CYBR 520

Module 5: Anomaly Detection

Chapter 3: Anomaly Detection(Network Intrusion Detection)***

PLEASE REFER TO MODULE 5 ON GITHUB TO OBTAIN THE CODES OR REFER TO THE BOOK'S GITHUB REPO

Topics

1. Introduction to Anomaly Detection
2. Anomaly Detection Vs. Supervised ML
3. Intrusion Detection with Heuristics
4. Data-Driven Methods
5. Feature Engineering for Anomaly Detection
6. Anomaly Detection with Data and Algorithms
7. Challenges using ML in Anomaly Detection

What is Anomaly Detection?

- Anomaly (also known as outlier) Identifying unusual patterns that do not conform to expected behavior, known as outliers.
- Crucial for fraud detection, network security, fault detection, system health monitoring, and other areas requiring critical event recognition.

Anomalies and Outliers

- In this chapter, we use "outlier" and "anomaly" interchangeably.
- There's a distinction between the two however:
  - Outlier detection (learning from data with both regular and outlier data)
  - Novelty detection (learning from data without outliers).
- Both are forms of anomaly detection.

Anomalies and Outliers

- Regular data points are non-anomalous.
- "Normal" in this context refers to a statistical normal (Gaussian) distribution,
- " Standard" refers to a normal distribution with mean zero and unit variance.

Approaches for Anomaly Detection

- **Supervised:** Labelled data used to identify anomalies.
- **Unsupervised:** Algorithms find anomalies in unlabeled data by looking for data points that are significantly different from the majority of the data.
- **Semi-supervised:** Models are built with a small amount of labeled anomaly samples and a larger amount of normal data.

Anomaly Detection Techniques

- **Statistical Methods:** Assumption-based (e.g., Gaussian distribution), where data points in low probability regions are considered anomalies.

- **Machine Learning Methods:**
  - **Clustering:** Algorithms like K-Means detect outliers as points far from the centroid of clusters.
  - **Classification:** Support Vector Machine (SVM) for high-dimensional spaces.
  - **Neural Networks:** Autoencoders that can reconstruct normal data but fail to do so for anomalies.

Anomaly Detection: Challenges

- High false-positive rate.
- Difficulty in obtaining labeled data for training.
- Adaptive anomalies over time (concept drift).

Anomaly Detection in security

- This module is about detecting unexpected events, or *anomalies*, in systems.
- In the context of network and host security, *anomaly detection* refers to identifying unexpected intruders or breaches
- Anomaly detection is essential for finding unusual events, like security breaches, system failures, or financial fraud.

Anomaly Detection in security

- After an attacker gains entry, however, the damage is usually done in a few days or less.
- Whether the nature of the attack is data exfiltration, extortion through ransomware, adware, or advanced persistent threats (APTs), it is clear that time is not on the defender's side.

Anomaly Detection Beyond Security

- Anomaly detection is not limited to security and has broader applications:
  - Identify early signs of system failure for critical systems
  - Power company detecting anomalies in the electrical grid.
  - Prevent expensive damage and outages.

Novelty Detection Vs. Outlier detection

- Time series is a sequence of data points observed at successive time intervals.
- It's essential in anomaly detection because anomalies are deviations from what's expected based on past observations.
- Anomalies are often defined as deviations from past observations, making time series analysis integral to anomaly detection.
  - We'll explore how to identify anomalies in data streams.

When to Use Anomaly Detection Versus Supervised Learning

- Anomaly detection and supervised learning are sometimes confused, but they serve different purposes.

Supervised Learning for Fraud Detection

- In cases like identifying fraudulent credit card transactions, supervised learning can be effective.
- It works well when you have a substantial dataset of both legitimate and fraudulent transactions for training, especially if you expect future fraud instances to resemble past ones.

Pattern Recognition in Credit Card Fraud

- Credit card companies often seek specific patterns that are more likely in fraudulent transactions, like large purchases after small ones, unusual locations, or purchases inconsistent with the customer's spending profile.
- These patterns can be learned using supervised learning from positive and negative examples.

Challenges in Server Breach Detection

- In scenarios like server breaches caused by zero-day attacks or new software vulnerabilities, it's challenging to predict intrusion methods in advance.
- The rarity of such events also leads to class imbalance, making supervised learning less suitable.

Anomaly Detection's Role in Rare Events

- Anomaly detection excels in scenarios where building a representative pool of positive examples is difficult, such as server breaches caused by unforeseen methods.
- It is well-suited to handling rare events and class imbalance.

Intrusion Detection with Heuristics

- Intrusion Detection Systems (IDSs) have been in use since 1986, relying on thresholds, heuristics, and simple statistical profiles to detect intrusions and anomalies.
- Using a threshold of 10 queries per hour as the upper limit for a database.
  - Anomaly detection is triggered when a user exceeds this threshold.

Challenges with Threshold-Based Logic

- Questions arise regarding threshold-based logic:
  - How to set the threshold?
  - Should different users have different thresholds?
  - Are there legitimate reasons for users to exceed the threshold?
  - How often should thresholds be updated?
  - Can attackers exfiltrate data by compromising multiple user accounts?
- Transition to using machine learning for solutions.

Dynamic Thresholds

- Replacing static thresholds with dynamically generated ones based on data.
- Example: Compute a moving average of queries per user per day and set the hourly threshold as a multiple of the daily average (e.g., 5/24).

Role-Based Thresholds

- Classifying users by roles and assigning different query thresholds for each role.
- Data analysts may have higher thresholds than receptionists.

Statistical Properties for Thresholds

- Using statistical properties (e.g., median, interquartile ranges) for setting thresholds.
- Improved resistance to outliers and tampering attempts.

Adaptive Thresholds and Machine Learning

- Introduction to adaptive-threshold solutions.
- Query thresholds resembling model parameters learned from regular events.
- Continuous training for adapting to changing user requirements.

Complex Systems and Threshold Challenges

- Challenges in more complex systems.
- Increasing number of thresholds.
- Scenarios where multiple thresholds interact.
- The need for probabilistic approaches to estimate anomaly likelihood.

Probabilistic Anomaly Detection

- Transition to probabilistic anomaly detection.
- Assigning risk scores based on likelihood.
- Estimating the probability of an event being anomalous.

Data-Driven Methods

- A solution based on data rather than thresholds or other crafted measures
- Before exploring solutions, it's crucial to define objectives for an ideal anomaly detection system.
- These objectives guide the development and implementation of effective anomaly detection solutions:
  1. Low false positives and false negatives
  2. Easy to configure, tune, and maintain
  3. Adapts to changing trends in the data
  4. Works well across datasets of different nature
  5. Resource-efficient and suitable for real-time application
  6. Explainable alerts

*(refer to the Anomaly Detection Objectives Jupyter Notebook)

Low false positives and false negatives

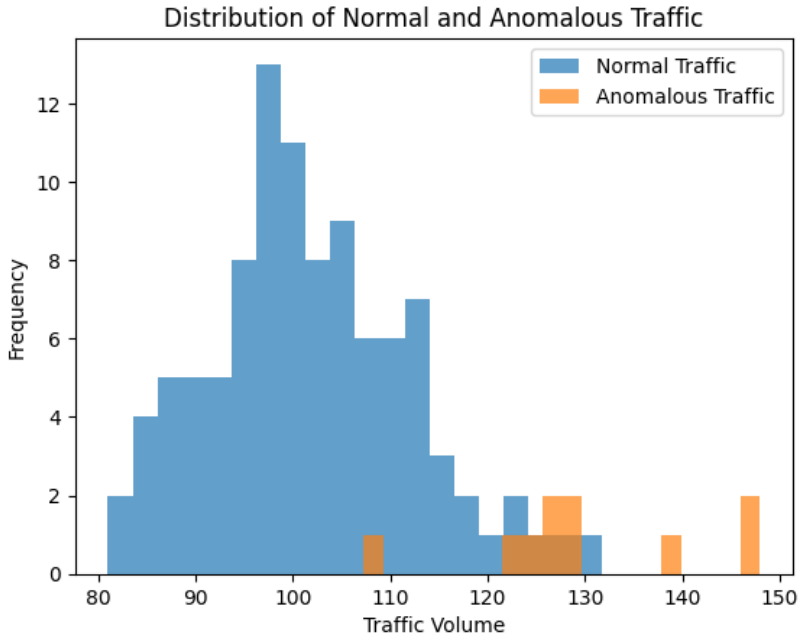
- False negatives miss actual anomalies:
  - The system does not find something that users intend to find
  - Lock working 9 out of 10 times, good?
- False positives flag normal events as anomalies.
  - Lock refuses to let a key holder in
- Minimizing both false positives and false negatives is a key objective.

Low false positives and false negatives

- In a “Network Traffic Anomalies” dataset, low false positives mean not flagging normal network traffic as anomalies.
- A false positive might occur if a legitimate user's routine traffic is flagged as suspicious.
- A false negative, on the other hand, means failing to detect a real anomaly.
- For instance, if a cyberattack goes unnoticed because it appears similar to normal traffic.

Low false positives and false negatives

- In this example, we generate histograms of network traffic volume.
- The objective here is to minimize the overlap between normal and anomalous traffic, reducing both false positives (normal data flagged as anomalies) and false negatives (anomalies missed).

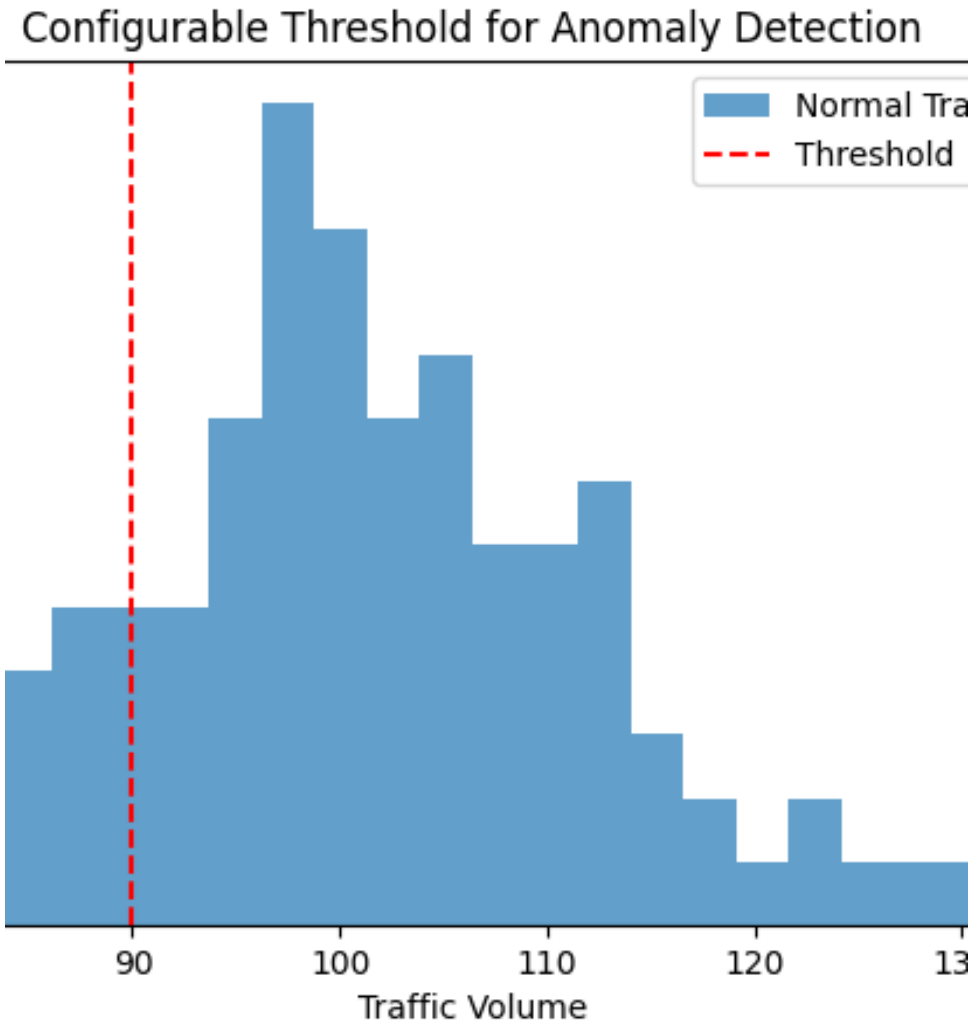


Easy to configure, tune, and maintain

- Easy configuration means that network administrators can easily set parameters like traffic thresholds or pattern recognition rules.
- This simplicity ensures that the system functions effectively.
- Maintenance involves adjusting settings as network behavior changes over time, such as adding new services or users.

Easy to configure, tune, and maintain

- This example allows users to interactively set a traffic threshold, making it easy for non-experts to configure the system.
- The threshold is visualized on a histogram to demonstrate its impact.

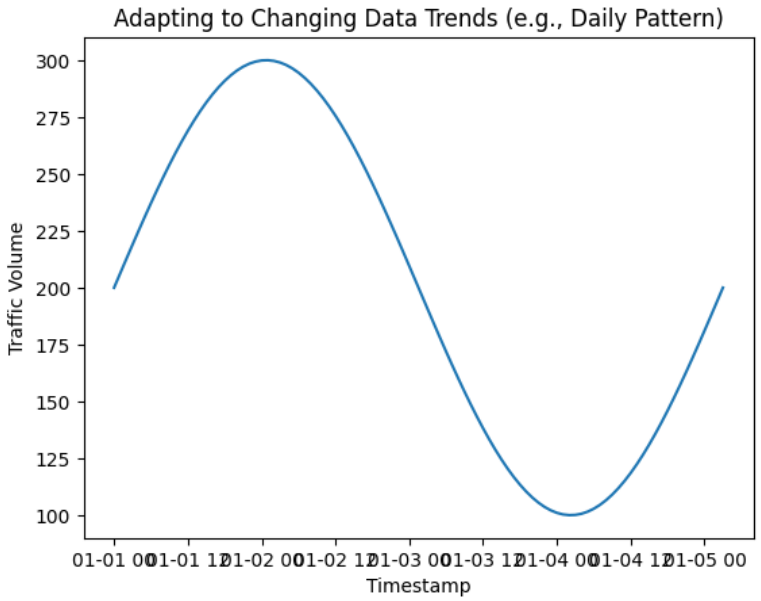


Adapts to changing trends in the data

- Seasonality, or regular patterns in data, can affect anomaly detection.
- An ideal system should be capable of identifying and adapting to changing data trends, reducing false positives.

Adapts to changing trends in the data

- In this example, we create a dataset with a daily traffic pattern.
- An effective anomaly detection system should adapt to such trends, reducing false positives during expected traffic changes.

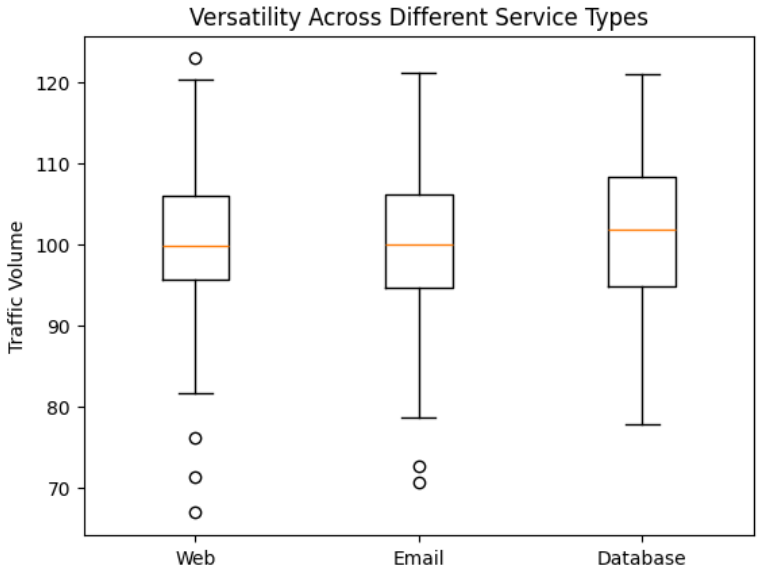


Versatility Across Different Datasets

- Not all datasets follow a Gaussian distribution, and assumptions about data distribution can limit effectiveness.
- The system should work well across diverse datasets with varying properties and distributions.

The system should work effectively with diverse datasets.

- This example generates box plots to demonstrate that the system can effectively handle different types of data, like web, email, and database traffic.



Resource Efficiency and Real-Time Application

- Anomaly detection, especially in security, is often time-sensitive.
- Efficient and real-time capable systems are essential to quickly identify potential threats or issues.

Explainable Alerts

- Auditing anomaly alerts is critical for system evaluation and investigation.
- Explainability is challenging, especially for machine learning-based systems.
- Clear explanations make debugging and decision-making more straightforward.

Feature Engineering

- Feature engineering refers to the methods of extracting meaningful representations from the raw data.
- Needled to say, this “better” your features are, the “better” your models are.
- For example, from somebody DOB, you can extract Age.

Feature Engineering for Anomaly Detection

- Many online (streaming) anomaly detection algorithms require input in the form of a time series data stream.
  - If your data (or the tools used to extract data) has time series, then you may not even need feature engineering.
  - For example, to detect whether CPU's utilization is abnormal, then you need to create your own features.

Feature Engineering for Anomaly Detection

- In this chapter, we focus our feature engineering discussions on three domains:
  1. *Host intrusion detection*
  2. *Network intrusion detection*
  3. *Web application intrusion detection*.
- We take a look at examples of tools that you can use to extract these features, and evaluate the pros and cons of the different methods of feature extraction.

Note

- Of course, anomaly detection is not restricted to hosts and networks only.
- Other use cases such as fraud detection and detecting anomalies in public API calls also rely on good feature extraction to achieve a reliable data source on which to apply algorithms.
- After we have discussed the principles of extracting useful features and time series data from the host and network domains, it will be your job to apply these principles to your specific application domain.

Host Intrusion Detection

- Host Intrusion Detection refers to the process and mechanisms used to detect unauthorized access or anomalous behavior on a host system, such as a workstation, server, or other networked device.
- This type of intrusion detection focuses on monitoring and analyzing the internals of a computing system as well as the network packets on its network interfaces.

Host Intrusion Detection

- Think, if you want to detect any intrusions, what would you do?
- You would “watch” your devices and the network during “normal” times, to establish a baseline.
  - Maybe you can then assume anything that does not fit with in the baseline is an intrusion attempt.
  - What do we need to watch for then?

Host Intrusion Detection

- Developing an intrusion detection agent for hosts involves generating our own metrics and correlating signals from various sources.
- The choice of metrics and tools depends on the threat model however, system-level and network-level statistics serving as a good starting point.

osquery

- qsquery is an OS instrumentation framework for collecting low-level OS metrics.
- qsquery is a powerful tool in the realm of cybersecurity, especially for intrusion detection on hosts.
  - It acts as a bridge between your computer's operating system and your queries, allowing you to explore and monitor the internals of your system like never before.

osquery

- Osquery specializes in collecting low-level operating system metrics.
  - These are like the fine-grained details of what's happening inside your computer's OS.
- Example Metrics osquery Can Provide:
  - **Running Processes**
  - **User Accounts**
  - **Kernel Modules Loaded**
  - **DNS Lookups**
  - **Network Connections:**
  - **….. and more**
- **See example on how to use this on GitHub**

`osquery Example`

Alternatives to osquery

- There are many open source and commercial alternatives to osquery that can help you to achieve the same end result: continuous and detailed introspection of your hosts.
- Mining the wealth of information that many Unix-based systems provide natively (`e.g., in /proc`) is a lightweight solution that might be sufficient for your use case.
- The Linux Auditing `System (auditd, etc.)` is much more mature than osquery and is a tool that forensics experts and operational gurus have sworn by for decades.

Network Intrusion

- Network Intrusion Detection (NID) is a crucial cybersecurity technology.
- It monitors network traffic for suspicious or malicious activity.
- Most host intrusions involve communication with the outside world.
- Detecting intrusions by focusing on network traffic.
- Objective: Prevent data theft and maintain network security.

Network Intrusion Techniques

- Botnets :remote command-and-control servers.
- APTs : hackers gaining remote access though misconfiguration.
- Adware requiring external server communication.
- Spyware transmitting covertly monitored data

Network Intrusion Detection (NID)

- NID helps identify and respond to cyber threats in real-time.
  - NID tools operate on the basic concept of inspecting traffic that passes between hosts.
- Focuses on collecting and analyzing network traffic of all sorts.
- Examples:
  - Tcpdump
  - Bro

Network Intrusion Detection (NID)

- NID can be categorized into two primary detection methods:

1. Matching traffic to known malicious signatures.
2. Anomaly detection by comparing traffic to baselines.

Network Intrusion Detection: Snort

- Snort is an open-source IDS that sniffs packets and traffic for real-time anomaly.
- **Sniffs Packets and Network Traffic**
- **Real-time Anomaly Detection**
- **De Facto Choice for Intrusion-Detection Monitoring**
- **….**

Feature Extraction for NIDs

- In extracting features for network intrusion detection, there is a noteworthy difference between :
  - **Extracting network traffic metadata**
    - Using Stateful Packet Inspection (SPI)
  - **Inspecting network traffic content**.
    - Using Deep Packet Inspection (DPI)

Extracting network traffic metadata

- Used in *stateful packet inspection* (SPI), working at the network and transport layers—OSI layers 3 and 4—and examining each network packet's header and footer without touching the packet context.
- SPI cannot detect signs of breaches or intrusions on the application level, because doing so would require a deeper level of inspection.

Inspecting network traffic content

- DPI examines data within network packets, not just headers.
- Collects signals and stats from the application layer.
- Detects spam, malware, intrusions, and anomalies.
- Computationally intensive for real-time streaming DPI.

Bro - Passive Network Monitoring

- Bro is an example of DPI
- Early system for passive network monitoring.
  - Provided with an event engine and policy engine.
  - Extracts signals from live traffic and enforces policies.
  - Bro can detect suspicious activity in web applications.
  - Example: Detecting SQL injections and XSS attacks.
  - Create profiles of POST body content.
  - Generate suspicion scores based on anomalous characters.

Features for network intrusion detection

- The Knowledge Discovery and Data Mining Special Interest Group (SIGKDD) from the Association of Computing Machinery (ACM) holds the *KDD Cup* every year, posing a different challenge to participants.
- Dataset is old but accessible here (https://kdd.org/kdd-cup/view/kdd-cup-1999/Data)

Anomaly Detection with Data and Algorithms

- After feature engineering from a raw event stream, it's time to use algorithms for anomaly detection.
- Anomaly detection is diverse, and there's no one-size-fits-all algorithm for all time series.
- Finding the best algorithm is an exploration and experimentation journe

Data Nature and Quality

- Consider the nature and quality of the data source before selecting an algorithm.
- Data pollution by anomalies impacts detection methodology.
- Task categorization: Novelty detection (no anomalies) vs. Outlier detection (anomalies present).

- Selecting the appropriate approach is crucial.
- Outlier detection requires algorithms insensitive to small deviations.
- Cleaning datasets to remove anomalies can be challenging.

Categorization of Anomaly Detection Methods

- Synthesizing anomaly detection methods into categories.
- Categories based on fundamental principles, each containing specific algorithms.
  1. Forecasting (supervised machine learning)
  2. Statistical metrics
  3. Goodness-of-fit tests (not included in our methodology)
  4. Unsupervised machine learning (we are mostly interested in this)
  5. Density-based methods.

Categorization of Anomaly Detection Methods

- Discussing strengths and weaknesses of each approach.
- Considerations for selecting an approach based on dataset characteristics.
- Example: Forecasting suitable for one-dimensional time series data.

Forecasting (Supervised Machine Learning)

- forecasting, a type of supervised machine learning, for anomaly detection.
- It draws an analogy with weather forecasting, where deviations from predicted weather patterns are considered anomalous.

Forecasting (Supervised Machine Learning)

- Forecasting algorithms use past data to predict current data and identify anomalies based on deviations from these predictions.
- The example of predicting rain in the absence of recent rain is used to illustrate the concept.
- Forecasting algorithms are suitable for single-dimensional time series datasets.
  - Time series data is represented in line charts, making it suitable for human interpretation, but challenging for machines due to noise and complex patterns.

Challenges in time series data

- Challenges in time series data, include noise caused by various factors.
- Using simple linear-fit methods to detect anomalies in such data may not yield accurate results.

Statistical Metrics

- Statistical tests help assess whether a new data point resembles previously observed data.
- For example, we can use moving averages as an adaptive metric to identify anomalies by detecting significant deviations from the average.
  - **Median Absolute Deviation (MAD)**
  - **Grubbs' Outlier Test**

Median Absolute Deviation (MAD)

- MAD, a robust alternative to standard deviation, identifies outliers in one-dimensional data. It calculates the median of absolute deviations from the series median, making it resistant to the influence of outliers.

Grubbs' Outlier Test

- Grubbs' test detects a single outlier in normally distributed datasets. It iteratively identifies outliers based on the current minimum or maximum value.
- This method is suitable for normal distributions but may be inefficient as it detects one anomaly per iteration.

Statistical Metrics: summary

- Statistical metric comparison, while simple, offers effective anomaly detection.
- It aligns with key features of optimal detectors, including reproducible alerts, adaptability to changing data trends, performance efficiency, ease of tuning, and maintenance.
- Though it has limited learning capabilities compared to machine learning algorithms, statistical metrics can excel in specific scenarios.

Unsupervised Machine Learning Algorithms

- We are mainly interested in this when it comes to anomaly detection.
- Supervised ML algorithm (classifiers) are used to solver problems with two or more classes.
- We can still use these algorithms in an unsupervised fashion:
  1. Once-class Support Vector Machine (SVM)
  2. Isolation forests
  3. Density-Based Methods
  4. Local outlier factor

  (refer to the code on GitHub for different Unsupervised ML implementations)

Challenges of Using Machine Learning in Anomaly Detection

- Machine learning has excelled in recommendation systems, but it faces unique challenges in anomaly detection.
- In recommendation systems, wrong suggestions have minor consequences. In anomaly detection, errors can lead to significant damage, like system breaches. Misclassifying even one anomaly can be disastrous.

Human Involvement

- Due to the high cost of errors, fully automated anomaly detection systems are rare.
  - Humans are usually involved to verify alerts before taking action.