleaning-data-preprocessing/. [Accessed: 21-Feb-2025].

[11] P. K. Chan and S. J. Stolfo, "Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection," *Florida Institute of Technology*, [Online]. Available: https://cs.fit.edu/~pkc/papers/mldmcs.pdf. [Accessed: 24-Feb-2025].

[12] Security Boulevard, "Fast, Effective N-Grams Extraction and Analysis with SQL," *Security Boulevard*, May 2021. [Online]. Available: https://securityboulevard.com/2021/05/fast-effective-n-grams-extraction-and-analysis-with-sql/. [Accessed: 01-Mar-2025].

[13] R. E. Fredrikson, "Dealing with Noisy Behavioral Analytics in Detection Engineering," *SEI Blog*, 2023. [Online]. Available: https://insights.sei.cmu.edu/blog/dealing-with-noisy-behavioral-analytics-in-detection-engineering/

[14] M. Homayoun and A. Namin, "Anomaly detection in cybersecurity using deep learning," in *Advances in Data Science and Intelligent Data Communication Technologies for COVID-19*, Springer, 2021, pp. 289–303. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-78612-0_2416

[15] Educative.io, "Supervised vs. Unsupervised vs. Reinforcement Learning," *Educative*, [Online]. Available:

https://www.educative.io/answers/supervised-vs-unsupervised-vs-reinforcement-learning

[16] Ejable, "Types of Machine Learning," *Ejable Tech Corner*, [Online]. Available: https://www.ejable.com/tech-corner/ai-machine-learning-and-deep-learning/types-of-machine-learning/

[17] Amazon Web Services, "What is a Recurrent Neural Network (RNN)?" *AWS Documentation*, [Online]. Available: https://aws.amazon.com/what-is/recurrent-neural-network/

[18] Analytics Vidhya, "LSTM Network Logic Diagram," *Analytics Vidhya*, [Online]. Available: https://cdn.analyticsvidhya.com/wp-content/uploads/2024/08/Screenshot-from-2021-03-16-13-45-35.webp

[19] R. Ghosh, "Introduction to Long Short Term Memory (LSTM)," *Analytics Vidhya*, Mar. 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

[20] K. C. Muniyandi, R. Rajeswari, and R. Rajaram, "Network anomaly detection by cascading K-Means clustering and C4.5 decision tree algorithm," *Procedia Engineering*, vol. 30, pp. 174–182, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705812008594

[21] L. C. Jain and H. S. Behera, "Hybrid model using K-means and support vector machine for network intrusion detection," in *2010 International Conference on Data Storage and Data Engineering*, IEEE, 2010, pp. 142–146. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5623690

[22] A. Lima, C. Carvalho, and J. C. Silva, "Anomaly detection system based on K-means and particle swarm optimization for network traffic," *Expert Systems with Applications*, vol. 194, p. 117116, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422000112

[23]W. Chua, A. L. D. Pajas, C. S. Castro, S. P. Panganiban, A. J. Pasuquin, M. J. Purganan, R. Malupeng, D. J. Pingad, J. P. Orolfo, H. H. Lua, and L. C. Velasco, "Web traffic anomaly detection using isolation forest," *Informatics*, vol. 11, no. 4, p. 83, 2024. [Online]. Available: https://doi.org/10.3390/informatics11040083 [Accessed: Mar. 3, 2025].

[24]J. Devry, "THREAT HUNTING THROUGH THE USE OF AN ISOLATION FOREST," *Cybersecurity Insiders*, Dec. 01, 2017. [Online]. Available: https://www.cybersecurity-insiders.com/threat-hunting-through-the-use-of-an-isolation-forest/ [Accessed: Mar. 3, 2025].

[25] "Anomaly Detection using Isolation Forest," *ResearchGate*, [Online]. Available: https://www.researchgate.net/figure/Anomaly-Detection-using-Isolation-Forest-18_fig3_350551253. [Accessed: Mar. 3, 2025].

[26] Google Developers, "Accuracy, Precision and Recall," *Google Machine Learning Crash Course*, [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

[27] Encord, "Confusion Matrix," *Encord Glossary*, [Online]. Available: https://encord.com/glossary/confusion-matrix/

[28] GeeksforGeeks, "F1 Score in Machine Learning," *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/f1-score-in-machine-learning/

[29] V. Sridevi and A. Amudhavel, "An Empirical Study on Anomaly Detection Using Density-based and Representative-based Clustering Algorithms," *ResearchGate*, [Online]. Available: https://www.researchgate.net/publication/370125117_An_Empirical_Study_on_Anomaly_Detection_Using_Density-based_and_Representative-based_Clustering_Algorithms