Exploring statistical techniques for threat detection and AI security

Raffaele Castagna

Statistics Academic Year 2025-2026

Contents

1 Introduction **2**
1.1 The Dual Role of Artificial Intelligence: Tool and Target 2

2 Statistical Anomaly Detection in Network Security **3**
2.1 Gaussian Assumption and Univariate Analysis 3
2.2 Multivariate Analysis and Mahalanobis Distance 3
2.3 Payload Detection 4
2.4 Dimensionality Reduction and PCA 4
2.5 Industrial Control Systems and Operational Technology applications 5

3 Time Series Analysis for Network Forecasting **6**
3.1 ARIMA Models 6
3.2 Seasonal ARIMA (SARIMA) and ISP variability 7

4 Bayesian Statistics **7**
4.1 Bayesian Networks and Attack Graphs 7
4.2 K2 and Junction Trees 8
4.3 Parameter Learning and Dirichlet Distributions 8
4.4 Naive Bayes Classifiers 9

5 Detecting Adversarial Attacks **9**
5.1 Kernel Density Estimation 9

1 Introduction

Cybersecurity has always been shaped by data analysis, ever since its inception. Even at the start when the *fortress model* was the optimal choice, sysadmins typically relied on the binary classification of traffic, files and behaviours that were either safe or were known malicious patterns, but to actually analyze a file most relied on static analysis of signatures, where the hash of a file was computed and then denoted as malicious, this could also be applied to byte sequences in network traffic or to generalize even more IP ranges could be blocked, either based on suspicious activity or on countries. Under the hood there was a statistical assumption: the distribution of malicious behaviour was disjointed from the distribution of normal behaviour, and therefore there was a boundary between the two that was immutable.

However with the continued expansion of the internet, and the users therein, this assumption became obsolete, the world has changed and with it the amount of data that is ingested, take for example Industrial Control systems or autonomous vehicles navigating public roads[8],in these kinds of environments the amount of data and heterogenity of the devices make signature based maintenance an impossible task. Expanding on this, the convergence of IT and OT (Operational Technology) has exposed critical industrial machine to a wide area of attack, and therefore shifted what can be considered normal behaviour, as well as the attack surface.[19]

With these new surfaces, the entire landscape changed, particularly with the rise of APT (Advanced Persistent Threats), which can even be state-sponsored threats, that are particularly interested in IO, these APTs usually utilize a technique called "living off the land" which utilizes legitimate software and functions to carry out their attacks, e.g. utilizing group policy changes to gain persistance and access to restricted controls and information, another type of vulnerability that is continuedly exploited are Zero-Day attacks, which due to their nature are impossible to defend against with static hashing, so there's a need to move to a dynamic analysis of behaviour, which is where statistics come into play, so that we see a threat as something that *deviates from a probabilist baseline*, this is what in the modern cyberseucurity field is called **statistical anomaly detection**[3].

Anomaly detection is rooted in the statistical hypothesis that malicious activity is rare and different from normal activity. It does not ask, "Is this specific packet known to be bad?" but rather, "What is the probability that this packet belongs to the distribution of normal traffic observed over the last month?" This shift requires security professionals to abandon binary certainty in favor of probabilistic reasoning. It demands a rigorous understanding of distributions, variance, and correlation. The security analyst of the future must be as fluent in covariance matrices and p-values as they are in firewalls and encryption protocols.

1.1 The Dual Role of Artificial Intelligence: Tool and Target

As statistical methods evolved, they gave rise to Machine Learning (ML) and Artificial Intelligence (AI), which are essentially statistical inference engines operating at scale. AI has become a powerful tool for defense, capable of ingesting terabytes of log data to identify subtle correlations that are impossible to spot for a human analysts[4].AI-driven statistical anomaly detection has been shown to outperform traditional analytical techniques in mitigating cybersecurity risks in complex environments like wireless networks[22].

However, this reliance on AI introduces a recursive vulnerability: the statistical models themselves are now targets. We are witnessing the rise of Adversarial Machine Learning (AML), where attackers exploit the probabilistic nature of AI. By carefully crafting inputs that are statistically indistinguishable from benign data to a human but catastrophic to a model's mathematical logic, attackers can blind surveillance systems or cause autonomous vehicles to misinterpret stop signs[20][23].

Thus, the domain of AI security is not merely about securing the software code but about securing the statistical inference process. It involves using advanced statistical tests such as Kernel Density Estimation (KDE), Maximum Mean Discrepancy (MMD), and Benford's Law to detect when a distribution is being manipulated. The defender must now monitor the monitor, applying statistical tools to ensure that the AI systems protecting the network are not themselves compromised by statistical illusions.

2 Statistical Anomaly Detection in Network Security

Network intrusion detection is the practice of monitoring network traffic for suspicious activity and unauthorized access. While rule-based systems (e.g. Snort) look for known byte sequences, statistical anomaly detection techniques build a model of what is considered "normal" traffic for a network and flags certain deviations. This approach has been widely used since the 2000s, and is currently being amplified by the use of AI.

2.1 Gaussian Assumption and Univariate Analysis

To be able to distinguish normal from anomalous traffic we must assume that normality exists. The Gaussian distribution (or what we called the normal distribution) is essential for anomaly detection because it's a mathematically convenient way to define what we consider normal. In a normal distribution, around 68% of data points fall within one standard deviation (SD, σ) or the mean (μ), 95% within 2 SDs and 99.7% within 3 SDs. Therefore we can statistically define anomalies as those data points that are statistically improbable, e.g. $p < 0.0003$.

In the simplest scenario a security analyst may apply **univariate analysis** to examine a single variable in isolation, take for example the CPU load of a database server, to do so we have multiple ways:

- **Grubb's test** The statistical test is used to detect a single outlier in an univariate dataset that follows a normal distribution[3]. It calculates a G statistic based on the difference between the maximum deviation and the mean.
- **Z-score** Modern systems utilizing the Z-score, which normalizes a data point x by subtracting the mean and dividing by the standard deviation $Z = \frac{x-\mu}{\sigma}$. If the Z-score is greater than 3 this indicates that the data point is an outlier.

But, as we learned, univariate analysis is limited, and for modern cybersecurity applications it is insufficient, if we were to analyze data points like "high outbound network traffic", this might be benign, for example we could be doing an update or a backup, but if an attacker were to be using a low speed or low bandwidth exfiltration technique, they would stay within normal bounds ($< 2\sigma$), and even if we direct it to an unusual IP address at an unusual time, we are not analyzing this variable and therefore we ignore that. So multivariate analysis is something that isn't needed but required for cybersecurity applications.[14]

2.2 Multivariate Analysis and Mahalanobis Distance

In network analysis, variables are often correlated, take, for example, the number of packets, it is typically correlated with the number of bytes transferred, if we were to rely on euclidean distance (which assumes independence), we might not see anomalies that arise when the relationship is broken.

For a more practical example, consider a server that typically sends small packets (high packet count but small number of bytes for each packet) or large file transfers (high count and high packet count). If an attacker were to try and perform a buffer overflow attack, he would send a small number of extremely large packets, if we were to map this into an Euclidean space, this point might still be close to the center of a data cloud, but it has almost no **correlation** to usual traffic.

To actually solve this we use **Mahalanobis Distance**, which measures the distance between a point P and a distribution D, effectively it measures how many standard deviations away P is from the mean of D, while also accounting for the covariance among the variables.[14][16]

The Mahalanobis distance D_M of an observation vector $x = (x_1, x_2, ..., x_n)^T$ from a set of observations with mean vector $\mu = (\mu_1, \mu_2, ..., \mu_n)^T$ and covariance matrix Σ is defined as:

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

where:

- $\mathbf{x}$ is the feature vector of the incoming traffic (packet length, inter-arrival time, port number).
- $\boldsymbol{\mu}$ is the mean vector calculated from the normal training data

The term Σ^{-1} either whitens or decorrelates data, if two variables typically vary together (so we have an high covariance) the matrix Σ captures this. By multiplying the inverse we penalize deviations that go against the known correlation structures.

For example, if Variable A and Variable B usually increase together, a data point where A is high and B is low will result in a large Mahalanobis distance, identifying it as an anomaly even if the individual values of A and B are within their respective normal ranges.

2.3 Payload Detection

Mahalanobis distance has been used to detect anomalies in packet payloads, systems like PAYL compute the byte frequency distribution of incoming payloads and compare them to a historical profile. It is then given a score (refered to as the "distance") and if for example the payload contains a shellcode or exploit, this would alter the natural frequency of bytes found in ASCII text or common protocols, and therefore gives us a higher distance.

The decision is typically governed by a threshold τ:

$$\text{Status}(\mathbf{x}) = \begin{cases} \text{Anomaly} & \text{if } D_M(\mathbf{x}) > \tau \\ \text{Normal} & \text{if } D_M(\mathbf{x}) \leq \tau \end{cases}$$

Because the squared mahalanobis distance of a Gaussian vector still follows a Chi-squared distribution with n degrees of freedom where n is the number of dimensions, the threshold τ can be rigorously derived from Chi-squared tables for whatever significance level is desired.[14]

2.4 Dimensionality Reduction and PCA

A problem that arises when using any high dimensionality dataset, be for statistics or machine learning is the **curse of dimensionality**, as the number of features (dimensions) increases, the computational cost of inverting the covariance matrix (Σ^{-1}) becomes prohibitive ($O(n^3)$), and the data becomes sparse, so distance measuring becomes less impactful.

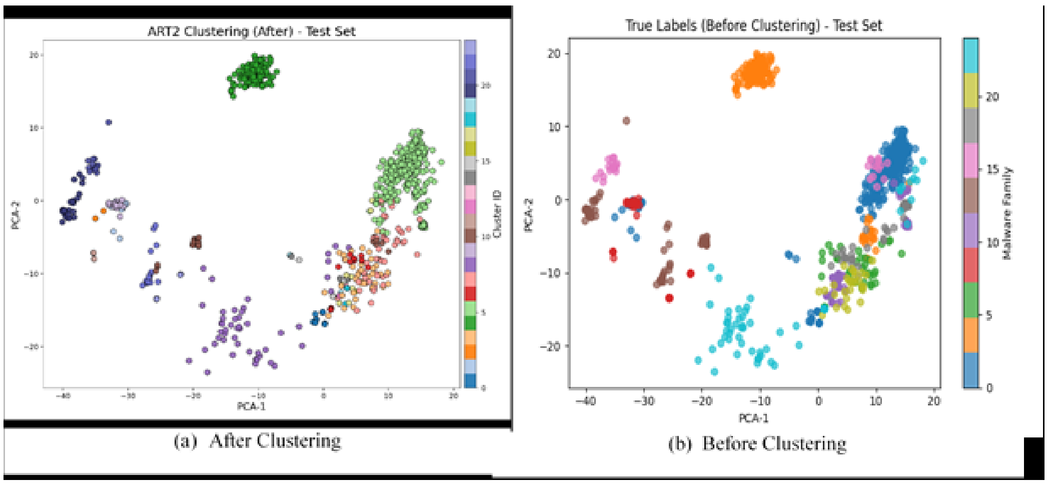
To counteract this we use **Principal Component Analysis** (PCA), which uses an orthogonal transformation to covnert a set of data points that are possibly correlated with each other into a set of values of linearly uncorrelated variables which are then called *principal components.* The first principal component has the largest possible variance (so it accounts for as much of the variability in the data as possible), and each succeeding component has the highest variance possible under the constraint that it is orthogonal to the preceding components.

One of the main ways PCA is used in to visualize the clustering of malware families, most malware analysis often involves extracting thousands of features, this would be impossible manually, but even with powerful computers analyzing this raw data is expensive and difficult to visualize withouth the help of PCA and other dimensionality reduction techniques.

Using PCA we can project high dimensionality data into lower dimensionality spaces (2D or 3D) while preserving the maximum variance. This reveals the underlying clusters of data present in malware families[10], to be more precise:

- **Clustering**: Different Malware families (e.g. ransomware, trojans, worms) often exhibit distinct behavioral patterns. When projected into a lower-dimensional space using PCA, these patterns can form separate clusters, making it easier to identify and categorize malware samples.
- **Anomaly Detection**: PCA can also help identify outliers or novel malware variants that do not fit into existing clusters. These anomalies may represent new threats that require further investigation.

Nowadays, we combine PCA with other techniques and clustering algorithms like K-MEANS and ART2 (Adaptive Resonance Theory) to visualize malware families. The spatial alignment of color-coded clusters in PCA space corresponds to true malware families, proving that the statistical variance capture by PCA corresponds to the function differences in malware code.[13]



(a) After Clustering (b) Before Clustering

As shown in the image above, PCA plots often show lines or trajectories within clusters, these can represent the evolution of a malware family over time (e.g. Racoon Stealer), this allows for the tracking of the genealogy of threats.[10]

2.5 Industrial Control Systems and Operational Technology applications

These statistical methods are particularly useful in ICS and OT, which both prioritize availability over confidentiality, in these kind of networks we can identify 3 major anomalies:

- **Cyber Anomalies**: Old-fashioned intrusions.
- **Operation Anomalies**: System malfunctions or misconfigurations.
- **Service Disruptions**: Availability issues.

The convergence of IT and OT means that industrial networks are now exposed to a wider range of threats. However, OT traffic is often more regular and deterministic than IT traffic (e.g., a sensor polling a valve every 500ms). This regularity actually makes statistical anomaly detection highly effective. Deviations from the strict, deterministic patterns of industrial protocols (like Modbus or DNP3) are easier to detect using statistical baselines than the chaotic traffic of a corporate web network.[19]

3 Time Series Analysis for Network Forecasting

While multivariate looks at the relationships between different variables, **time series analysis** is focused on the relationship between a variable and its past self, network traffic is temporal, and it exhibits trends alongside seasonality and noise, during a normal work day you would normally receive similar day-to-day traffic. So anomaly detection is also a forecasting problem, our statistical model should be able to predict what the traffic should look like at time t, based on previously known data, the anomaly detection system will then compare this prediction $\hat{Y}_t$ with the observed value Y_t, if the error $e_t = |Y_t - \hat{Y}_t|$ exceeds a threshold, then it is considered anomalous, and in turn an alert is raised.[2][21]

3.1 ARIMA Models

Auto Regressive Integrated Moving Average models are widely used for statistical time series forecasting, due to its robust mathematical foundations and the ability to decompose traffic signals into predictable variations and noise.[21]

An ARIMA model is characterized by the parameters (p, d, q)[6]:

- p (Auto-Regressive order): The number of times the raw observations are differenced to make the signal stationary.
- d (Integrated order): The number of times the raw observations are differenced to make the signal stationary.
- q (Moving Average order): The size of the moving average windows, which captures the relationship between the current observation and past errors.

The general equation for an ARIMA(p, d, q) model is essentially a linear regression of the current value against past values and past errors. Using the lag operator B (where $BX_t = X_{t-1}$), it is expressed as:

$$\phi(B)(1 - B)^d X_t = \theta(B)\epsilon_t$$

- $\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$ is the autoregressive polynomial.
- $\theta(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$ is the moving average polynomial.
- $(1 - B)^d$ represents the differencing operation performed d times.
- ϵ_t is the white noise error term[6]

A prerequisite for ARIMA models it is stationarity, we say that a time series is stationary if its statistical properties (mean,variance,autocorrelation) are constant over time. But network traffic is rarely stationary, most of the time there's a trend or change of variance.
So to solve this we use the Integrated d component, which performs differencing.[9] [21]:

If $d = 1$, the model analyzes the change between consecutive points $Y'_t = Y_t - Y_{t-1}$. This removes linear trends.
If $d = 2$, it analyzes the change of the changes, removing quadratic trends. Most network traffic analysis uses $d = 1$ to account for traffic drift.[9]

To actually detect anomalies ARIMA models follow these steps:

- Training: Fitting the parameters (ϕ, θ) on a window of "clean" historical traffic.
- Forecasting: Predicting the next value $\hat{X}_{t+1}$.
- Residual Analysis: Calculating the error $e_{t+1} = X_{t+1} - \hat{X}_{t+1}$

- Thresholding: If the error is significantly large (e.g., outside the 95% confidence interval of the prediction), it indicates that the underlying process generating the traffic has changed—likely due to an attack like a DDoS (Distributed Denial of Service) or a massive data exfiltration.

3.2 Seasonal ARIMA (SARIMA) and ISP variability

Network traffic is seasonal, with the peaks being in the evening (referred to as "streaming hours"), while Corporate traffic peaks in the morning, an ARIMA model might interpret this as an anomaly, therefore to need to account for seasonality, we can extend the ARIMA model to include seasonal components, resulting in the Seasonal ARIMA (SARIMA) model. We denote SARIMA as ARIMA$(p, d, q) \times (P, D, Q)_s$, where s is the length of the season (e.g., 24 for hourly data). The modelincludes seasonal autoregressive and moving average terms:

$$\Phi_P(B^s)\phi(B)\nabla_s^D\nabla^d X_t = \Theta_Q(B^s)\theta(B)w_t$$

This allows our model to correlate traffic at different times,which is esssential for ISPs. Newer Deep Learning models like LSTM (Long Short-Term Memory) are also being experimented upon due to the fact that they can model non-linear temporal dependencies, but ARIMA remains competitive and has an entire ecosystem built around it. If we were to compare them ARIMA is optimal for most cases while ML DL models are useful in complex environments.[18][1][17]

4 Bayesian Statistics

As we've seen in the previous sections, ARIMA deals with temporal trends, but we also need a way to deal with uncertainty and causality, most of the time defenders do not see an attack directly, but rather evidence of it, Bayesian statistics provides us with a framework to update the probability of a breach as we find new evidence.

The biggest challenge in intrusion detection is being able to distinguish true threats from false positives, we can use Bayes' Theorem to calculate the posterior probability of an attack given an alerts:

$$P(\text{Attack}|\text{Alert}) = \frac{P(\text{Alert}|\text{Attack}) \cdot P(\text{Attack})}{P(\text{Alert})}$$

This formula highlights what we call the **base rate fallacy**, even if our Intrusion Detection System (IDS) is highly accurate (high $P(\text{Alert}|\text{Attack})$), if the probability of an actual attack $P(\text{Attack})$ is very low, the posterior probability of an attack given an alert may still be low due to high number of benign events.

Suppose that only 1 in 10,000 events is an attack, a system with a 1% false positive rate will generate 100 false alarms for every 10,000 legitimate events, and the single real attack will be under a mountain of noise. By applying Bayesian analysis security teams account explicitly for these priors and help tune the system to optimize the **Positive Predictive Value**, rather than raw detection rates.

4.1 Bayesian Networks and Attack Graphs

Usually attacks come one after another, as an initial breach only widens the surface area for further attacks, a **Bayesian Network** is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a DAG. We can further specialize these graphs into what we call *Bayesian Attack Graphs*, where nodes represent system states or vulnerabilities, and edges represent the exploits that transition the attacker from one state to another.

Bayesian Attack Graphs allow for the calculation of Reachability Probabilities, the likelihood that an attacker can reach an asset from a certain state.

The Static Reachability Probability $P_b(S_i)$ of a node S_i depends on the successful compromise of its parent nodes and the exploitability of the vulnerability itself.[5] Nodes may have a relationship between each:

- AND Relationship (All preconditions must be met):

  $$P_b(S_i) = P_a(S_i) \cdot \prod_{j=1}^{n} P_b(\text{parent}_j)$$

  Here, $P_a(S_i)$ represents the intrinsic probability of exploiting the vulnerability (often derived from CVSS scores).

- OR Relationship (Any precondition suffices):

  $$P_b(S_i) = P_a(S_i) \cdot (\text{any precondition probability})$$

  .

However though this is useful to understand the subject, these rules are static, and in a real world scenario we need to adapt them to be dynamic, so that it may work even while an attack progresses, therefore we need **Dynamic Reachability**.

If an IDS detects a webserver that has been/is being exploited ($P(S_{web}) = 1$), this value propagates through the network, dramatically increasing the probability of compromise for downstream nodes. The dynamic reachability $P_c(S_j|S_i)$ (probability of parent S_j given child S_i is compromised) involves inverse probabilistic reasoning[5]:

$$P_c(S_j|S_i) = P(S_i|S_j) \cdot \frac{P_b(S_j)}{P_b(S_i)}$$

This allows IDS that incorporate Bayesian Attack Graphs to predict the next most likely targets of an attacker and tighten defences.

4.2 K2 and Junction Trees

Actually querying the graphs (and networks) built by this algorithms is computationally expensive, therefore we use **K2 algorithm**, an heuristic search method used to learn the structure of Bayesian Networks from data. The algoritm starts from a single node and incrementally adds parents that maximize the network's posterior probability, dealing effectively with incomplete records in security logs[24]. Once we have actually built the structure of the Bayesian Network, we can use **Junction Tree Inference** to actually calculate the probabilities. This algorithm transforms the Bayesian Netowrk intro a tree structure where exact inference is tractable. This allows for the efficient propagation of alerts through the graph to update all probabilities simultaneously.[24]

4.3 Parameter Learning and Dirichlet Distributions

We still need to define the conditional probabilities in the nework, therefore we need to estimate parameters (θ) from data. Since attack data is spare we estimate this data using a **Dirichlet Distribution**, which is the conjugate prior to the multinomial distribution. The posterior mean for a probability parameter p, given prior count $K\lambda$ and observed data count N, is[24]:

$$E(p|\text{data}) = \frac{N}{N+K}\left(\frac{x}{N}\right) + \frac{K}{N+K}\lambda$$

This formula balances the observed empirical frequency (x/N) with the prior belief (λ). In the early stages we have a small N, and the system relies on the prior data from previous engagements. As more data is collected (large N), the systems begins to rely on that data. This estimator based on Bayes is mostly needed to stabilize detection systems against the volatility of small samples.

4.4 Naive Bayes Classifiers

To better our spam filters and malware classifiers we can use **Naive Bayes Classifiers**, which simplifies calculations by assuming that all features are conditionally independent given the class, an assumption which is false most of the time, but its effective[7].

The classification rule maximizes the posterior probability:

$$\hat{y} = \underset{k\in\{1,\dots,K\}}{\operatorname{argmax}} P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

Recently, Naive Bayes Classifiers have been modified to address the Zero Probability problem (where a never-before-seen word creates a probability of 0, wiping out the entire calculation) by changing multiplication operations to addition or eliminating zero-probability variables.

5 Detecting Adversarial Attacks

Increasingly more organizations are relying on AI agents to increase performance, bust this does increase the surface are and vectors of attack, we class these kinds of attacks **Adversarial Machine Learning attacks**, where attackers can create adversarial example, inputs with perturbations designed to fool statistical models. An example could be modifying the magic bytes of certain files (e.g. PDFs) to bypass neural network classifiers.

Detecting these attacks is a problem attributed to **Statistical Hypothesis Testing**, the defender's goal is to determine whetever an input comes from a natural distribution of data H_0 or from an adversarial distribution H_A.

To actually compare two distributions we can use **Maximum Mean Discrepancy** (MMD), a non parametric statistical test that can distinguish between two distributions P and Q[15], it embeds the distribution into a Reproducing Kernel Hilbert Space (RKHS)[12]. The squared MMD is the distance between the mean embeddings of the two distrubtions in the RKHS:

$$\text{MMD}^2(P, Q) = \|\mu_P - \mu_Q\|^2_{\mathscr{H}}$$

In practice we use the empirical estimate based on samples $X = \{x_1, \dots, x_n\}$ (benign data) and $Y = \{y_1, \dots, y_m\}$ (incoming batch of suspicious data):

$$\text{MMD}^2(X, Y) = \frac{1}{n^2}\sum_{i,j} k(x_i, x_j) + \frac{1}{m^2}\sum_{i,j} k(y_i, y_j) - \frac{2}{nm}\sum_{i,j} k(x_i, y_j)$$

Here, $k(\cdot,\cdot)$ is a kernel function (like the Gaussian RBF kernel). If the MMD value exceeds a critical threshold derived from permutation testing, the null hypothesis ($P = Q$) is rejected, and the batch Y is flagged as adversarial[25].

MMD is powerful since it is model agnostic, it doesnt need the know the weights of the neural network, only the distribution of inputs, but it does require a batch of samples to actually become statistically significant, making it not as useful for once and done attacks.

5.1 Kernel Density Estimation

KDE is used to detect adversarial examples by analyzing the activation patterns within neural networks, while adversarial examples might look like normal data in the input space, they often land in low-probability regions of the feature space[11].

References

[1] AKvelon. *Time Series and How to Detect Anomalies in Them.* Tech. rep. 2025.

[2] Aayush Bajaj. *https://neptune.ai/blog/anomaly-detection-in-time-series.* Tech. rep. Neptune AI, 2025.

[3] VARUN CHANDOLA. “Anomaly Detection: A Survey”. In: (2009).

[4] SONG WANG JUAN FERNANDO BALAREZO SITHAMPARANATHAN KANDEEPAN AKRAM AL-HOURANI KARINA GOMEZ CHAVEZ and BENJAMIN RUBINSTEIN. *Machine Learning in Network Anomaly Detection: A Survey.* Tech. rep. IEEE, 2021.

[5] Lu Chen. “A Bayesian-Attack-Graph-Based Security Assessment Method for Power Systems”. In: *Electronics* 13.13 (2024).

[6] Wikipedia contributors. “Autoregressive Integrated Moving Average”. In: *Wikipedia* (2025).

[7] Wikipedia contributors. “Naive Bayes classifier”. In: *Wikipedia* (2025).

[8] Meera Sridhar Danial Abshari. “A Survey of Anomaly Detection in Cyber-Physical Systems”. In: *ArXiv* 1.1 (18 Feb 2025).

[9] Duke.edu. *Introduction to ARIMA: nonseasonal models.* Tech. rep. Duke University.

[10] Walid El-Shafai. “Visualized Malware Multi-Classification Framework Using Fine-Tuned CNN-Based Transfer Learning Models”. In: *Appl. Sci* 14.11 (2021).

[11] Reuben Feinman. “Detecting Adversarial Samples from Artifacts”. In: *ArXiv* (2017).

[12] Ruize Gao. “Maximum Mean Discrepancy Test is Aware of Adversarial Attacks”. In: *ArXiv* (2020).

[13] David george. “Static Malware Family Clustering via Structural and Functional Characteristics”. In: *SMU Data Science Review* 7.2 (2023).

[14] Hamid Ghorbani. “MAHALANOBIS DISTANCE AND ITS APPLICATION FOR DETECTING MULTIVARIATE OUTLIERS”. In: *Ser. Math. Inform.* 32.3 (2019).

[15] Praveen Manoharan Kathrin Grosse. “On the (Statistical) Detection of Adversarial Examples”. In: *CISPA* (2017).

[16] Salvatore J. Stolfo Ke Wang. “Anomalous Payload-based Network Intrusion Detection”. In: ().

[17] Vaia I. Kontopoulou. “A Review of ARIMA vs. Machine Learning Approaches for Time Series Forecasting in Data Driven Networks”. In: *Future Internet* 8.15 (2024).

[18] Čejka T Koumar J Hynek K. “CESNET-TimeSeries24: Time Series Dataset for Network Traffic Anomaly Detection and Forecasting.” In: *Sci Data.* 26.12 (2025).

[19] Karel Kuchar and Radek Fujdiak. “Anomaly Detection in Industrial Networks: Current State, Classification, and Key Challenges”. In: *IEEE SENSORS JOURNAL* 25.3 (2025).

[20] Kathrin Grosse†. Praveen Manoharan†. Nicolas Papernot‡. Michael Backes†. Patrick McDaniel‡. “On the Statistical Detection of Adversarial Examples”. In: *CISPA, Saarland Informatics Campus†; Penn State University‡; MPI SWS* (2024).

[21] H.Zare Moayedi. “Arima Model for Network Traffic Prediction and Anomaly Detection”. In: (2008).

[22] Ali Mohanad Faris Mohammed Q. Mohammed Mohammed G. S. Al-Safi. *Statistical Anomaly Detection for Enhanced Cybersecurity Using AI-Based Wireless Networks.* Tech. rep. Ingénierie des Systèmes d’Information, 2024.

[23] Palo Alto Networks. “What Is an Adversarial AI Attack?” In: (2024). https://www.paloaltonetworks.com/cyb
are-adversarial-attacks-on-AI-Machine-Learning.

[24] Manoj Sharma. “Intrusion Detection System using Bayesian Approach for Wireless Network”. In: *International Journal of Computer Applications* 48.0975-888 (2012).

[25] Zhaokun Zhou. “DC-MMD-GAN: A New Maximum Mean Discrepancy Generative Adversarial Network Using Divide and Conquer”. In: *Appl. Sci.* 18.10 (2020).