POLITECNICO
MILANO 1863

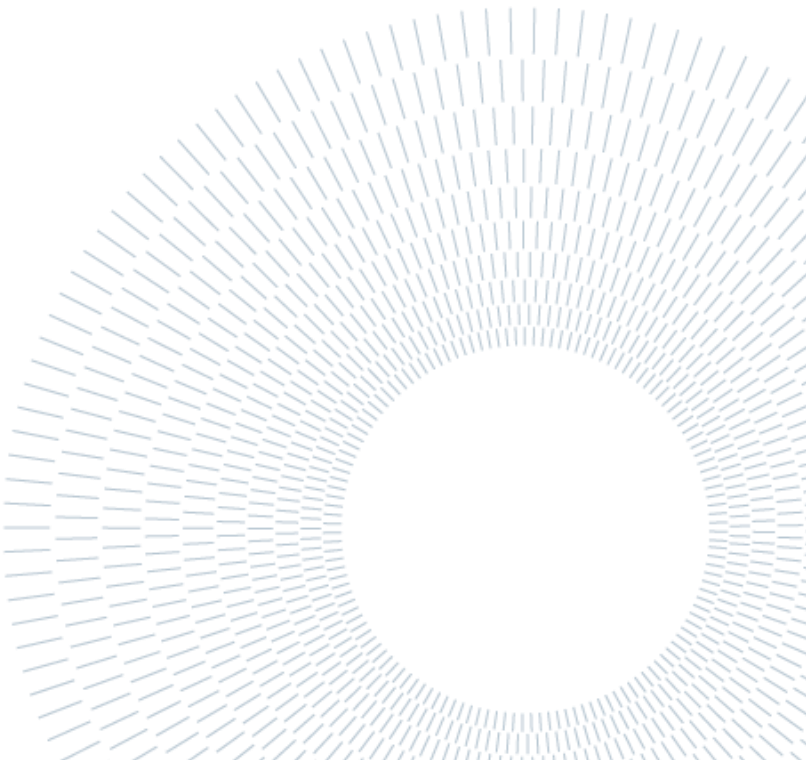
SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Methodologies to Enhance anomaly-based Network Intrusion Detection

TESI DI LAUREA MAGISTRALE IN
TELECOMMUNICATION ENGINEERING
INGEGNERIA DELLE TELECOMUNICAZIONI

Author: **Minhao liao**

Student ID: 10862005
Advisor: Prof. Giacomo Verticale
Co-advisor: Ing. Luca Giacometti
Academic Year: 2024-2025



Abstract

Nowadays, technology involved in cyber attacks is continuously evolving, increasing the complexity of these threats alongside the growing throughput of network infrastructures. To address these challenges, intrusion detection systems increasingly rely on machine learning algorithms capable of handling high packet rates.

In this thesis, we focus on intrusion detection for network attacks by leveraging a pre-existing inference algorithm in dynamic and resource-constrained environments, where lightweight and adaptive anomaly-based approaches are required. We investigate several thresholding methodologies to determine appropriate decision boundaries for anomaly detection. Furthermore, we conduct a systematic performance comparison across multiple datasets, both with and without the inclusion of 2D statistical features, to quantify their contribution to detection accuracy and robustness.

We provide a comprehensive evaluation of three statistical thresholding strategies: a robust distribution-free method, a quantile-based approach, and a Gaussian assumption-based method. These approaches are analyzed with respect to both balanced and highly imbalanced datasets.A unified evaluation pipeline was developed to assess AUC, precision, recall, F1-score, ROC characteristics, parameter sensitivity, and statistical distribution properties for each strategy.

The results indicate that although all methods achieve high AUC values, their practical detection performance strongly depends on attack prevalence and score distribution characteristics. The robust distribution-free and quantile-based thresholds demonstrate high recall but may suffer under rare-attack conditions. In contrast, the Gaussian-based approach can outperform others in extremely sparse anomaly scenarios, though it exhibits higher variance and reduced stability. Overall, the quantile-based method provides the most consistent and robust performance across datasets, effectively balancing recall stability and false-positive control.

This study highlights that threshold design is a critical component in the development of an intrusion detection system. There is no universally optimal strategy; instead, the best choice depends on anomaly frequency and score distribution properties. The findings provide actionable guidance for deploying the selected algorithm in conjunction with a quantile-based thresholding strategy in real-world environments and establish a foundation for future research on adaptive and data-driven thresholding mechanisms for online intrusion detection.

Key-words: Statistical Thresholding; Quantile-based Method

Abstract in italiano

Oggi, le tecnologie impiegate negli attacchi informatici stanno evolvendo in modo continuo, aumentando la complessità di tali minacce insieme alla crescente capacità di trasmissione delle infrastrutture di rete. Per rilevare questi attacchi, i sistemi di Intrusion Detection sono stati progettati facendo affidamento su algoritmi di Machine Learning, con l'obiettivo di operare a velocità compatibili con l'elevato throughput dei pacchetti di rete.

In questa tesi ci concentriamo sulla rilevazione di intrusioni nella rete, sfruttando un algoritmo di inferenza già esistente in ambienti dinamici e con risorse limitate, dove è necessario adottare un approccio leggero, adattivo e basato sull'anomalia. Esploriamo diverse metodologie di soglia per definire un limite superiore nella rilevazione delle anomalie. Inoltre, conduciamo un confronto sistematico delle prestazioni su tutti i dataset, con e senza l'inclusione di caratteristiche statistiche bidimensionali (2D), al fine di quantificarne il contributo in termini di accuratezza e robustezza.

Forniamo una valutazione completa di tre strategie statistiche di thresholding, includendo approcci robusti indipendenti dalla distribuzione, basati sui quantili e fondati sull'assunzione di gaussianità, analizzandone il comportamento sia in dataset bilanciati sia fortemente sbilanciati.

È stata sviluppata una pipeline di valutazione unificata per misurare AUC, precisione, richiamo (recall), F1-score, andamento della curva ROC, sensibilità ai parametri e caratteristiche delle distribuzioni statistiche per ciascuna strategia. I risultati mostrano che, sebbene tutti i metodi raggiungano valori elevati di AUC, le prestazioni pratiche di rilevamento dipendono fortemente dalla prevalenza degli attacchi e dalla distribuzione dei punteggi di anomalia.

Le soglie robuste e basate sui quantili presentano un richiamo elevato, ma risultano meno efficaci in scenari in cui gli attacchi sono rari. Al contrario, l'approccio basato sulla distribuzione gaussiana può offrire prestazioni migliori in condizioni di anomalie estremamente sparse, ma mostra la maggiore variabilità e instabilità.

Nel complesso, il metodo quantile-based si dimostra il più robusto e consistente tra i dataset analizzati, bilanciando la stabilità del richiamo e il controllo dei falsi positivi.

Questo studio evidenzia come la definizione della soglia rappresenti un elemento critico nella progettazione di un sistema di intrusion detection e che non esista una strategia ottimale universale: la scelta migliore dipende dalla frequenza delle anomalie e dalle proprietà della distribuzione dei punteggi.

I risultati forniscono indicazioni operative per l'impiego dell'algoritmo selezionato in combinazione con un approccio quantile-based in ambienti reali e costituiscono una base per future ricerche su meccanismi adattivi e data-driven di thresholding per l'intrusion detection online.

Parole chiave: Sogliatura statistica; Metodo basato sui quantili

Contents

1 Introduction

Over the past years, the frequency and sophistication of attacks on computer networks have continued to grow [1]. To defend against such threats, network intrusion detection systems (NIDS) have become a fundamental component of modern network security architectures. A NIDS, whether implemented as hardware or software, continuously monitors network traffic passing through key points in the system to identify suspicious or malicious behavior. Once an intrusion attempt is detected, the system generates an alert to notify the network administrator.

Several machine learning models, such as deep neural networks and ensemble methods, have been proposed for NIDS. However, these models are often computationally expensive, requiring significant memory and processing resources, which limits their deployment on resource-constrained devices such as routers or gateways. Moreover, many of them rely on labeled datasets and supervised training, which are impractical for real-time and adaptive security monitoring.

Many studies have demonstrated that neural network–based pattern recognition using the backpropagation algorithm can effectively detect intrusions. Among various neural network classifiers, the backpropagation neural network (BPN) has been found to be particularly efficient for building intrusion detection systems.

Although neural network–based classifiers have shown strong detection capability, their application in resource-constrained environments remains limited.

Training neural network–based intrusion detection systems presents several practical challenges that limit their real-world deployment. First, supervised models require all labeled data to be locally available, which is infeasible for simple network gateways, where even a single hour of traffic may contain millions of packets. Although some studies suggest transferring data to remote servers for training, this approach introduces considerable communication overhead and does not scale well across large networks. Second, supervised learning itself is both time-consuming and costly. The notion of normal traffic varies between environments, while attack patterns constantly evolve over time [10], making it difficult to maintain a comprehensive and up-to-date repository of labeled malicious samples. Moreover, such models operate under a closed-world assumption—they can only recognize attack types seen during training, which is unrealistic for modern, dynamic networks.

Due to these limitations, traditional ANN-based intrusion detection systems are not well suited for scalable or real-time deployment. To address these challenges, researchers have proposed lightweight and adaptive frameworks capable of learning network behavior online without labeled data. One notable example is Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection [1], which employs an unsupervised and incremental learning approach that allows efficient operation even on resource-constrained devices.

Kitsune achieves this through three main design principles. First, it performs online processing, meaning that each instance of network traffic is analyzed once and then immediately discarded, ensuring low memory usage and real-time responsiveness. Second, it adopts an unsupervised learning paradigm, eliminating the need for labeled data and enabling continuous adaptation to new or evolving network behaviors. Finally, Kitsune maintains low computational complexity, ensuring that packet processing rates remain higher than incoming traffic rates, which prevents system bottlenecks.

These properties make Kitsune a highly practical solution for real-time intrusion detection in distributed and resource-constrained network environments.

In this thesis, we improve the Kitsune framework by systematically investigating the impact of threshold selection on its detection performance. Specifically, we evaluate multiple statistical thresholding strategies, analyze their behavior across diverse network environments, and explore parameter sensitivity to improve the accuracy and reliability of anomaly detection. We also investigate the influence of 2D statistical features by conducting comparative experiments with and without 2D feature representations to quantify their contribution to detection performance. Through comprehensive experiments on traffic datasets, we aim to provide a deeper understanding of threshold design for online NIDS and suggest which threshold methodology should be applied when deploying Kitsune in resource-constrained settings.

2 Background

2.1. Feedforward Neural Network (FNN)

Deep Feedforward Networks, also known as feedforward neural networks, are typical models of deep learning. The goal of a feedforward network is to approximate some function f^*.For example, in a classification problem, $y = f^*(x)$maps an input $x$to a class label y.The network defines a mapping $y = f(x;\theta)$and learns the parameters θso that the resulting function $f(x;\theta)$is a good approximation of f^*.

This model is called feedforward because information flows through the function f from the input x, through intermediate computations, and finally to the output y.There are no feedback connections between the output of the model and the model itself.

2.1.1. Executing an FNN

Feedforward neural networks are called networks because they typically involve the composition of multiple functions connected together. Such a model can be viewed as a chain of functions, where each function represents one layer of the network, and the overall network describes how these functions are composed. For example, we can define three functions $f^{(1)}, f^{(2)}, f^{(3)}$that are connected in series to form

$$f(x) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(x)\right)\right) \tag{2.1}$$

This kind of structure is the most common form of neural networks. In this case, $f^{(1)}$is called the first layer, $f^{(2)}$the second layer, and so on. The total number of layers in the model determines its depth, which gives rise to the term deep learning. The last layer of a feedforward network is referred to as the output layer. During the training process, we attempt to make the network's output $f(x)$match the target output y.The training data provide the desired outputs (labels) for different inputs, where each example (x, y)satisfies $y \approx f^*(x)$.The learning algorithm must determine how to adjust the internal layers so that the overall function $f(x;\theta)$approximates the desired mapping $f^*(x)$.The intermediate layers between the input and output are not directly specified by the training data; instead, the algorithm learns how to use them to achieve a good approximation of f^*.These intermediate layers are therefore called hidden layers.

Another key aspect in the design of a neural network is its architecture, which refers to the overall structure of the network, specifically, how many units it contains and how these units are connected to one another.

Most neural networks are organized into layers of units. These layers are typically arranged in a chain-like structure, where each layer represents a function of the output from the previous layer.In such an architecture, the first layer can be expressed as:

$$h^{(1)} = g^{(1)}\left(W^{(1)T}x + b^{(1)}\right) \tag{2.2}$$

and the second layer as:

$$h^{(2)} = g^{(2)}\left(W^{(2)T}h^{(1)} + b^{(2)}\right) \tag{2.3}$$

and so on. Here, f denotes the activation function applied by each neuron. In Kitsune, the sigmoid activation is adopted, defined as:

$$f(x) = \frac{1}{1 + e^{-x}}. [1] \tag{2.4}$$

2.1.2. Autoencoders

Autoencoders are a type of neural network that, through training, attempt to reproduce their input at the output. An autoencoder contains an internal hidden layer that produces a code (or latent representation) to represent the input data. The network can be viewed as consisting of two main components: an encoder $h = f(x)$, which maps the input $x$to a code h,and a decoder $r = g(h)$, which reconstructs the original input from that code. If an autoencoder simply learns to make $g\left(f(x)\right) = x$ for all inputs

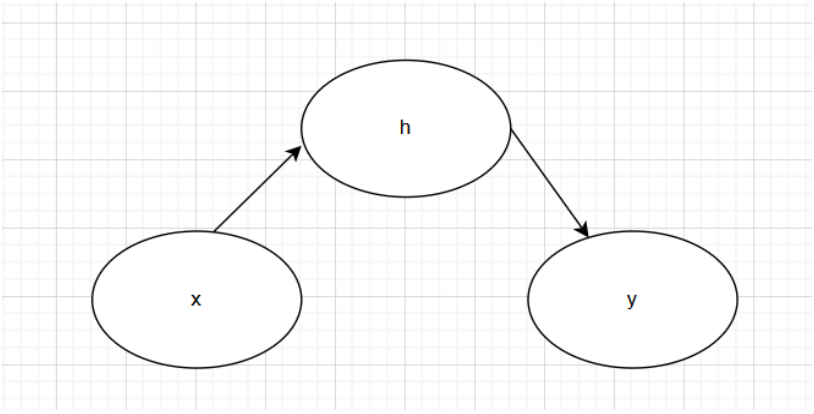


Figure 2.1: Basic structure of an autoencoder.

In general, an autoencoder trained on a dataset Xlearns an internal representation that enables it to accurately reconstruct inputs drawn from the same data distribution. When a new instance does not conform to the learned patterns or features in X, the network's reconstruction tends to diverge from the original input, producing a larger reconstruction error. For a given input vector $\tilde{x}$, the reconstruction error is commonly

quantified by the Root Mean Squared Error (RMSE) between the input $\tilde{x}$ and its reconstructed output $\tilde{y}$, defined as:

$$\mathrm{RMSE}(\tilde{x}, \tilde{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.5}$$

where n represents the dimensionality of the input vector. A higher RMSE value indicates a greater deviation between the input and its reconstruction, suggesting that the instance may be anomalous with respect to the data distribution learned by the autoencoder. Let ϕ denote the anomaly threshold, initialized to -1, and let $\beta \in [1, \infty]$represents a user-defined sensitivity parameter. The autoencoder can be employed for anomaly detection through the following procedure:

1) Training Phase:

The autoencoder is first trained using benign (normal) data. For each instance $\tilde{x}_i$in the training set X:

a) Compute the reconstruction error $s = \mathrm{RMSE}\big(\tilde{x}_i, h_\theta(\tilde{x}_i)\big)$.

b) Update the threshold: if $s \geq \phi$, then set $\phi = s$.

c) Update the model parameters θthrough learning from $\tilde{x}_i$.

2) Detection Phase:

When a new, unseen instance $\tilde{x}$is observed:

a) Compute its reconstruction error $s = \mathrm{RMSE}\big(\tilde{x}, h_\theta(\tilde{x})\big)$.

b) Classify the instance: if $s \geq \phi\beta$, the instance is considered anomalous and an alert is raised.[1]

2.2. Network Security

Computer networks underpin many critical infrastructures and services, including financial systems, healthcare platforms, and industrial control environments. Although preventive security mechanisms such as firewalls, access control, and virtual private networks (VPNs) serve as the first line of defense, they cannot guarantee complete protection against increasingly sophisticated cyber threats. Zero-day vulnerabilities, insider attacks, and the complexity of modern network infrastructures necessitate the adoption of complementary detection mechanisms capable of identifying malicious activities that evade traditional defenses.

2.2.1. Detection Approaches

Intrusion detection approaches are generally classified into two main paradigms, each representing a different security philosophy.

Signature-based detection relies on predefined patterns of malicious activity, such as specific byte sequences, packet structures, or known indicators of compromise. This approach is computationally efficient and highly accurate when identifying previously observed attacks, since signatures act as explicit fingerprints of malicious behavior. However, its effectiveness is limited when facing novel exploits or polymorphic malware, which can evade detection by modifying their observable characteristics.

In contrast, anomaly-based detection models the normal behavior of a network or host system and raises alerts when significant deviations are observed. This paradigm enables the identification of previously unseen attack patterns and is particularly suitable for detecting zero-day threats. Its adaptability, however, often leads to higher false positive rates, as legitimate but unusual activities may also be classified as anomalous. The work presented in this thesis follows this direction, consistent with the anomaly-based approach adopted by Mirsky et al. in the Kitsune framework [10].

In practice, statistical and machine learning techniques are widely used to construct behavioral baselines and refine the definition of normal activity. While anomaly-based methods extend detection capabilities beyond known threats, excessive false alarms may reduce system reliability and operator trust. For this reason, modern intrusion detection systems increasingly adopt hybrid strategies that combine signature-based detection for known attacks with anomaly detection for emerging threats, often leveraging machine learning to balance detection accuracy, coverage, and operational efficiency.

2.2.2. AI-Enhanced Network Security

Machine learning (ML) has become a key component of modern intrusion detection systems. Supervised algorithms such as decision trees, random forests, and support vector machines (SVMs) can be trained on labeled datasets to classify network traffic as benign or malicious. In contrast, unsupervised approaches, including clustering methods and one-class SVMs, focus on modeling normal behavior and generating alerts when deviations are detected, thereby reducing reliance on labeled data.

Beyond traditional ML techniques, deep learning methods have attracted increasing attention due to their ability to capture complex patterns in network traffic. Convolutional neural networks (CNNs) can learn spatial structures within packet representations, while recurrent neural networks (RNNs) model temporal dependencies across traffic flows. Autoencoders are particularly popular in anomaly detection: when trained on normal traffic, they learn compact latent representations and signal anomalies when reconstruction errors exceed a predefined threshold.

Despite these advances, deploying ML-based intrusion detection in real-world environments remains challenging. Models often require representative datasets, which are difficult to obtain in practice. They must also cope with class imbalance, where attack traffic represents only a small portion of overall traffic, as well as concept

drift caused by evolving network behavior and potential adversarial manipulation. In addition, detection systems must operate under strict performance constraints, ensuring high throughput and low latency. Consequently, designing AI-enhanced intrusion detection systems requires balancing detection accuracy, computational efficiency, and robustness to meet the demands of modern network environments.

2.3. THRESHOLD SELECTION METHODS

2.3.1. UNSUPERVISED THRESHOLD SELECTION METHODS

Unlike supervised threshold-selection approaches, unsupervised methods determine decision thresholds without relying on ground-truth labels. Instead, they typically use statistical assumptions about the anomaly-score distribution to identify a threshold that separates normal and abnormal observations. For these methods to be effective, anomaly scores should exhibit distinguishable statistical characteristics between normal and anomalous behavior.

In some cases, unsupervised threshold-selection techniques operate using only anomaly scores derived from normal data. Under this assumption, the threshold is set near the upper boundary of the normal-score distribution, so that observations exceeding this limit are considered anomalous. This approach is particularly useful in scenarios where labeled attack data are unavailable or difficult to obtain.

The unsupervised threshold-selection methods reviewed in this work are based on statistical modeling of anomaly scores. They are summarized and categorized according to their underlying assumptions and implementation strategies. Heuristic approaches tailored to specific datasets or anomaly-detection models are not considered here, as the focus is on general-purpose statistical thresholding techniques.

2.3.2. SUPERVISED THRESHOLD SELECTION METHODS

Supervised threshold-selection methods rely on the availability of labeled data, where each anomaly score is associated with a ground-truth label indicating whether the observation is normal or malicious. Using both anomaly scores and labels, thresholds can be selected so that the predicted classification outcomes closely match the true labels.

The core objective of supervised threshold selection is to optimize a chosen evaluation metric, such as accuracy, F1-score, precision–recall balance, or other classification-performance measures. In this context, anomalous observations are typically treated as the positive class, while normal traffic is considered the negative class.

Supervised threshold-selection approaches identified in the literature differ mainly in the evaluation metric they optimize and in how candidate thresholds are generated and validated. These methods generally provide reliable threshold estimates when

labeled datasets are available, but their applicability may be limited in real-world anomaly-detection scenarios where labeled attack data are scarce or incomplete.

2.4. Kitsune framework

The Kitsune framework consists of several modular components, each responsible for a specific stage in the online network intrusion detection pipeline:

In this work, we exclude the implementation details of the Packet Capturer and Parser, as they do not constitute novel contributions. Instead, our analysis focuses on the Feature Extractor (FE), Feature Mapper (FM), and Anomaly Detector (AD), which together form the core of the proposed system. Notably, both the FM and AD are task-independent and can be generalized to other online anomaly detection applications.

Feature Extractor (FE):

The feature extraction module is designed to efficiently obtain real-time statistical features from network traffic streams to support online anomaly detection. The framework maintains incremental statistics for each network channel using a damped-window mechanism, which enables lightweight updates with 0(1) computational complexity and minimal memory overhead, thereby achieving both high throughput and real-time processing capability.

Feature Mapper (FM)

The Feature Mapper (FM) is designed to transform the n-dimensional feature vector $\tilde{x}$ into a set of k smaller sub-instances $v = \{\tilde{v}_1, \tilde{v}_2, \dots, \tilde{v}_k\}$, each corresponding to an individual autoencoder in the ensemble layer of the Anomaly Detector (AD). To achieve this, the FM employs an agglomerative hierarchical clustering algorithm that groups correlated features into subsets (subspaces) containing no more than m features each, where m is a user-defined parameter. This procedure ensures that every feature is assigned exactly once while maintaining low computational complexity and preserving statistical dependencies among features.

The hierarchical clustering process begins by treating each feature as an independent cluster. At each iteration, the two clusters with the smallest distance are merged until the maximum cluster size constraint m is reached. In our implementation, the distance between features is defined using the correlation distance, which measures the linear dependency between two feature vectors. For two features u and v, the correlation distance is computed as:

$$d_{cor}(u, v) = 1 - \frac{(u - \bar{u}) \cdot (v - \bar{v})}{\| u - \bar{u} \|_2 \| v - \bar{v} \|_2} \tag{2.6}$$

where $\bar{u}$ and $\bar{v}$ represent the mean values of u and v, respectively.A smaller $d_{cor}(u, v)$ indicates a stronger correlation, implying that the features share similar

temporal or statistical behavior and are thus more likely to be grouped into the same cluster.The parameter m directly determines the number of autoencoders in the ensemble—smaller values of $m$result in more, simpler models, whereas larger values produce fewer but more complex ones.

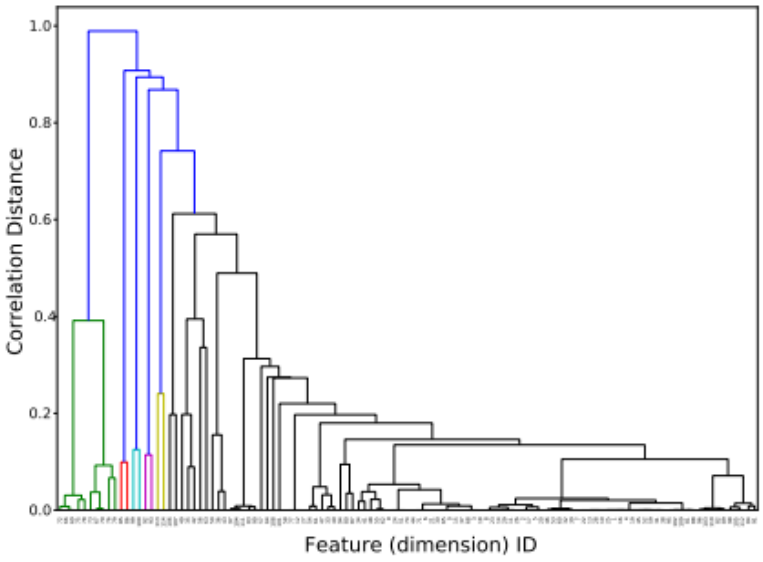


Figure 2.2: Hierarchical clustering of features based on correlation distance

Anomaly Detector (AD)

In the execute mode, KitNET ceases to modify its internal weights or parameters. Instead, it performs only a forward pass across all layers and computes the reconstruction error (RMSE) produced by the output layer $L^{(2)}$. This value serves as an indicator of how unusual the current network instance is, based on the learned relationships among the subspaces represented in v.

To illustrate, suppose two autoencoders $\theta_i$and $\theta_j$in the ensemble layer $L^{(1)}$exhibited correlated RMSE values during training. If, during execution, this correlation disappears, the system interprets it as a more significant anomaly than a simple increase in their individual errors. Essentially, the output layer $L^{(2)}$has learned to model such interdependencies and other non-linear patterns, allowing its reconstruction error to highlight deviations from normal behavior.

The anomaly score produced by KitNET is the RMSE value $s \in [0, \infty]$. A higher score indicates a greater degree of abnormality. To determine whether an instance is anomalous, a threshold ϕmust be chosen. A straightforward method is to set $\phi$to the maximum score observed during training, assuming that all training data are benign. Alternatively, ϕcan be derived statistically—for instance, by fitting the training RMSE distribution to a log-normal or other non-standard distribution and triggering an alert

when s falls in a region of very low probability. The optimal thresholding strategy depends on the specific deployment context.

3 Related Work

In recent years, numerous approaches have been proposed to improve the accuracy and effectiveness of network intrusion detection systems (NIDS). Researchers have explored various strategies, ranging from traditional signature-based mechanisms to advanced anomaly-based and machine learning techniques. These methods aim to enhance detection capability, particularly against novel or previously unseen attacks.

A critical component in anomaly-based detection systems is the selection of an appropriate decision threshold, which determines how anomaly scores are translated into binary classification outcomes. Improper threshold selection may lead to excessive false positives or missed attacks, especially in dynamic network environments.

Freeman and Moisen (2008) conducted a comprehensive comparison of different threshold selection criteria for binary classification models. They evaluated several commonly used approaches such as maximizing accuracy, maximizing Cohen's Kappa, balancing sensitivity and specificity, and matching predicted to observed prevalence ,and analyzed their performance in terms of predicted prevalence and classification agreement. Their results showed that no single threshold criterion performs best in all cases; instead, the optimal choice depends on the dataset characteristics and the specific evaluation objective. This study highlights the importance of selecting an appropriate threshold rather than using a fixed value like 0.5, especially when dealing with imbalanced data or probabilistic outputs [2].

In network intrusion detection systems (NIDS), most approaches to improving detection accuracy focus on feature selection or choosing an appropriate detection threshold.

Feature selection refers to the process of identifying, from a large set of features, those that have the most significant impact on the final prediction results; in other words, it aims to eliminate features that contribute little or no value to the model's performance.

Recent studies on improving the accuracy and scalability of network intrusion detection systems (NIDS) have explored different directions ranging from big-data integration to adaptive learning and feature optimization. Deepak (2020) focused on scalability, employing XGBoost within a PySparkling Water [3] environment to parallelize the training process and efficiently manage large network datasets. His approach demonstrated that combining distributed data processing and gradient-

Boosted decision trees can reduce computational cost while maintaining competitive detection accuracy. In contrast, Dong et al. (2023) aimed to improve the learning capability of decision ensembles through a Gradient-Boosted Neural Decision Forest, which integrates neural feature extraction with hierarchical tree learning. This hybrid design enhances generalization in complex, high-dimensional spaces, an advantage for detecting subtle or evolving attack behaviors. More recently, Faizin et al. (2024) [4] concentrated on feature-level optimization, introducing a threshold-based feature selection strategy that prunes redundant attributes before classification. Using mutual-information scoring and adaptive thresholds, their system achieved notable gains in accuracy and F1-score across several benchmark datasets while significantly lowering training time.

Together, these studies illustrate the complementary nature of current NIDS research: scalable infrastructure, adaptive model architectures, and efficient feature selection represent three converging paths toward building faster, more accurate, and resource-aware intrusion detection frameworks. The model demonstrates that combining big-data frameworks and ensemble learning can significantly enhance the overall performance and responsiveness of NIDS.

Another approach to improving the accuracy of NIDS is by setting appropriate detection thresholds.

Jung et al. (2004) proposed a statistical method for detecting port-scanning behavior based on Sequential Hypothesis Testing (SHT)[5]. Instead of using fixed rules, their approach continuously evaluates connection outcomes, such as success or failure rates, to determine whether a host's behavior indicates scanning activity. This adaptive statistical process enables faster and more accurate detection while reducing false positives, laying the groundwork for later threshold-based anomaly detection methods in NIDS.

Chou and Wang (2015) proposed an adaptive intrusion detection approach tailored for cloud computing environments [6], where traffic characteristics and resource availability change dynamically. Their system continuously monitors network behavior and adjusts its detection parameters in real time according to workload and attack intensity. By combining machine learning-based anomaly detection with adaptive thresholding, the model maintains high accuracy even under fluctuating cloud traffic patterns. The authors emphasized that static detection models often fail in elastic infrastructures and demonstrated that adaptive tuning can effectively balance detection sensitivity and computational cost in virtualized settings.

4 system model

This section presents the system used in this study, which is based on Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection (Mirsky et al., 2018). The original Kitsune framework is briefly introduced to provide context for the subsequent experiments. In this work, two main components of the system were adjusted: the feature extraction module and the threshold selection mechanism with the goal of evaluating their impact on detection accuracy. The following subsections describe the baseline system and detail the modifications introduced in this study.

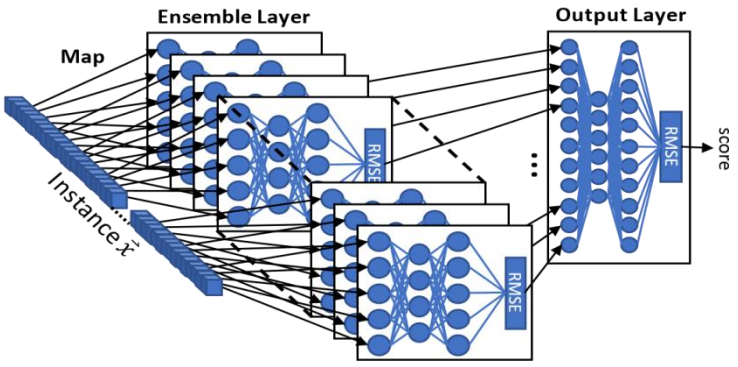


Figure 4.1: Architecture of the KitNET-based anomaly detection model[1]

Detail of Feature Extractor (FE):

When a packet is received, Kitsune extracts a brief behavioral profile describing how the sender and receiver have communicated over recent moments. The extracted features capture patterns such as the general activity of the sender, the flow between the two hosts, and the exchange through specific communication channels or sockets. These statistics provide a compact view of the network's short-term behavior. If certain attributes are not relevant for a given packet, their values are set to zero, ensuring that the resulting feature vector remains uniform in structure.

The extracted features can be conceptually divided into one-dimensional (1D) and two-dimensional (2D) types, depending on whether they describe individual traffic statistics or relationships between them.

1D features capture *independent statistical properties* of network entities, such as hosts, IP pairs, or sockets, over short temporal windows. These include fundamental traffic

measurements that characterize the local behavior of a host or connection. Examples of 1D features are:

- packet counts and byte volumes sent or received by a specific host or channel,
- average and variance of packet lengths or inter-arrival times,
- inbound/outbound traffic ratios,
- counts of active connections or ports observed in recent time windows.

Such 1D statistics provide a snapshot of how active or stable a communication endpoint is. They are computed for different aggregation levels, including the source MAC–IP pair, source IP, IP channel (source–destination pair), and TCP/UDP socket. These features are independent and directly serve as input to the Feature Mapper.

2D features, on the other hand, capture *relationships and dependencies* among different 1D statistics. They are derived to represent how various traffic dimensions evolve or interact with each other. Typical examples include:

- correlations between incoming and outgoing byte rates,
- ratios between sent and received packets for a given connection,
- differences or growth rates of traffic metrics across consecutive time windows,
- cross-entity comparisons (e.g., between two sockets or IP flows belonging to the same host).

These 2D features encode behavioral patterns that cannot be observed from isolated statistics alone. By modeling such correlations, the system can better detect subtle anomalies that involve coordinated or time-dependent changes in traffic.

Type	Statistic	Notation	Calculation
1D	Weight	w	w
	Mean	μ_{S_i}	LS/w
	Std.	σ_{S_i}	$\sqrt{\lvert SS/w - (LS/w)^2 \rvert}$
2D	Magnitude	$\lVert S_i, S_j \rVert$	$\sqrt{\mu_{S_i}^2 + \mu_{S_j}^2}$
	Radius	R_{S_i,S_j}	$\sqrt{\left(\sigma_{S_i}^2\right)^2 + \left(\sigma_{S_j}^2\right)^2}$
	Approx. Covariance	Cov_{S_i,S_j}	$\frac{SR_{ij}}{w_i + w_j}$
	Correlation Coefficient	P_{S_i,S_j}	$\frac{Cov_{S_i,S_j}}{\sigma_{S_i}\,\sigma_{S_j}}$

Figure 4.2: Statistical features used in Kitsune feature extraction.

Details of Anomaly Detector (AD):

The Anomaly Detector (AD) is the core component of the system, responsible for determining whether incoming packets are normal or anomalous. It contains a specialized autoencoder that reconstructs input features and measures the reconstruction error to assess abnormality.

The Ensemble Layer[1] consists of multiple three-layer autoencoders, each trained on a specific subspace of the input vector. In training mode, they learn the normal patterns of their subspaces; in both training and execution, each autoencoder outputs its reconstruction error, which reflects the subspace's level of abnormality.

The Output Layer[1] is a three-layer autoencoder that models the normal reconstruction errors of the Ensemble Layer and produces the final anomaly score, capturing both correlations among subspaces and inherent noise in network traffic.

The Anomaly Detector (AD) functions in two operational modes: train-mode and execute mode. During train-mode, the AD incrementally learns the statistical characteristics of normal network traffic. Each autoencoder in the Ensemble Layer adapts its weights to capture the normal relationships within its assigned feature subspace, while the Output Layer learns to model the collective behavior of these subspaces based on their reconstruction errors. In execute-mode, the learned parameters remain fixed. Incoming packets are propagated through the same architecture, and the system computes the reconstruction error at each layer. The final error produced by the Output Layer serves as an anomaly score, reflecting the degree of deviation from previously learned normal patterns.

5 Testing and Performance

This chapter presents the implementation of the proposed improvements to Kitsune. The objective is to enhance Kitsune's anomaly-detection performance by introducing. We also have to evaluate the performance of ktisune when when the 2D statistics are removed and by optimizing the threshold-selection strategy used for binary decision making.

Based on the methodology discussed in the previous chapter, the implementation is organized into two main stages. First, we evaluate and compare the system's detection performance both with and without the inclusion of 2D features, considering metrics such as accuracy and false-positive rate.

Second, we implement and compare three different threshold-selection strategies for anomaly classification. The purpose is to investigate how different decision-boundary mechanisms impact detection sensitivity and robustness, and to identify which method offers the most consistent trade-off between detection rate and false alarms.

5.1. Performance comparison of the system with and without 2-D statistical features

5.1.1. Metrics Explanation

To evaluate the anomaly-detection performance, the following metrics are employed:

• ROC Curve (Receiver Operating Characteristic Curve)

The ROC curve illustrates the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) across different anomaly-score thresholds. In other words, it evaluates how the model performs when the decision boundary varies from very strict to very liberal.

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + TN} \tag{5.1}$$

The curve starts at (0,0) and ends at (1,1). Each point on the curve corresponds to a specific threshold value.

- A higher TPR with a lower FPR indicates better detection performance.
- A curve closer to the top-left corner (0,1) represents a more effective detector, achieving high detection while minimizing false alarms.
- A diagonal line from (0,0) to (1,1) represents random guessing, meaning the detector has no discriminative ability.

Thus, ROC analysis provides a threshold-independent evaluation, making it especially useful in anomaly detection scenarios where choosing the optimal threshold may not be straightforward.

• Precision

Precision reflects the proportion of correctly detected anomalies among all instances classified as anomalous. High precision indicates that the system generates fewer false alarms, which is critical in practical NIDS deployments.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

• Recall (True Positive Rate)

Recall measures the proportion of actual anomalies that are correctly detected. Higher recall means the detector is able to identify more attacks, avoiding missed threats.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

• F1 Score

The F1 score is the harmonic mean of precision and recall. It balances false alarms and missed detections, providing a single metric for overall detection quality.

$$F1 = 2 \times \frac{Precision \cdot Recall}{Precision + Recall} \tag{5.4}$$

Unlike accuracy, which can be misleading in highly imbalanced datasets, F1 is more suitable for anomaly-detection scenarios where attack samples are typically scarce.

5.2. Performance Comparison of Three Different Threshold Selection Methods

The data used in this work originates from the Kitsune Network Attack Dataset introduced in [1]and made available on Kaggle.

In this study, we compare three threshold selection methods across nine network datasets. These datasets cover both low-anomaly-ratio and high-anomaly-ratio scenarios, allowing us to analyze the strengths and limitations of each method under different anomaly distributions. Our goal is to identify which thresholding strategy performs more robustly across diverse traffic conditions.

To evaluate thresholding behavior in an unsupervised setting, we adopt the following three unsupervised threshold selection strategies:

5.2.1. Percentile Method

In this method, the threshold is determined by the k-th percentile of the anomaly score distribution. That percentile corresponds to the score value below which k percent of the observations fall. Consequently, any score exceeding this value is classified as anomalous.[7]

calculated using the following equation:

$$T = perc(k, X) \tag{5.5}$$

where perc computes the k-th percentile of the data, with k being a positive integer, and Xreferring to the anomaly score set.

it's not sensitive to extreme outliers and it uses an empirical distribution of the anomaly score to set a static or adaptive threshold.[7]

5.2.2. K-SIGMA Method

final threshold value is calculated as follows:

$$T = \mu + k \times \sigma \tag{5.6}$$

in this formulation, μdenotes the mean anomaly score, σ its standard deviation, and ka positive scalar parameter. The method is rooted in the assumption that the score distribution can be approximated by a normal distribution.

5.2.3. Median + k × MAD

A robust alternative is the Median + MAD method, which sets the threshold as the median score plus k times the median absolute deviation. This approach avoids Gaussian assumptions and resists outliers, making it effective for anomaly-score distributions with noise or heavy tails.

$$T = median(x) + k \cdot MAD \tag{5.7}$$

$$MAD = median(|x_i - median(x)|) \tag{5.8}$$

where:

- x is the set of all anomaly scores (RMSE values) collected during the training phase,
- x_i is the anomaly score of the i-th sample, and
- k is a positive tuning parameter.

We apply these three thresholding methods across nine datasets and observe the resulting NIDS performance.

5.3. Data Source

We conducted experiments on nine datasets, and the table below shows the anomaly distribution for each dataset.

Table 5.1: Dataset Traffic Distribution and Attack Ratios

Dataset	Total Packets	Attack Samples	Attack Rate
Active_Wiretap	2,223,689	923,216	41.52%
Mirai_Botnet	709,137	642,516	90.61%
ARP_MitM	2,449,267	1,145,272	46.76%
SYN_DoS	2,716,276	7,038	0.26%
Fuzzing	2,189,139	432,783	19.80%
Video_Injection	2,417,401	102,499	4.24%
SSDP_Flood	4,022,266	1,439,604	35.79%
OS_Scan	1,642,851	65,700	4.00%

5.4. Comparison With and Without 2D Features

5.4.1. evaluate the performances of ktisune WHEN WE REMOVE THE 2D STATISTICS

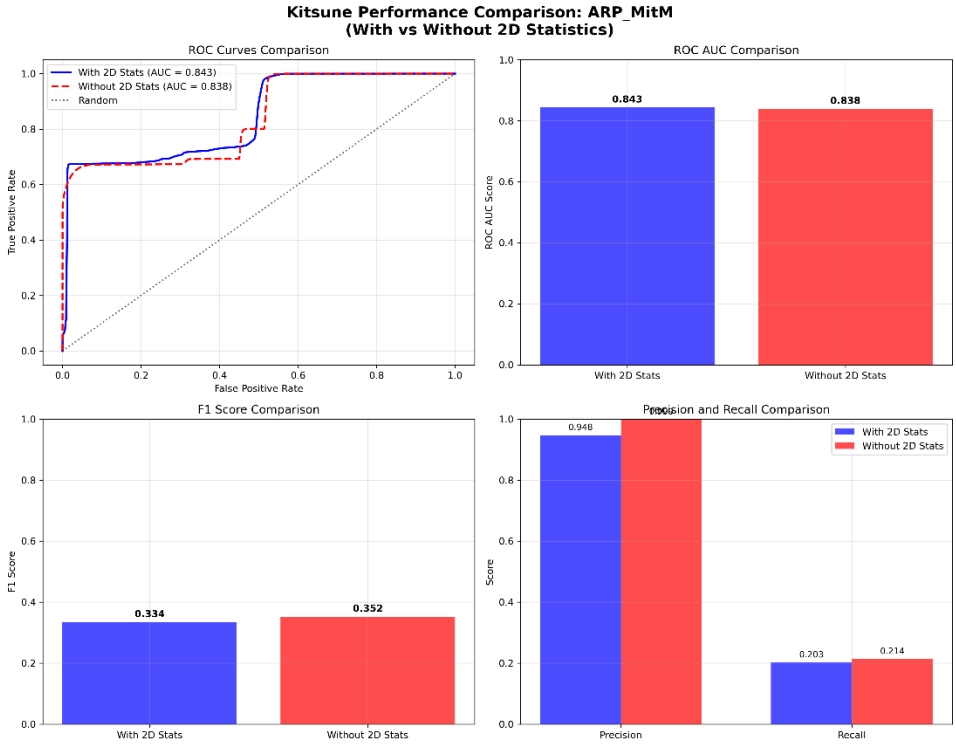


Figure 5.1: Performance Comparison With and Without 2D Features on ARP-MitM

Figure 5.1 shows that the inclusion of 2D statistical features leads to only marginal changes in AUC and F1-score. This suggests that 2D features provide limited performance improvement on the ARP-MitM dataset.

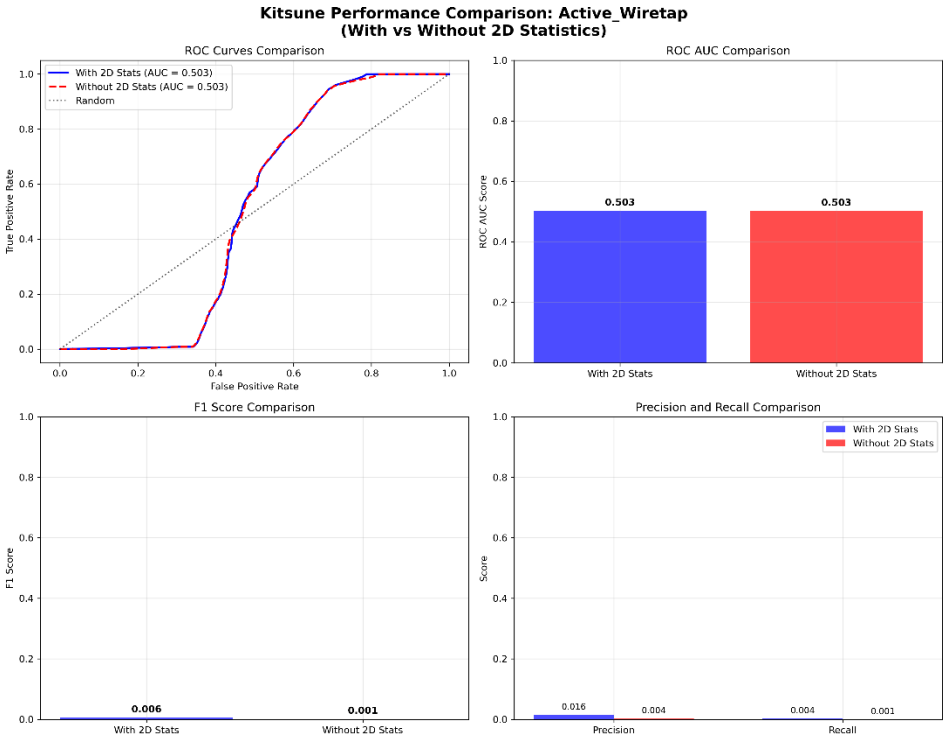


Figure 5.2: Performance Comparison With and Without 2D Features on Active_wiretap

As shown in Figure 5.2, the results are similar to those observed in the previous dataset, with negligible differences between the two configurations and overall limited detection performance.

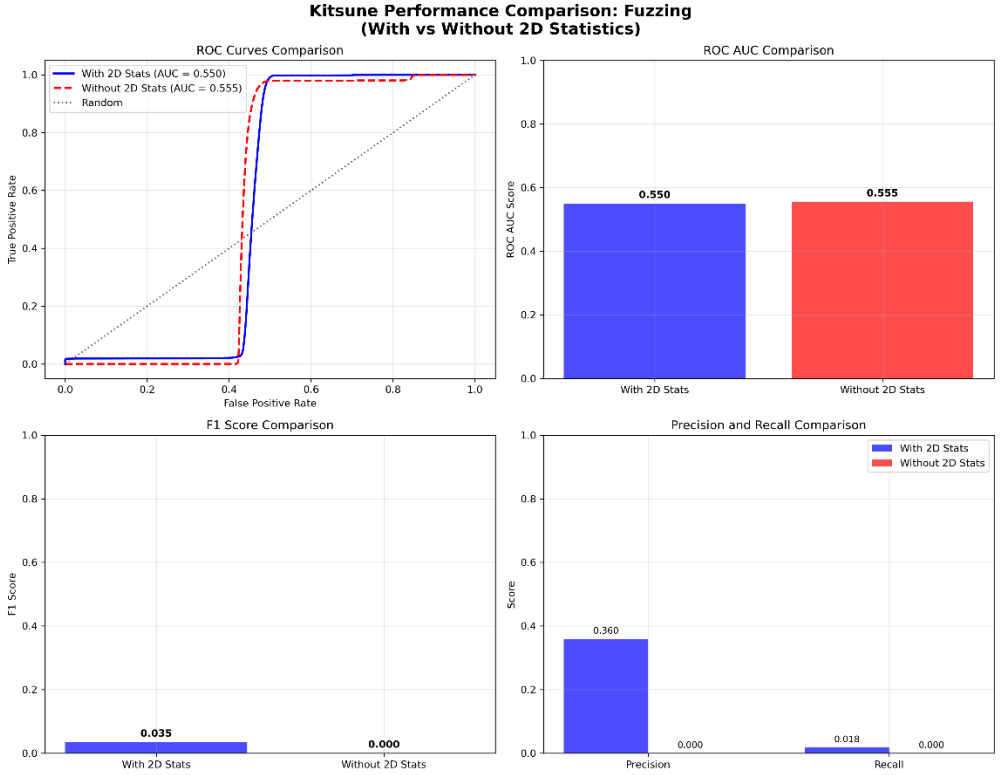


Figure 5.3: Performance Comparison With and Without 2D Features on Fuzzing

Figure 5.3 shows a similar overall trend, with only minor AUC differences; however, the inclusion of 2D features slightly improves precision and F1-score.

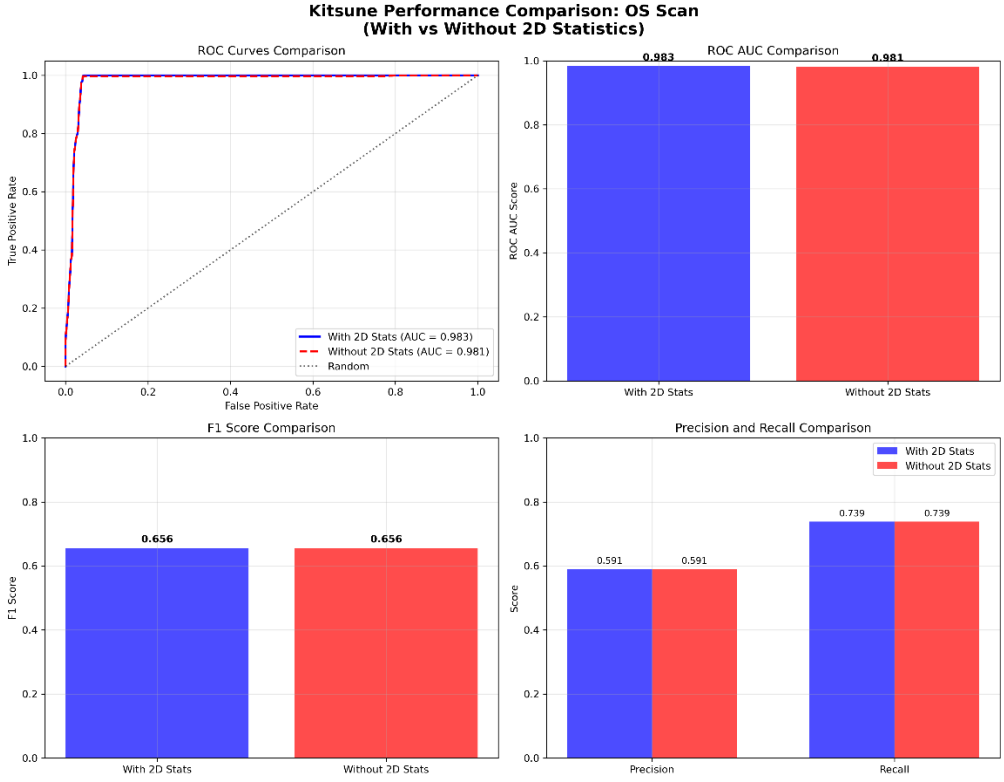


Figure 5.4: Performance Comparison With and Without 2D Features on os scan

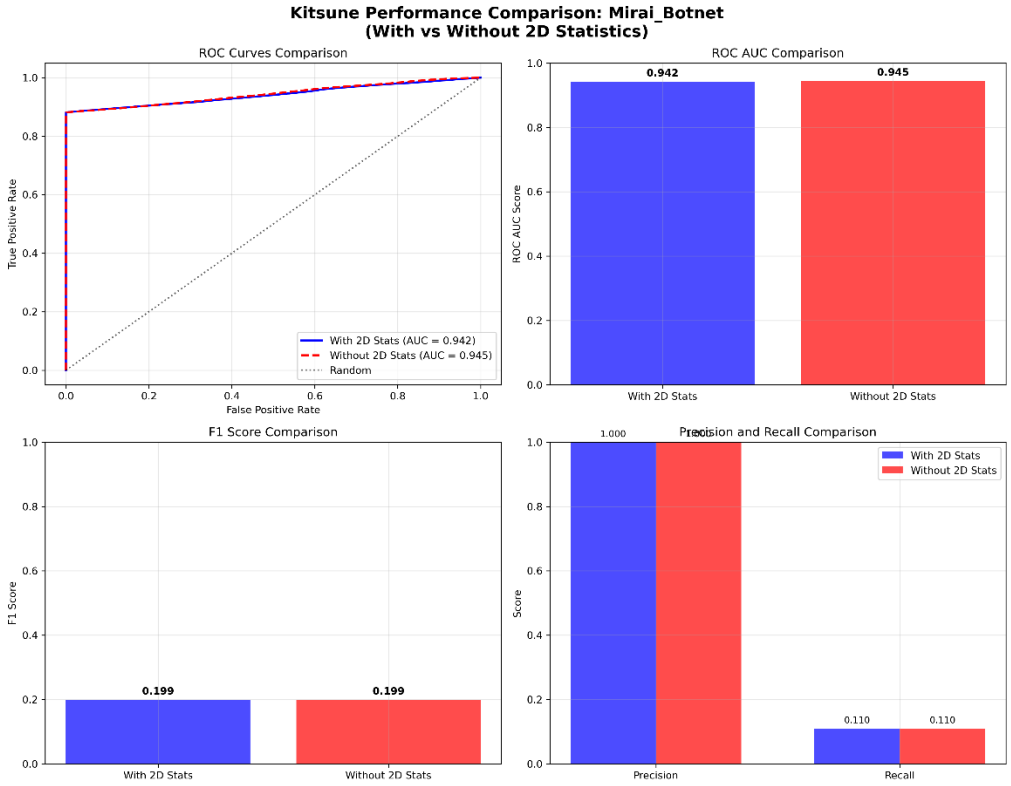


Figure 5.5: Performance Comparison With and Without 2D Features on os scan

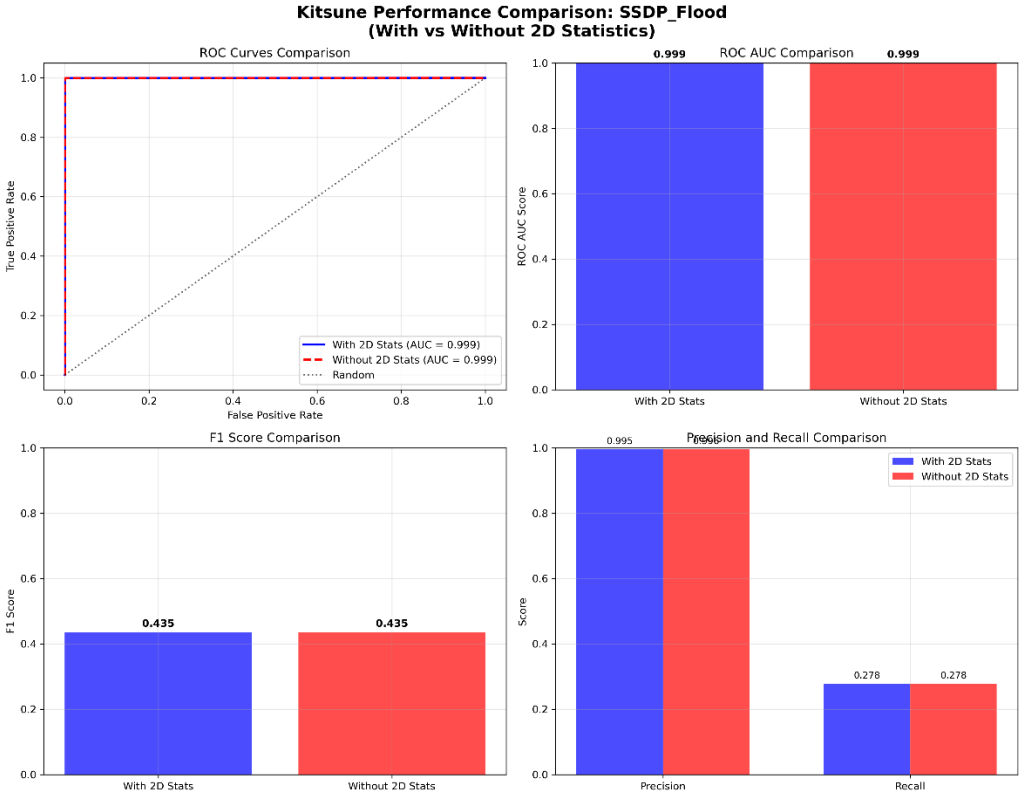


Figure 5.6: Performance Comparison With and Without 2D Statistical Features on the SSDP

Figures 5.4–5.6 exhibit similar performance trends, with only marginal differences between the configurations with and without 2D features. Overall, the inclusion of 2D statistics does not significantly affect detection performance on these datasets.

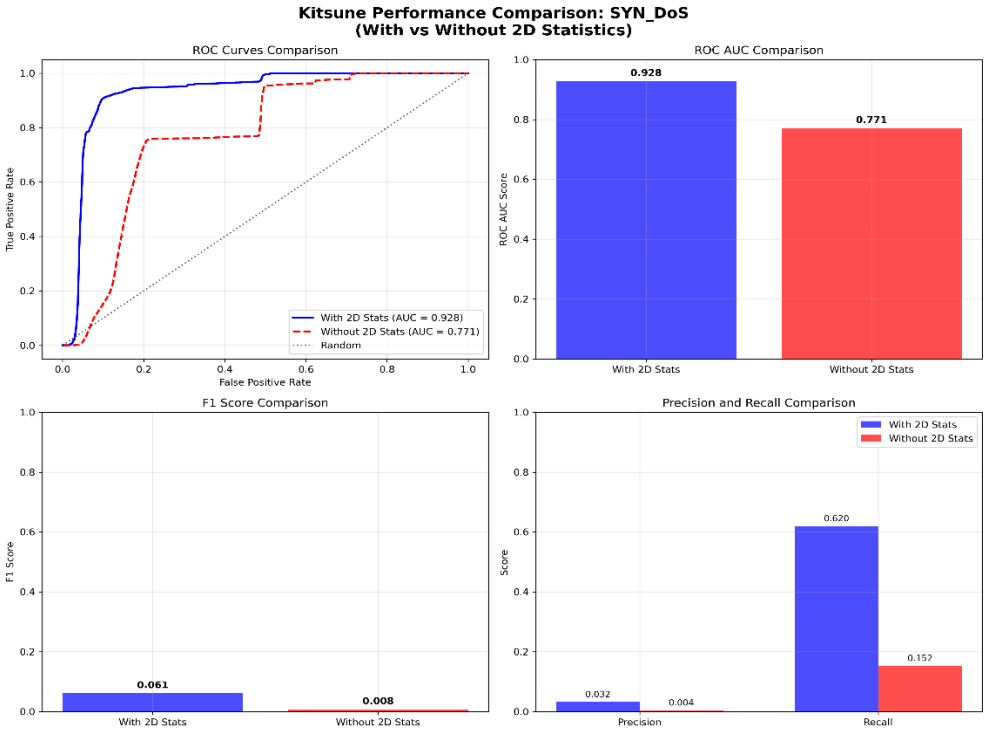


Figure 5.7: Performance Comparison With and Without 2D Statistical Features on the SYN_dos

Figure 5.7 shows a significant improvement when 2D features are included. Both AUC and recall increase substantially, indicating that 2D statistical features play an important role in detecting SYN_DoS attacks.

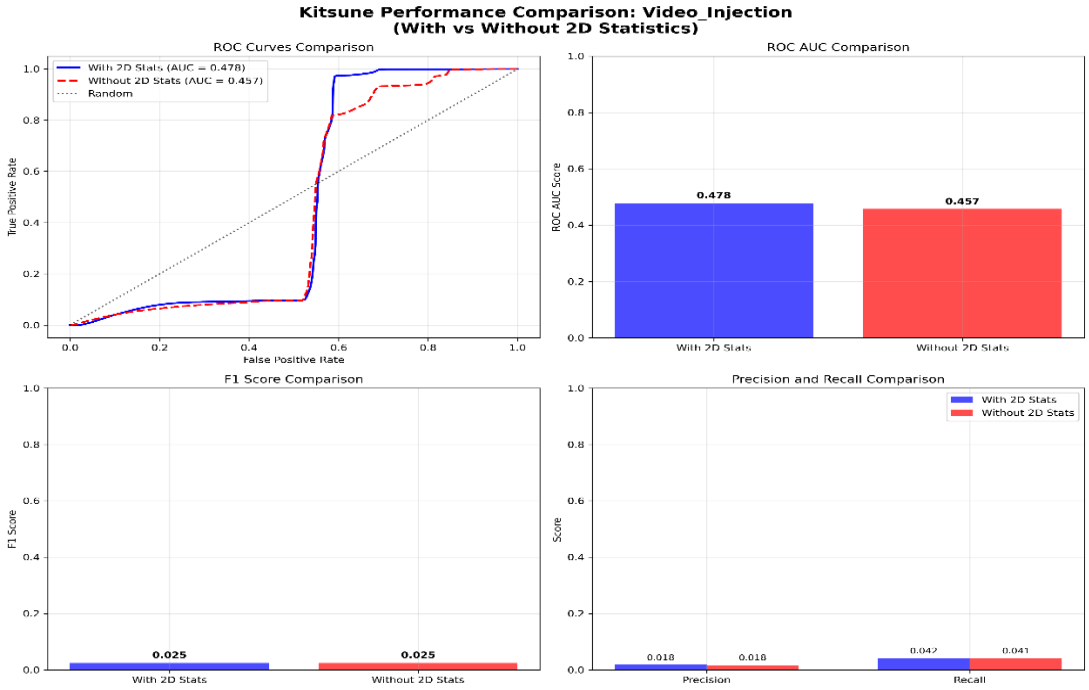


Figure 5.8: Performance Comparison With and Without 2D Statistical Features on the Video_injection

Figure 5.8 indicates only a marginal AUC gain with 2D features, with nearly identical F1 and recall values, suggesting limited practical impact.

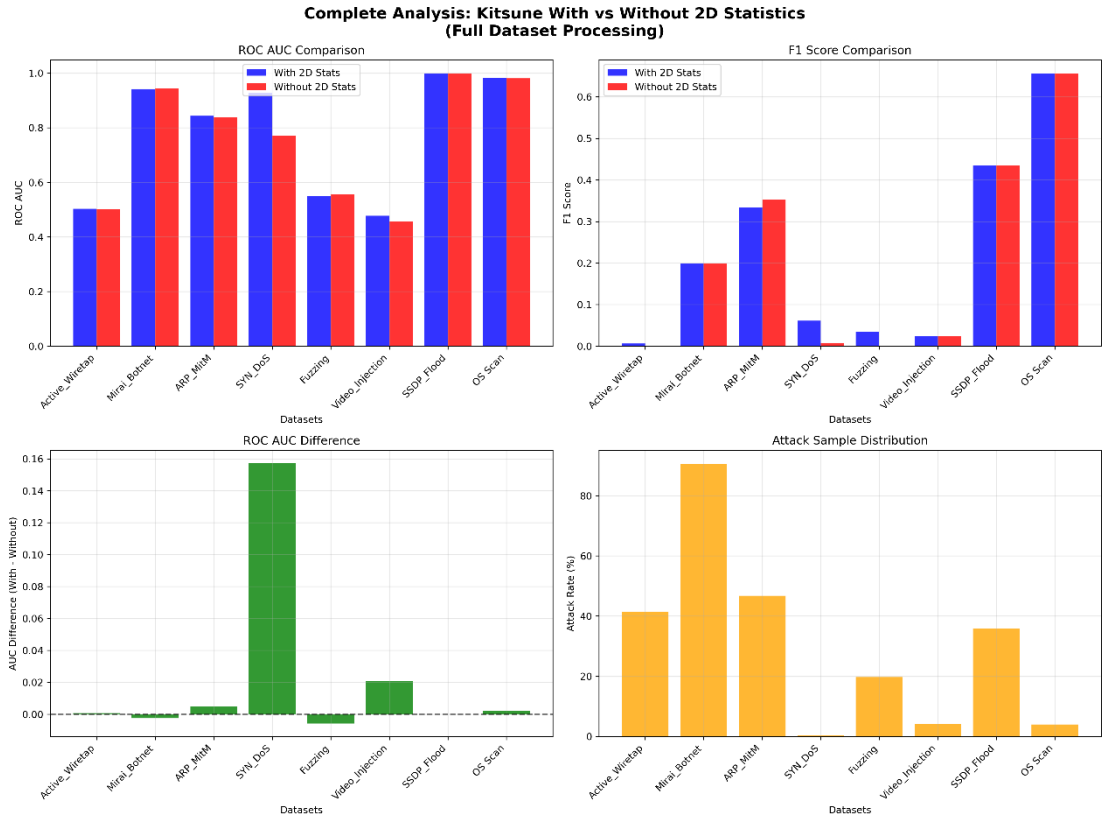


Figure 5.9: Performance comparison of Kitsune with and without 2D statistical features

5.4.2. Summary Analysis of 2D Feature Effects (All Datasets)

Figure 5.9 provides an integrated comparison of Kitsune's performance with and without 2D statistical features across all nine datasets. Overall, the results indicate that 2D temporal statistics have a selective and dataset-dependent impact on detection performance.

From the ROC AUC comparison, most datasets show marginal performance improvement or parity when 2D statistics are enabled. Notably, ARP_Mitm, Video_Injection, and SYN_Dos exhibit the most visible AUC gains, suggesting that attacks with temporal progression benefit from additional time-window context. In contrast, traffic dominated by obvious or persistent attack patterns, such as Mirai Botnet and SSDP Flood, shows a negligible difference.

The AUC plot reinforces these observations: the largest improvement appears in SYN_Dos, where the rare-event nature makes temporal context useful for stabilizing detection decisions. For most other datasets, the 2D feature contribution remains within ±0.02, indicating that the base Kitsune already captures sufficient structure in many traffic scenarios.

The F1-score comparison further highlights this nuance. Although ROC AUC improves slightly in certain cases, F1 gains are not consistent, especially in imbalanced scenarios (e.g., SYN_Dos and Fuzzing), where the model struggles to maintain recall

regardless of feature dimensionality. This confirms that class imbalance, not feature design, is the dominant limitation in rare-attack environments.

5.5. compare three threshold selection methods

In this experiment, we evaluate the system on the same nine Kitsune datasets to ensure comparability with the results from Experiment I. The goal of Experiment II is to analyze the effect of different thresholding strategies and feature configurations on the detection performance across varied traffic profiles and attack intensities.

Before Experiment II, we performed a preliminary analysis on two datasets to demonstrate how different threshold choices influence Kitsune's performance. This confirms that threshold selection plays a critical role in ensuring stable and accurate detection results.

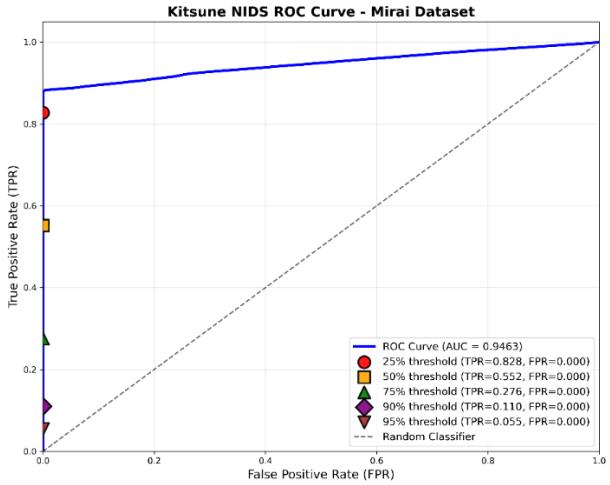


Figure 5.10: ROC curve of Kitsune on the Mirai dataset with different threshold selections

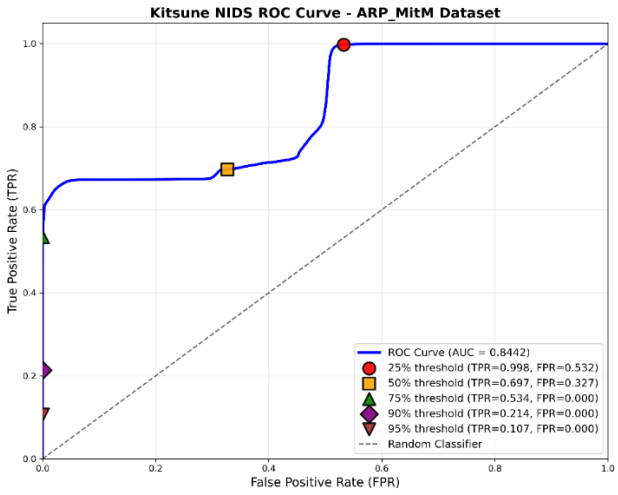


Figure 5.11: ROC curve of Kitsune on the ARP_Mitm dataset with different threshold selections.

Figures 5.10 and 5.11 demonstrate that different threshold selections lead to substantially different operating points on the ROC curve. While lower thresholds achieve higher TPR, they may significantly increase FPR, whereas stricter thresholds sharply reduce detection rates. These results highlight that threshold choice critically impacts the practical detection performance of Kitsune.

A consistent evaluation pipeline was applied across all datasets. The analysis included comparing multiple thresholding strategies, assessing parameter sensitivity, determining optimal threshold values, generating ROC curves, and examining the statistical characteristics of the anomaly scores. This unified procedure ensures fair comparison and consistent interpretation of the results.

5.5.1. Active_Wiretap

The figure 5.12 shows that the Active Wiretap attack represents a traffic interception scenario in which an adversary deliberately captures communication flows to monitor sensitive information. The dataset used in our evaluation contains 2,223,689 packets. As shown in Figure5.12, attack traffic accounts for 41.5% of the dataset, while normal traffic represents 58.5%, resulting in a moderately balanced traffic distribution suitable for evaluating intrusion detection performance.

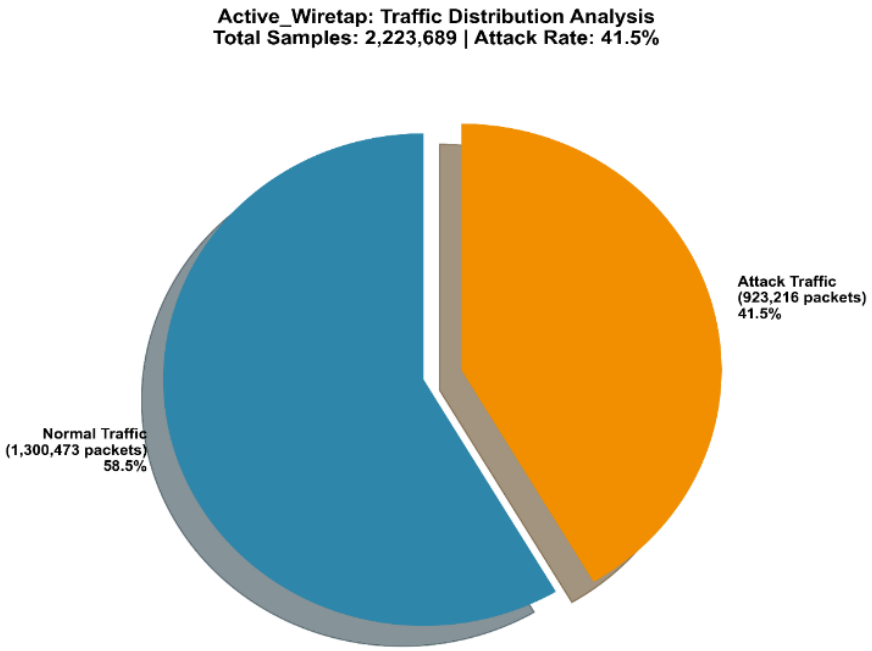


Figure 5.13: Traffic distribution in Active_Wiretap

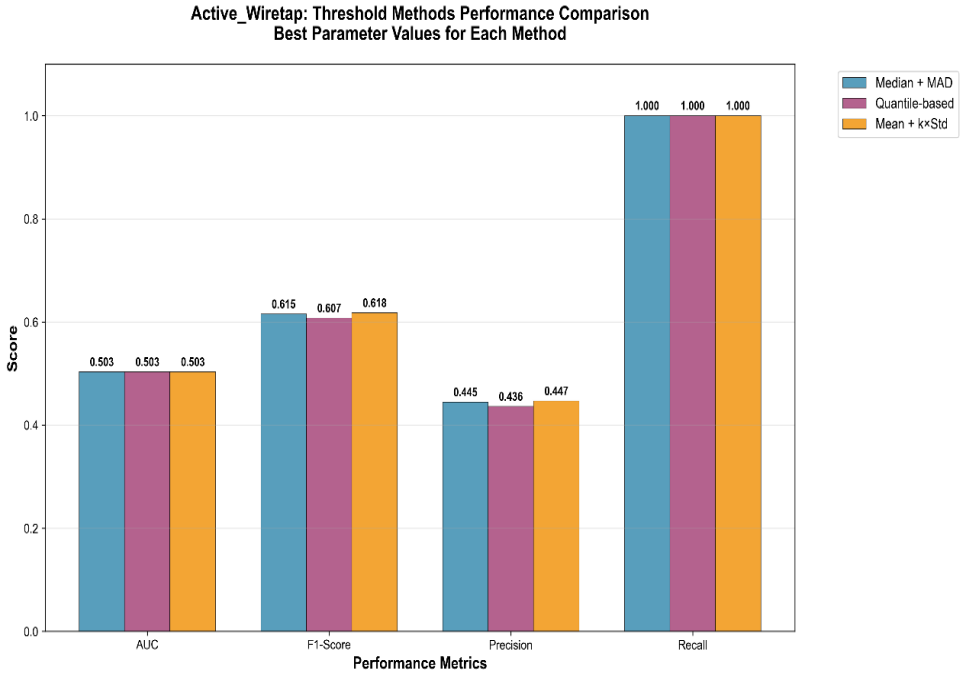


Figure 5.13: performance comparison of thresholding methods on the Active_Wiretap dataset.

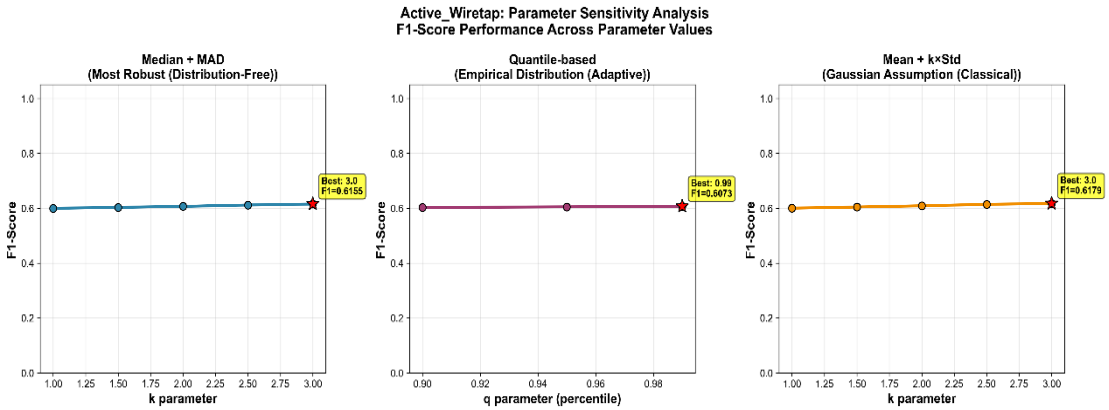


Figure 5.14: parameter sensitivity analysis in Active_Wiretap

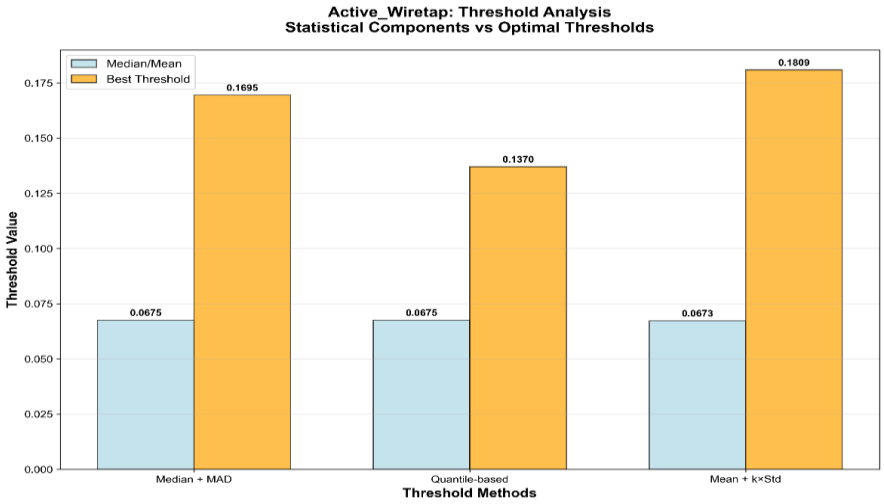


Figure 5.15: optimal thresholds on the Active_Wiretap dataset.

As shown in **Figures 5.13–5.15**, the three statistical thresholding strategies achieve nearly identical performance on the Active_Wiretap dataset. The traffic distribution is moderately balanced (41.5% attack vs. 58.5% normal), enabling fair comparison. All methods produce similar AUC values (≈0.503), and parameter sensitivity analysis indicates stable performance across different settings. The ROC results suggest performance close to random classification, indicating limited separability between normal and attack traffic. Overall, threshold selection alone has limited impact in this scenario.

5.5.2. ARP_Mitm

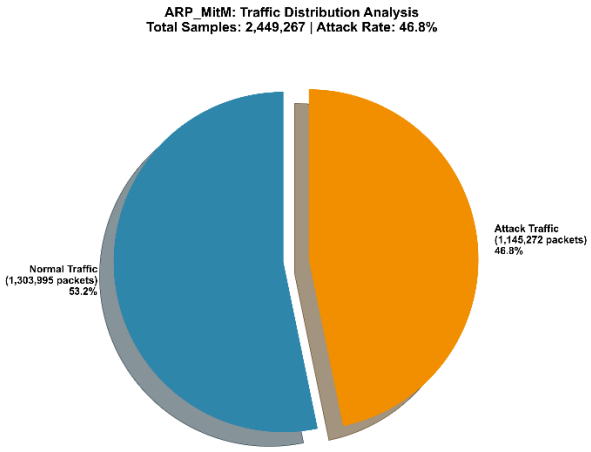


Figure 5.16: Traffic distribution in ARP_Mitm

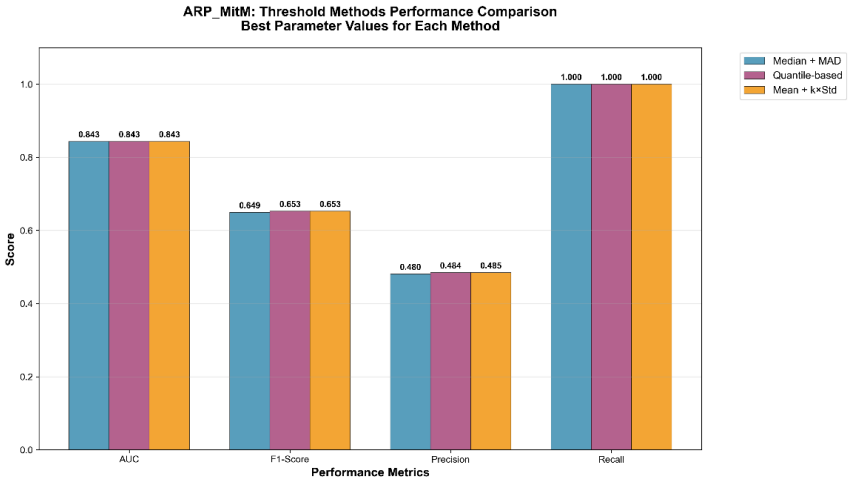


Figure 5.17: performance comparison of thresholding methods on the ARP_Mitm dataset.

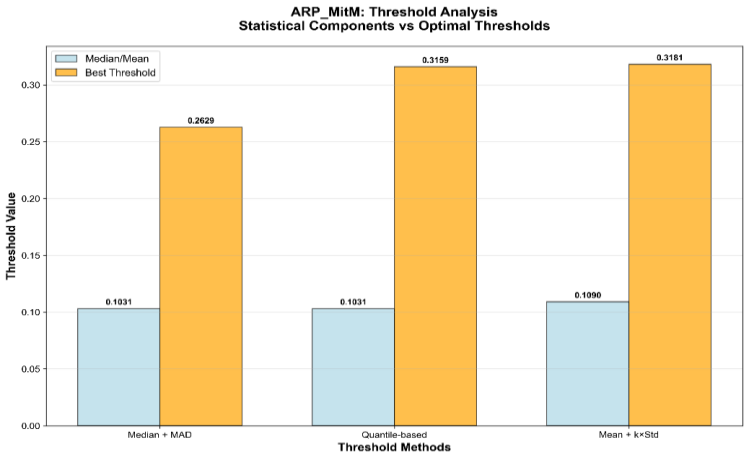


Figure 5.18: optimal thresholds on the ARP_Mitm dataset.

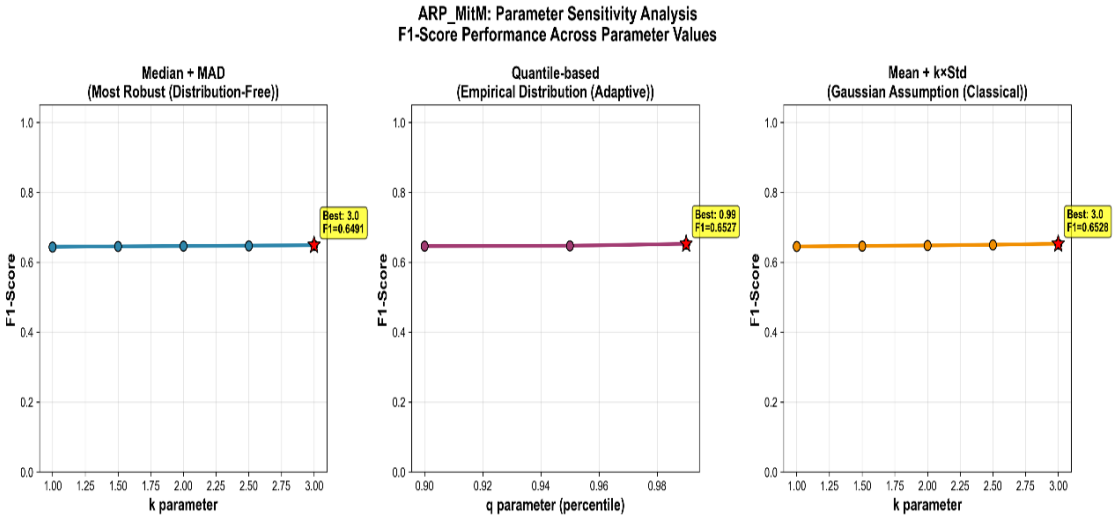


Figure 5.19: parameter sensitivity analysis in Active_Wiretap

0n the **ARP_Mitm** dataset, the three thresholding methods exhibit very similar detection performance, with only minor differences in AUC, F1-score, and precision. All approaches achieve near-perfect recall, indicating strong separability between normal and attack traffic. Parameter sensitivity analysis shows stable behavior across different parameter values, suggesting that performance is not highly dependent on fine-grained parameter tuning.

As illustrated in the threshold analysis, the optimal thresholds are consistently higher than their corresponding baseline statistical values, indicating that direct use of central tendency measures (e.g., median or mean) would be insufficient. Instead, calibrated thresholds are necessary to achieve balanced detection performance.

Since the **Fuzzing** dataset demonstrates similar anomaly score characteristics and performance patterns, the same observations apply, and further repetition of the analysis is omitted.

5.5.3. Mirai_Botnet

Next, we evaluate the methods on the *mirai_Bontnet* dataset, as figure5.16 which features an extremely high attack ratio. This scenario tests the thresholding strategies under conditions of heavy contamination by anomalous traffic.

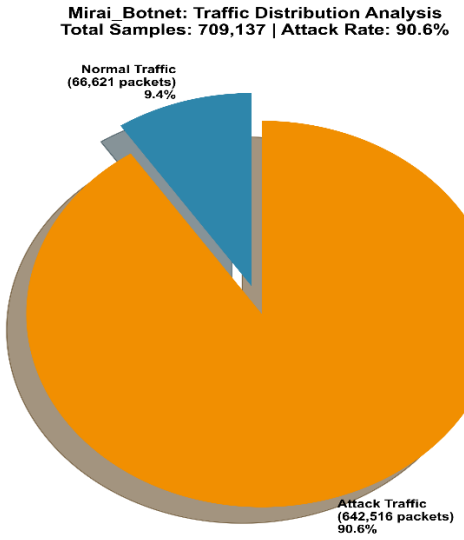


Figure 5.20: Traffic distribution in Mirai_Botent

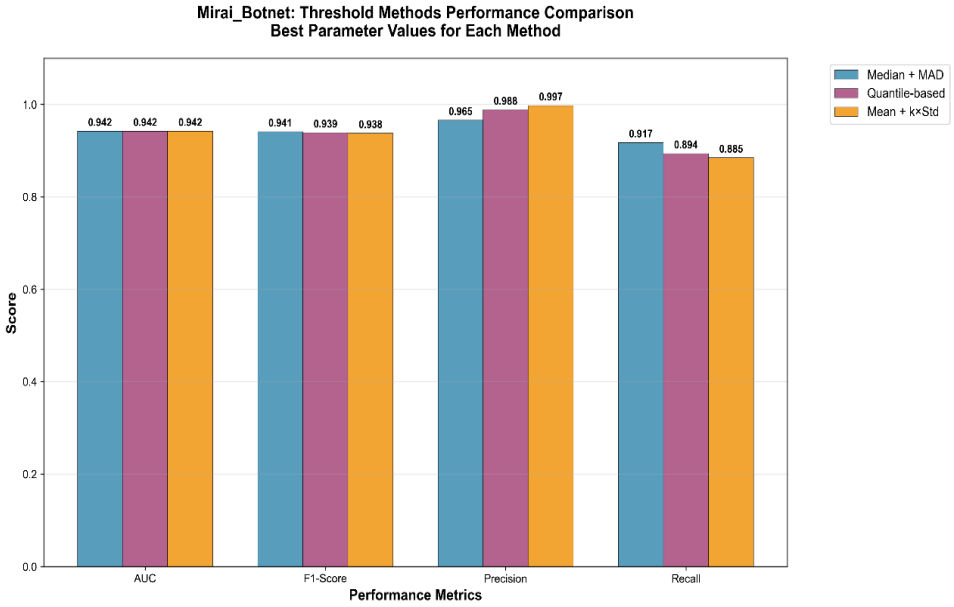


Figure 5.21: performance comparison of thresholding methods on the Mirai_Botent dataset.

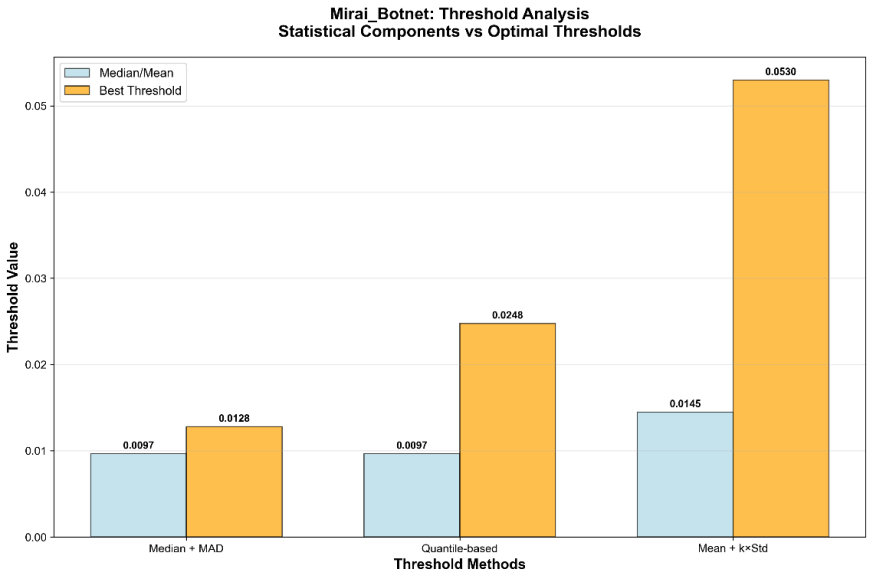
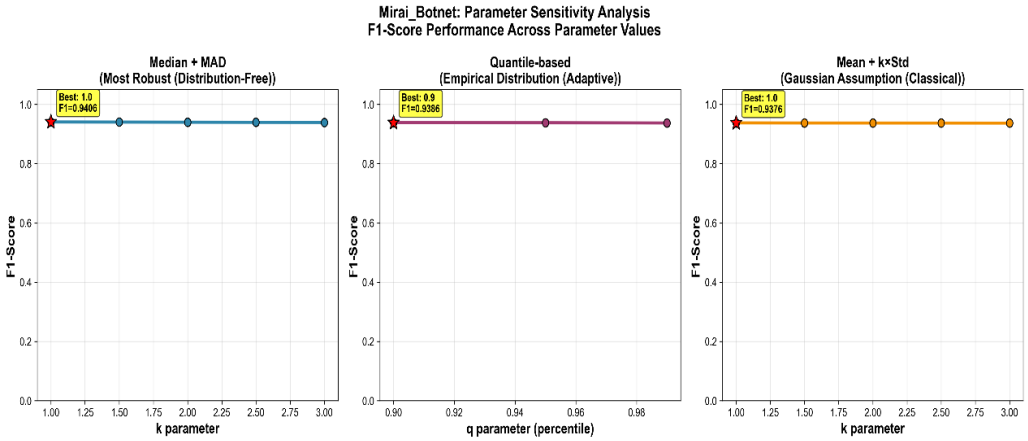


Figure 5.22: optimal thresholds on the Mirai_Botent dataset.



(d)

Figure 5.23: parameter sensitivity analysis in Mirai_Botent

As shown in **Figure 5.17**, the Mirai_Botnet dataset is heavily dominated by attack traffic, with the vast majority of packets labeled as malicious. Under this high-contamination scenario, Kitsune achieves consistently strong detection performance across all thresholding methods, as illustrated in **Figure 5.18**, where AUC and F1-scores remain above 0.93 with minimal variation.

The threshold analysis in **Figure 5.19** indicates that optimal thresholds are higher than their corresponding baseline statistics, confirming the need for calibration. However, the parameter sensitivity results in **Figure 5.20** demonstrate that performance remains highly stable across different parameter values. Overall, in such a highly separable and attack-dominant setting, threshold selection has only a limited impact on detection performance compared to mixed or low-attack scenarios.

5.5.4. OS_Scan

We then apply the analysis to the *OS_Scan* dataset, in which malicious traffic represents only about 4% of the data. This allows us to assess performance under a highly imbalanced, low-attack-rate setting.

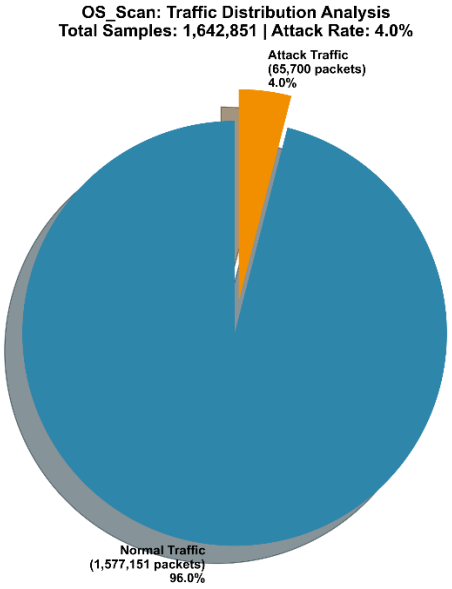


Figure 5.24: Traffic distribution in OS_Scan

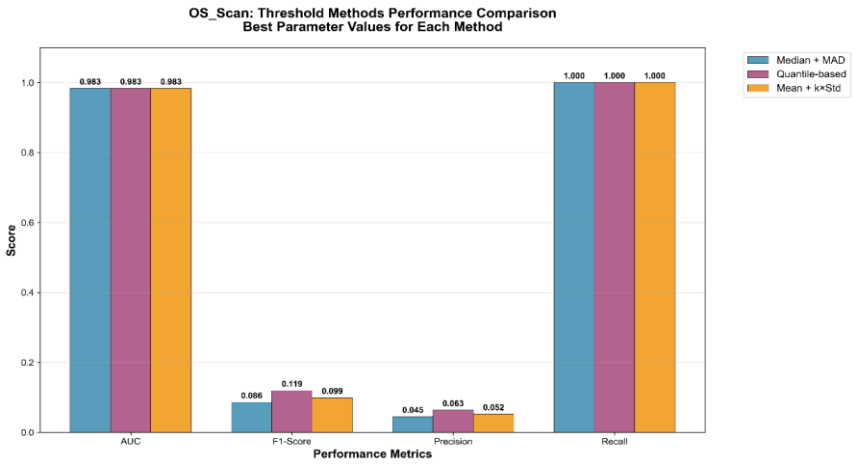


Figure 5.25: performance comparison of thresholding methods on the OS_Scan dataset.

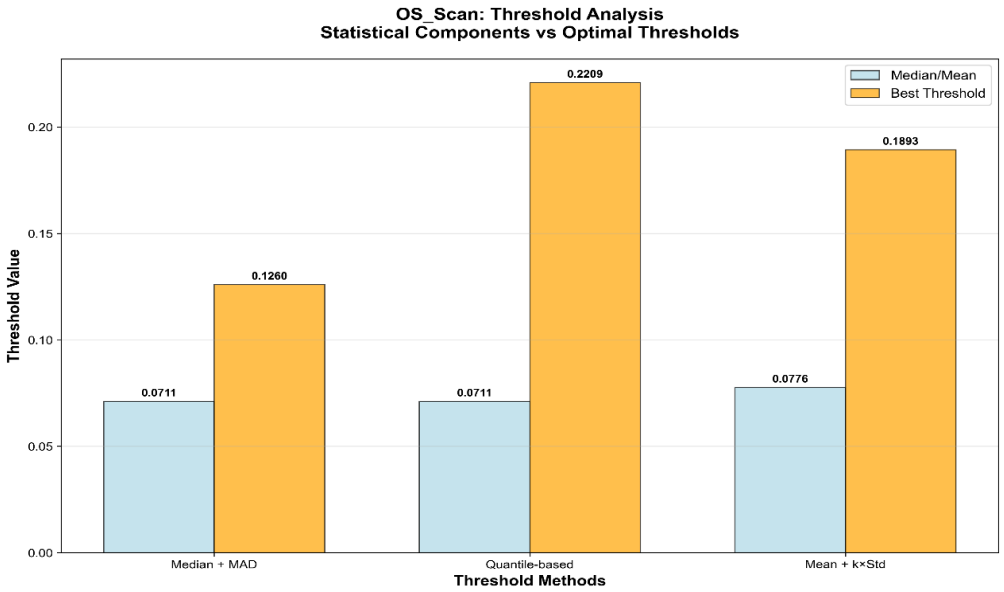


Figure 5.26: optimal thresholds on the OS_Scan dataset.

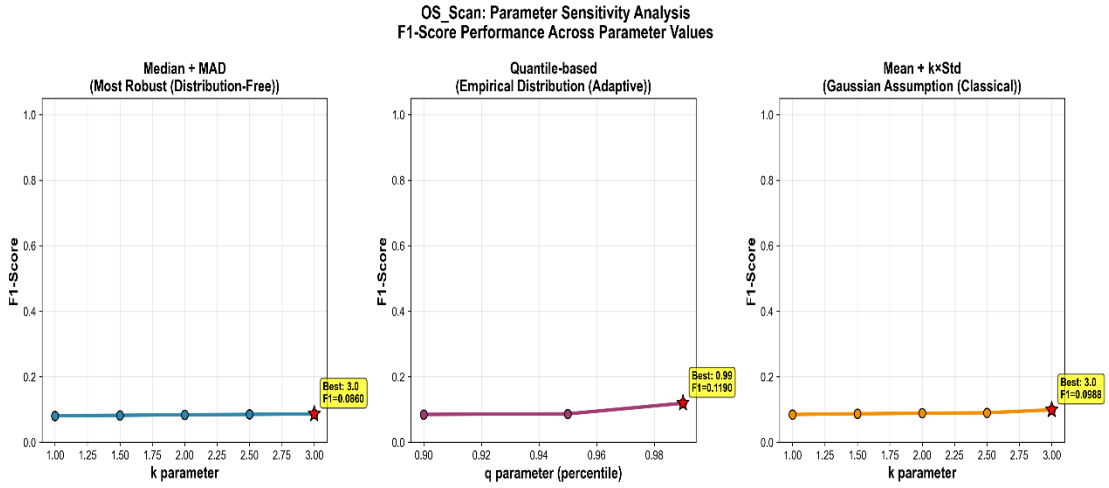


Figure 5.27: parameter sensitivity analysis in OS_Scan

As shown in **Figure 5.21**, the OS_Scan dataset is highly imbalanced, with attacks representing only about 4% of total traffic. Despite this imbalance, all thresholding methods achieve near-perfect recall, as illustrated in **Figure 5.22**. However, precision and F1-score remain very low, reflecting a large number of false positives caused by the rare-anomaly nature of the dataset.

The threshold analysis in **Figure 5.23** indicates that the optimal thresholds are substantially higher than the corresponding baseline statistical values, confirming that direct use of the median or mean is insufficient in this scenario. The parameter sensitivity results in **Figure 5.24** show limited variation across parameter settings, although the quantile-based method achieves slightly better F1 performance.

Overall, in highly imbalanced conditions where attacks are scarce, threshold calibration becomes important, yet the system remains prone to false alarms despite maintaining high recall.

5.5.5. SSDP_Flood

Next, we analyze the *SSDP_Flood* dataset, in which roughly 35% of the packets are malicious. This dataset represents a medium-attack-ratio

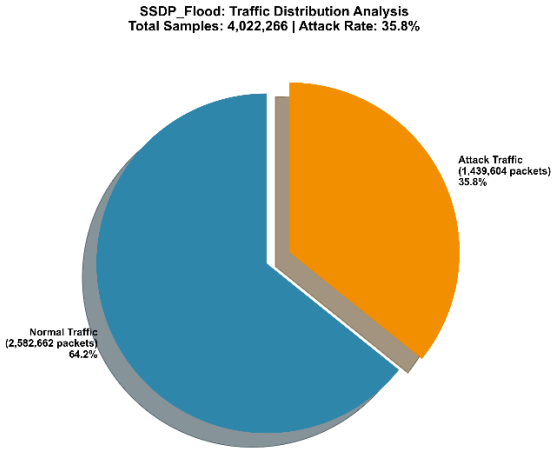


Figure 5.28: Traffic distribution in SSDP_Flood

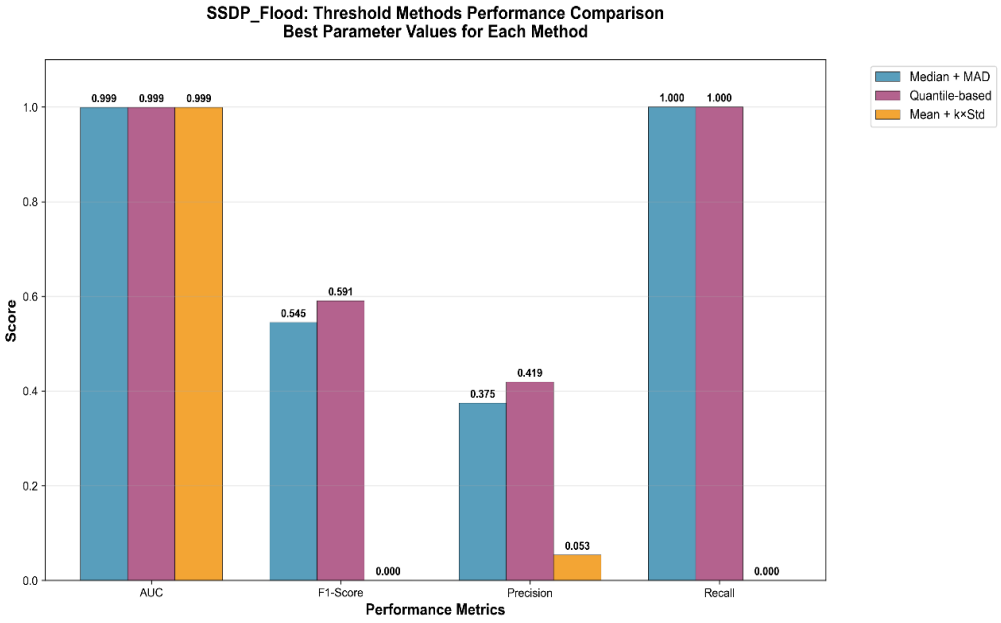


Figure 5.29: performance comparison of thresholding methods on the SSDP_Flood dataset

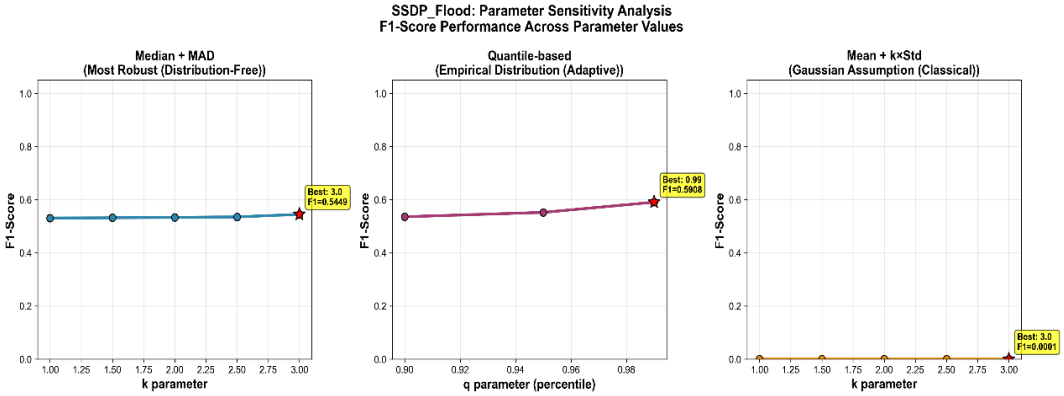


Figure 5.30: parameter sensitivity analysis in SDP_Flood

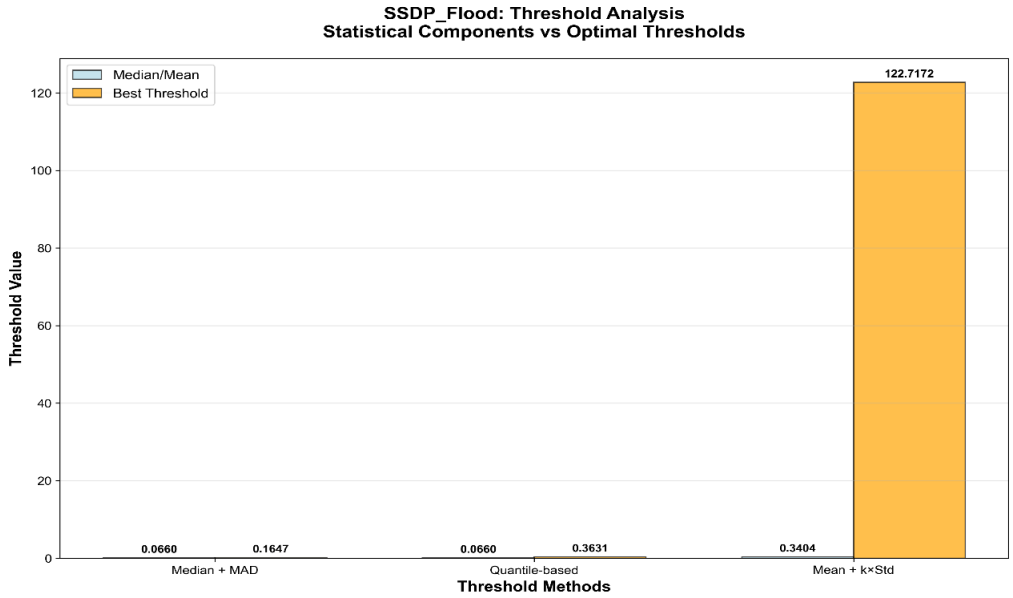


Figure 5.31: optimal thresholds on the SSDP_Flood dataset.

For the SSDP_Flood dataset, where attack traffic accounts for approximately 35% of total packets (Figure 5.28), the three thresholding strategies demonstrate markedly different behaviors. Although all methods achieve near-perfect AUC values (≈1.000), substantial differences emerge in classification performance. The quantile-based method achieves the best overall balance, with the highest F1-score (≈0.59) and precision (≈0.42), followed by the median + MAD approach with moderate performance (F1 ≈ 0.50). In contrast, the mean + k×std method collapses entirely, yielding an F1-score close to zero due to extremely poor precision.

The parameter sensitivity analysis (Figure 5.30) shows that both the robust statistical method and the quantile-based approach remain relatively stable across parameter

variations. However, the Gaussian-assumption-based method exhibits instability and strong sensitivity to parameter choices, resulting in severe performance degradation.

Furthermore, the threshold comparison in Figure 5.31 reveals that the optimal threshold selected by mean + k×std is excessively large compared to the corresponding statistical baseline components. This phenomenon is attributed to the heavy-tailed distribution of anomaly scores, which inflates the mean and standard deviation, ultimately causing mean-based thresholding to fail.

Overall, when anomalies constitute a substantial but not dominant proportion of traffic (≈35%), empirical and robust thresholding strategies remain reliable, whereas distribution-assumption-based approaches may break down dramatically under heavy-tailed score distributions.

5.5.6. SSL_Renegotiation

Next, we conduct experiments on the *SSL_Renegotiation* dataset, in which attack traffic is extremely scarce. This allows us to assess performance under a highly imbalanced, rare-attack scenario.

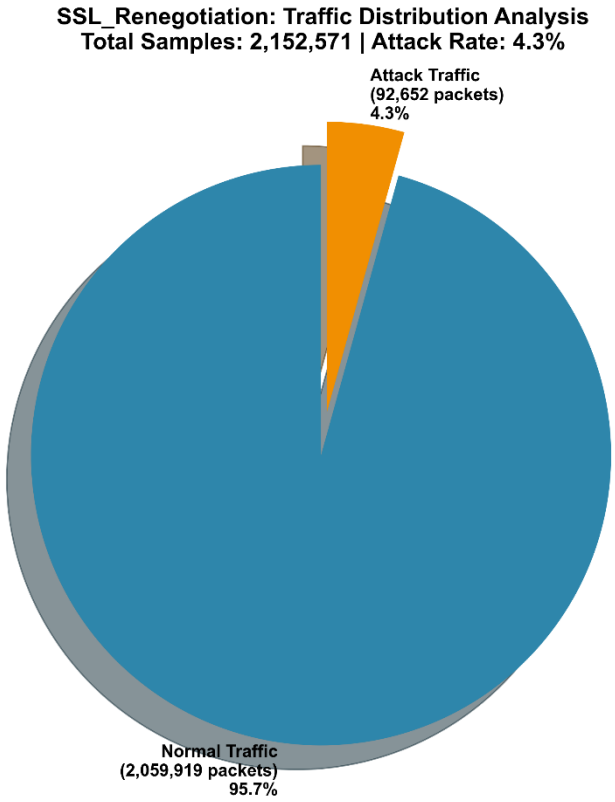


Figure 5.32: Traffic distribution in SSL_Renegotiation

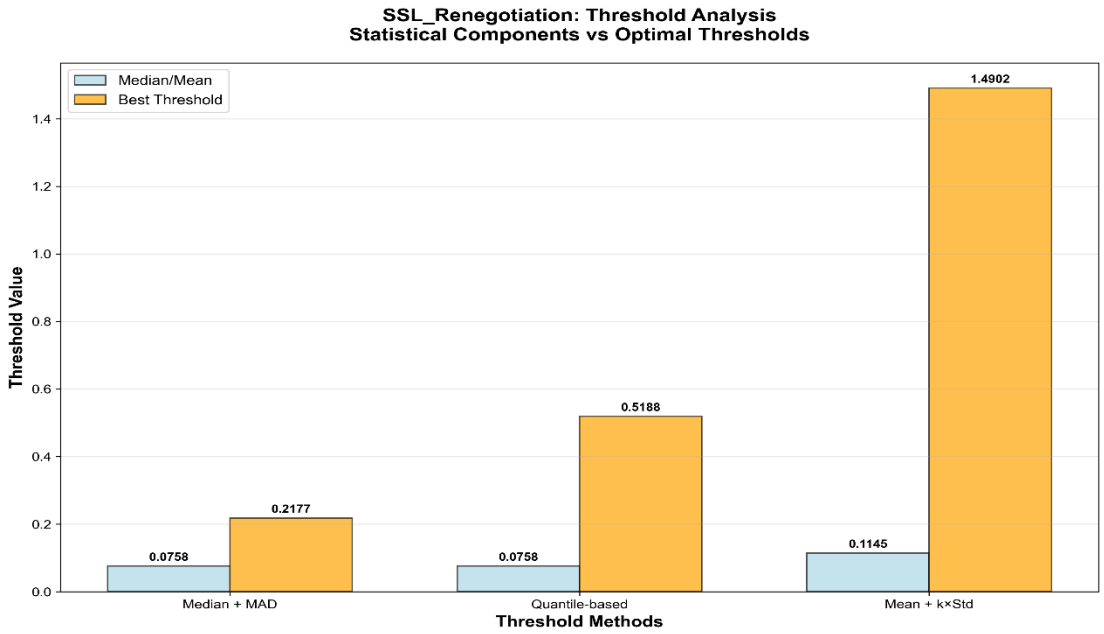


Figure 5.33: optimal thresholds on the SSL_Renegotiation dataset.

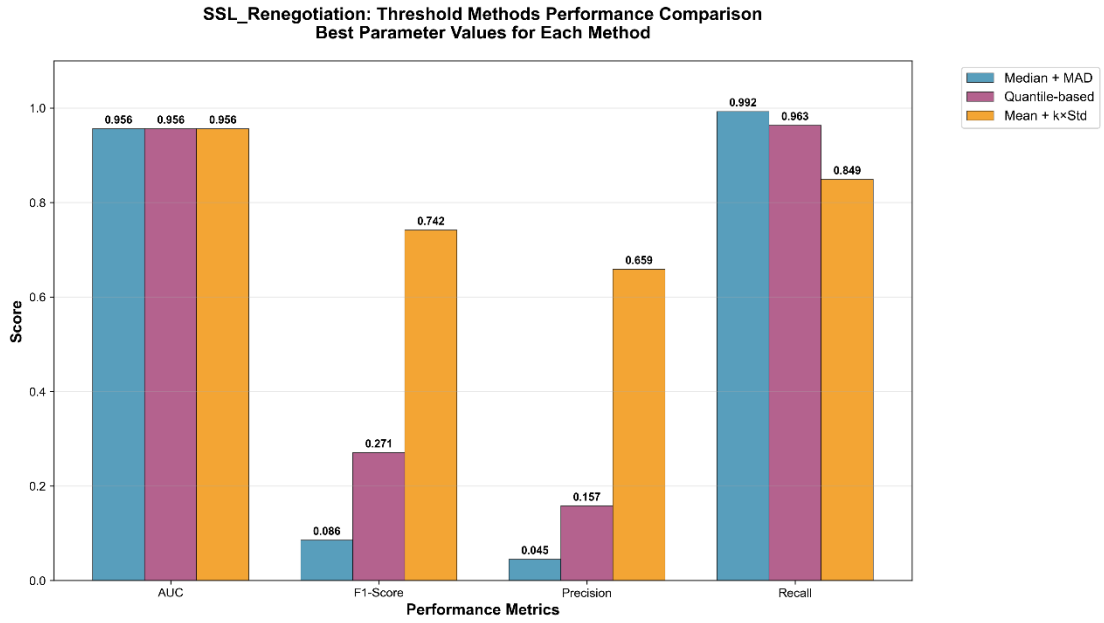
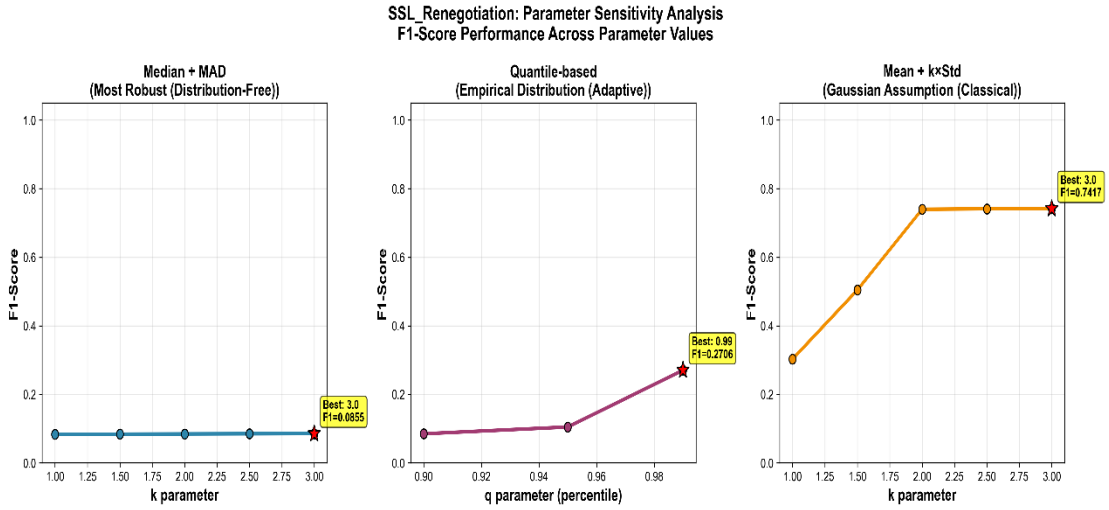


Figure 5.34: performance comparison of thresholding methods on the SSL_Renegotiation dataset



(d)

Figure 5.35: parameter sensitivity analysis in SDP_Flood

As shown in **Figure 5.29**, the SSL_Renegotiation dataset represents an extremely rare-attack scenario, with attacks accounting for only a small fraction of total traffic. In this setting, the performance comparison in **Figure 5.31** indicates that robust distribution-free approaches such as Median + MAD yield very low F1-scores, while the mean + k·std method achieves substantially better detection performance.

The threshold analysis in **Figure 5.30** shows that the optimal threshold for mean + k·std is significantly higher than its baseline statistical value, reflecting strong sensitivity to tail deviations. The parameter sensitivity results in **Figure 5.32** further demonstrate that increasing the k parameter improves F1 performance, suggesting that tail-aware calibration is beneficial in sparse anomaly conditions.

Overall, in extremely imbalanced scenarios, robustness may become overly conservative, whereas tail-sensitive thresholding strategies are more effective in capturing subtle distribution shifts introduced by rare attacks.

5.5.7. SYN_DoS:

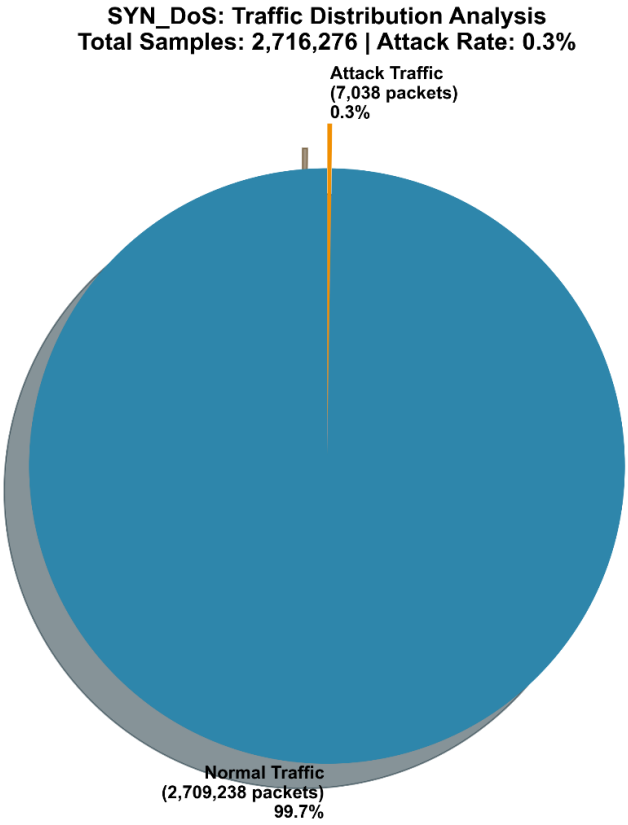


Figure 5.36: Traffic distribution in SYN_Dos

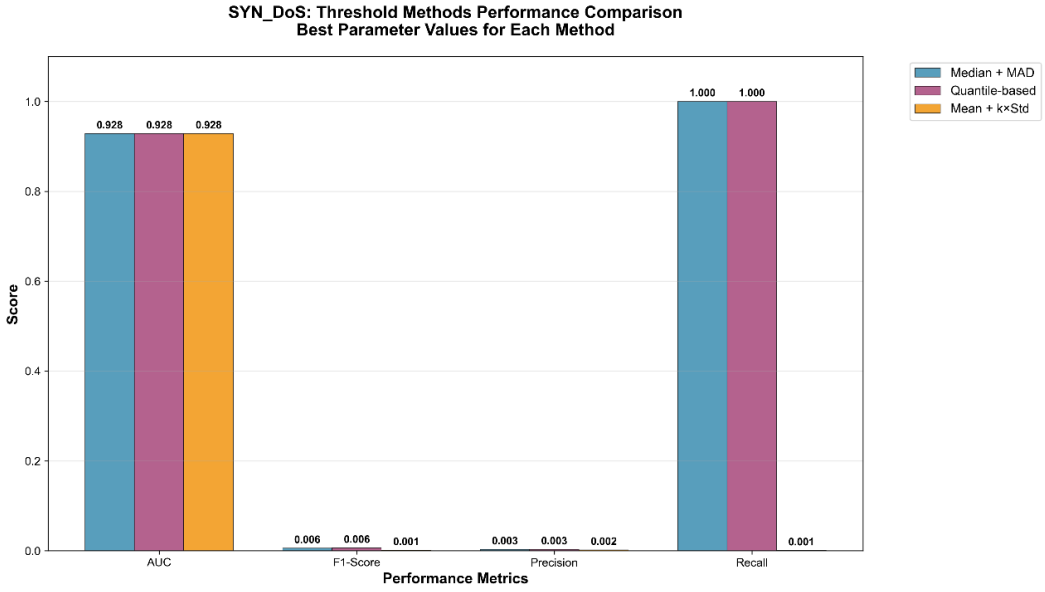


Figure 5.37: performance comparison of thresholding methods on the SYN_Dos dataset

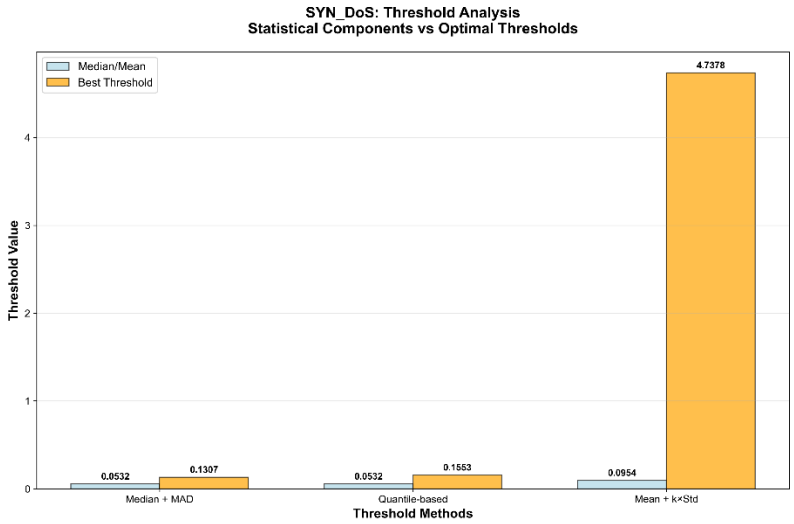


Figure 5.38: optimal thresholds on the SYN_Dos dataset

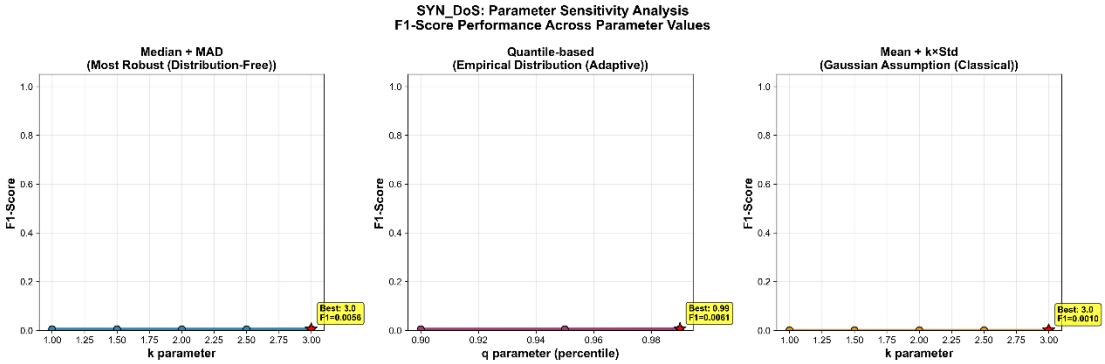


Figure 5.39: parameter sensitivity analysis in SDP_Flood

The SYN_DoS dataset is highly imbalanced, with only 0.3% attack traffic among 2,716,276 packets. Although all thresholding methods achieve high AUC values (≈0.928), indicating good separability between normal and attack traffic, the F1-score and precision remain extremely low due to the severe class imbalance. Parameter sensitivity analysis shows stable performance across different parameter values, suggesting limited influence of threshold tuning. Overall, these results indicate that classification performance in the SYN_DoS dataset is primarily constrained by data imbalance , rather than threshold selection

5.5.8. Video_injection

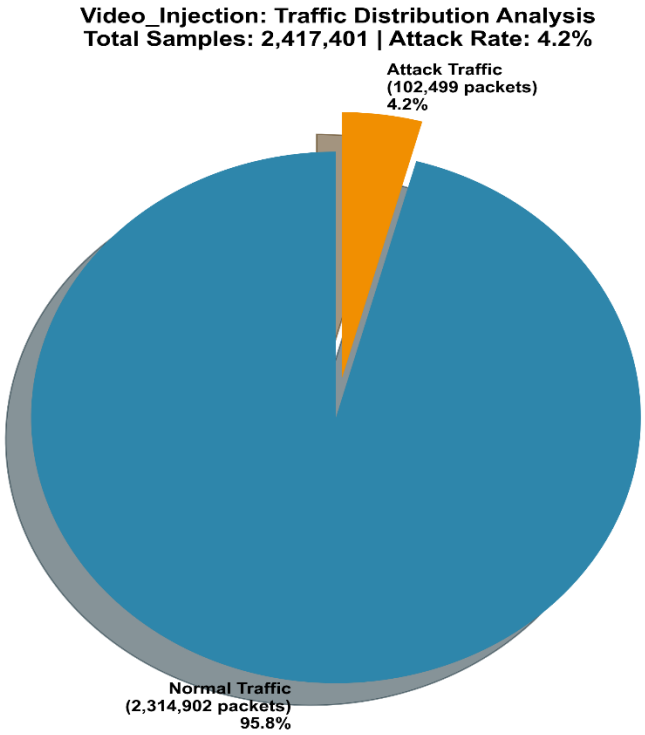


Figure 5.40: Traffic distribution in Video_injection

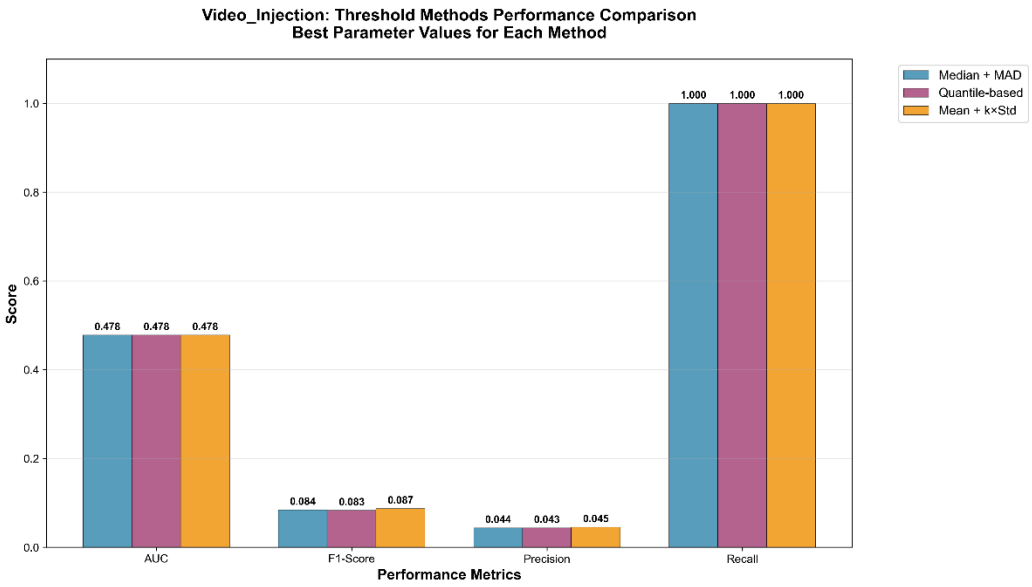


Figure 5.41: performance comparison of thresholding methods on the Video_injection dataset

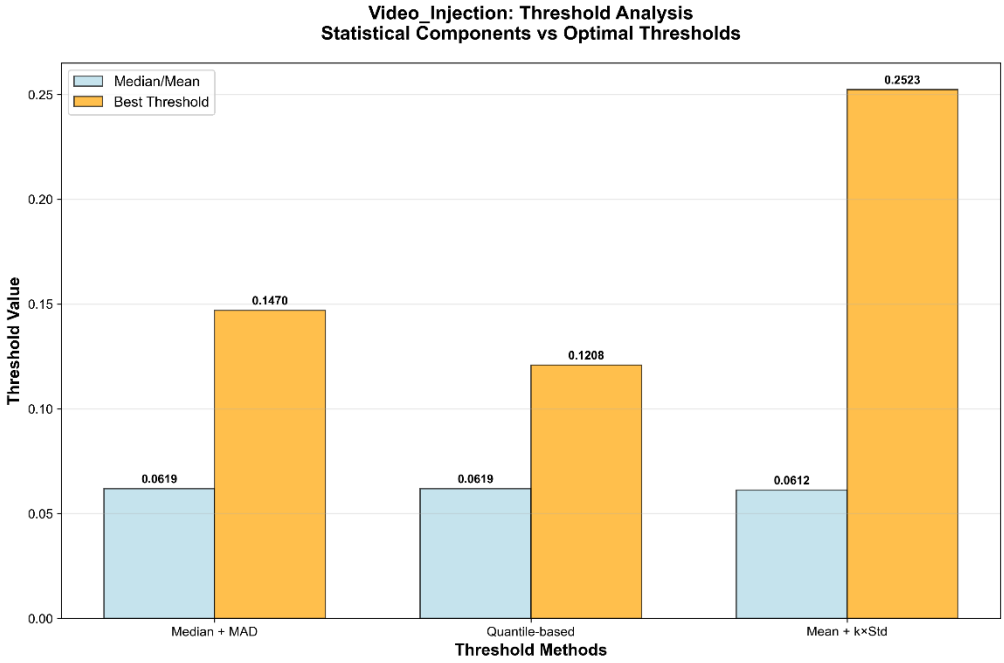


Figure 5.42: optimal thresholds on the Video_injection dataset.

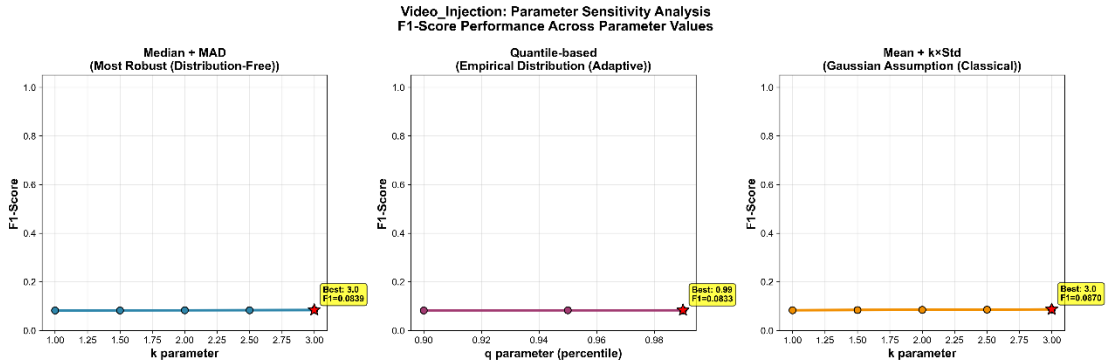


Figure 5.43: parameter sensitivity analysis in Video_injection

Similar behavior is observed in the SYN_DoS dataset; therefore, we omit a detailed discussion here.

5.5.9. comparison of the three thresholding strategies

To evaluate method robustness under varying anomaly frequencies, Figure X plots performance metrics against attack rate. This analysis reveals how each thresholding strategy scales from rare-attack to high-attack environments and highlights their stability and sensitivity to class imbalance.

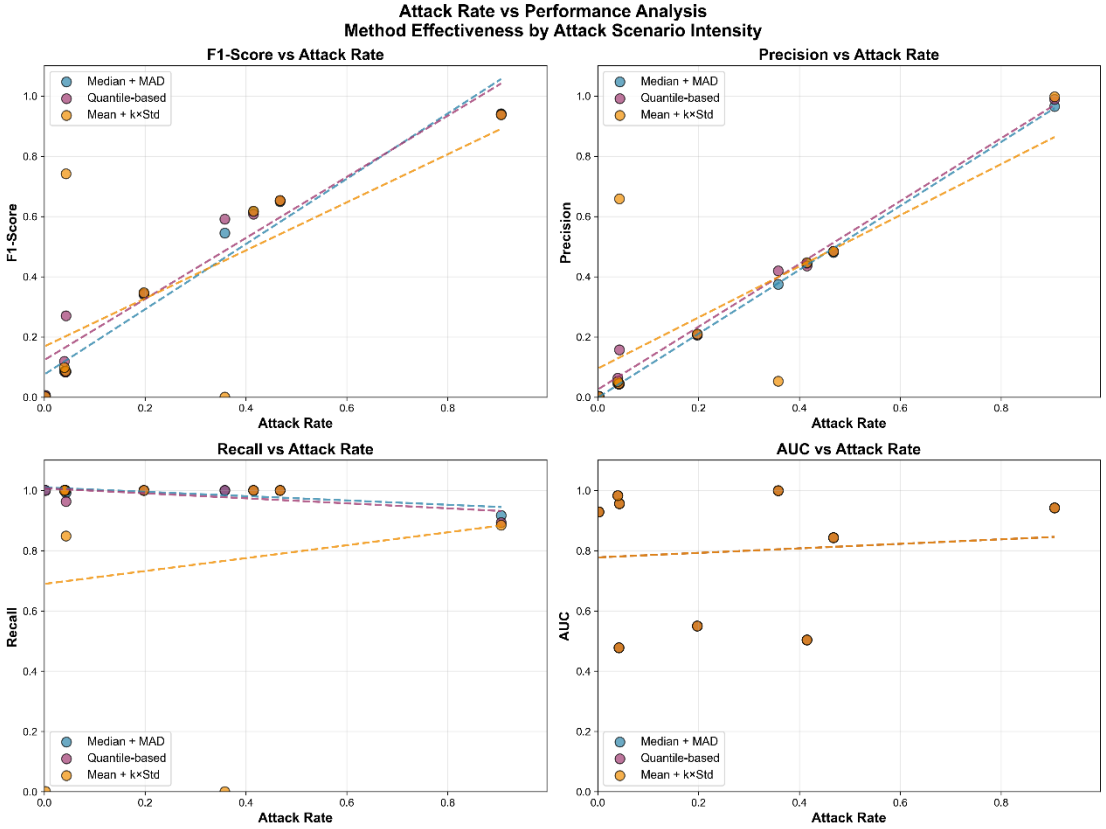


Figure 5.44: Performance comparison of threshold selection methods under different attack rates.

As the attack rate increases, all methods generally improve their F1-score and precision, reflecting reduced class imbalance difficulty. Median+MAD and quantile-based strategies maintain consistently high recall, while Mean + kσ is highly sensitive in low-attack settings but scales as attacks become more frequent. Overall, the quantile-based method shows the most stable behavior across varying attack intensities.

To assess reliability beyond average accuracy, Figure 5.45 presents a robustness analysis of the three thresholding methods. By examining F1-score dispersion, stability, and cross-method consistency, we evaluate how sensitive each approach is to different traffic scenarios and anomaly rates.

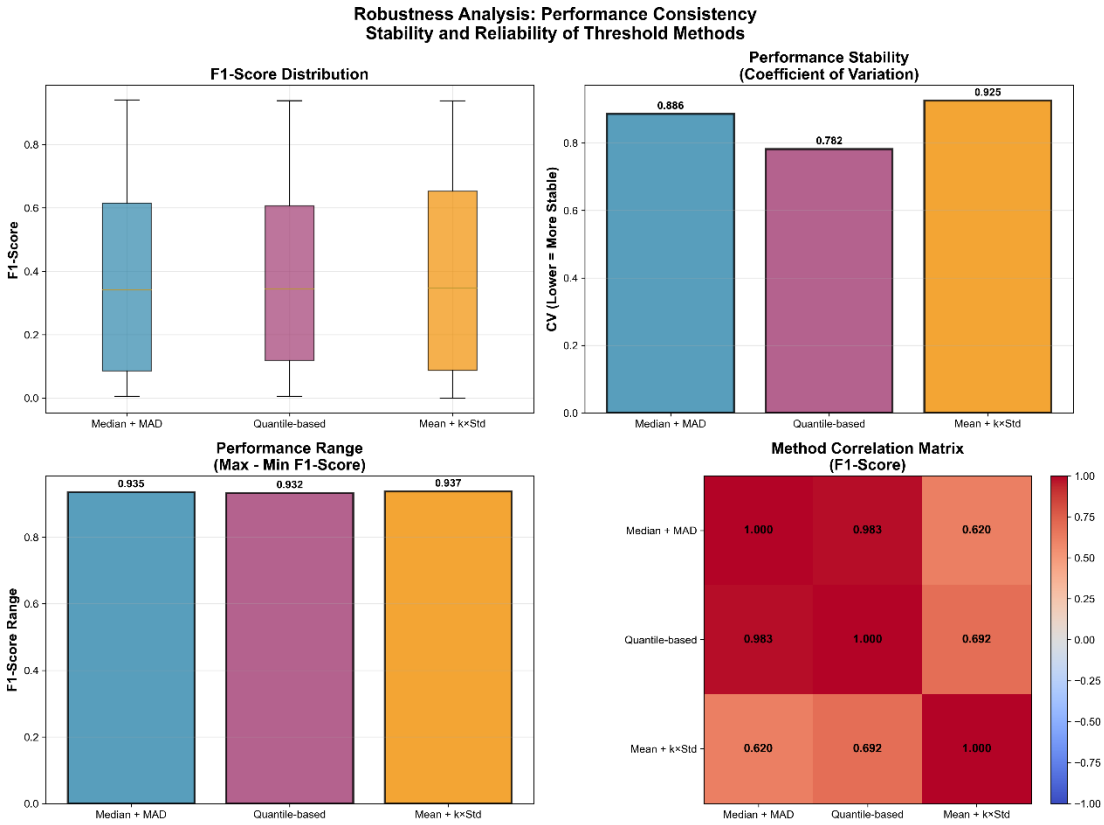


Figure 5.45: Robustness analysis of threshold selection methods.

This figure 5.45 shows that the quantile-based method is the most stable across datasets, median+MAD behaves similarly but slightly less consistently, and mean+std is the most unstable and sensitive to dataset characteristics. Overall, quantile-based thresholding achieves the best robustness.

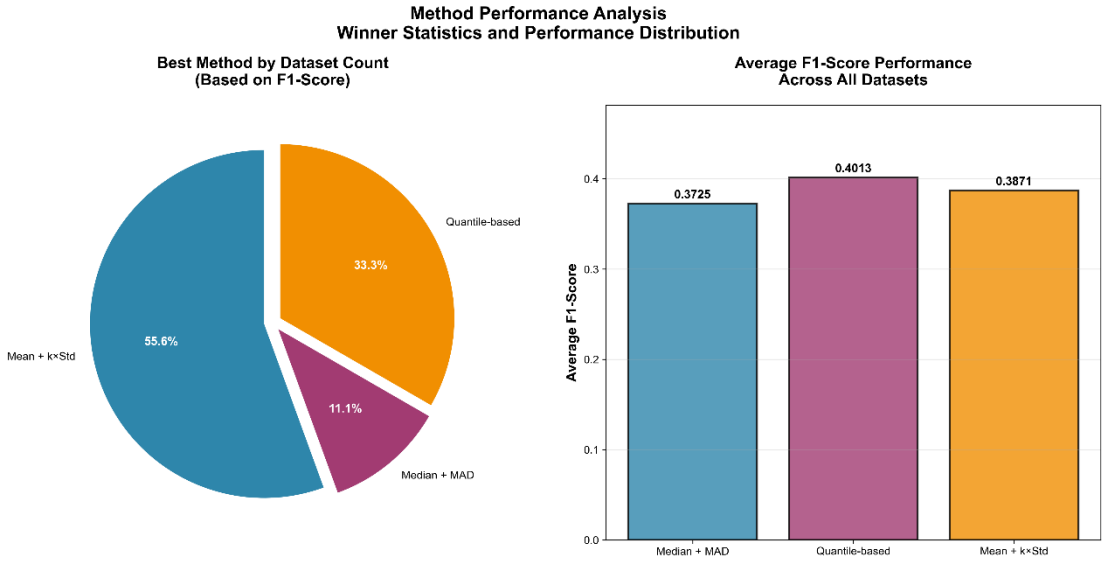


Figure 5.46: Overall comparison of threshold selection methods across datasets.

Figure 5.46 summarizes the overall performance of the three statistical thresholding strategies across all evaluated datasets. The left chart shows the number of datasets for which each method achieves the best F1-score, while the right chart presents the average F1-score across all datasets.

The results indicate that the mean + k·std method achieves the highest number of dataset-level wins (55.6%), suggesting strong consistency across different traffic scenarios. The quantile-based method ranks second in terms of wins (33.3%), while the median + MAD method achieves the best performance in fewer datasets (11.1%).

However, when considering average performance across all datasets, the quantile-based method achieves the highest mean F1-score (0.4013), followed by mean + k·std (0.3871) and median + MAD (0.3725). This suggests that although the mean + k·std method more frequently achieves the top performance on individual datasets, the quantile-based approach provides slightly better overall performance on average.

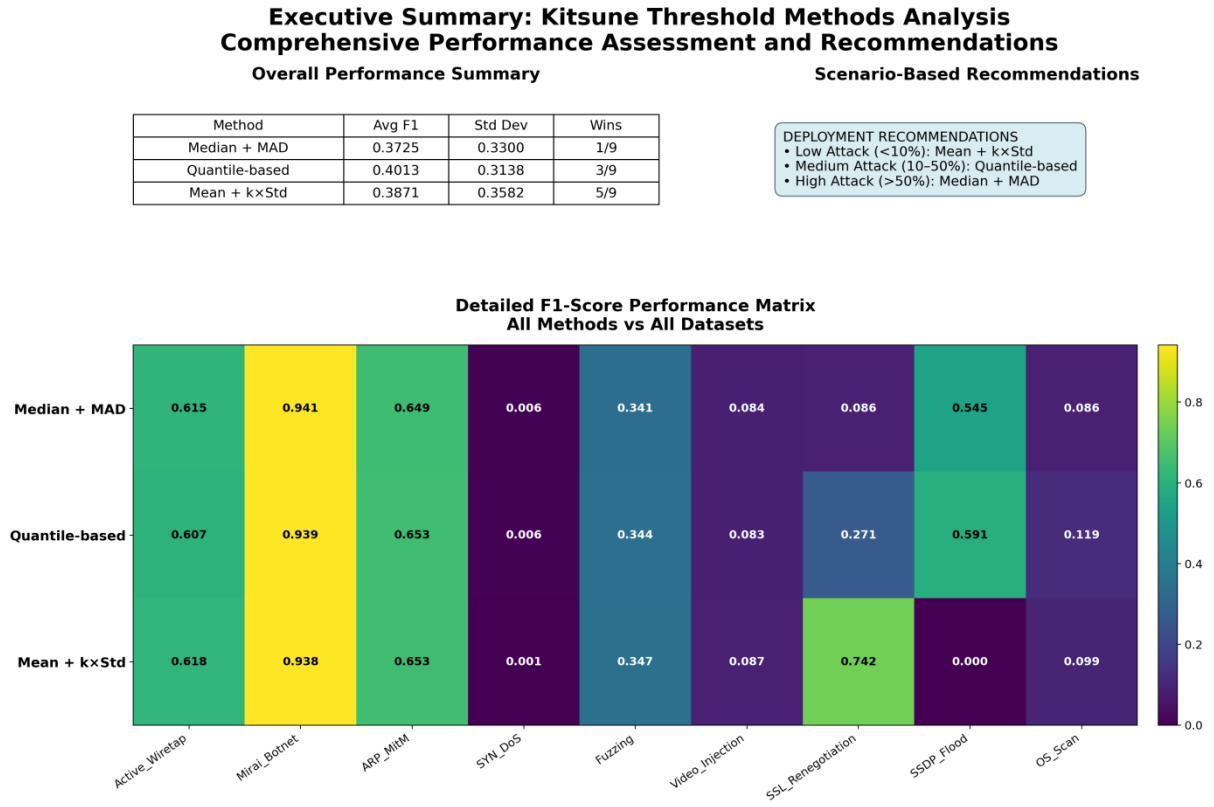


Figure 5.47: Executive summary of threshold selection performance across datasets.

Figure 5.47 provides a comprehensive summary of the performance of the three thresholding strategies across all evaluated datasets. The table in the upper-left section reports the overall statistics, including average F1-score, standard deviation, and the number of dataset-level wins for each method.

The results show that the quantile-based method achieves the highest average F1-score (0.4013), indicating better overall performance across datasets. The mean + k·std

method achieves the highest number of wins (5 out of 9 datasets), suggesting strong consistency in different traffic scenarios. In contrast, the median + MAD method demonstrates lower average performance but maintains robustness in certain datasets.

The heatmap in the lower section illustrates the F1-score performance of each method across individual datasets, highlighting that no single thresholding strategy consistently dominates all scenarios. Instead, different datasets favor different threshold-selection approaches.

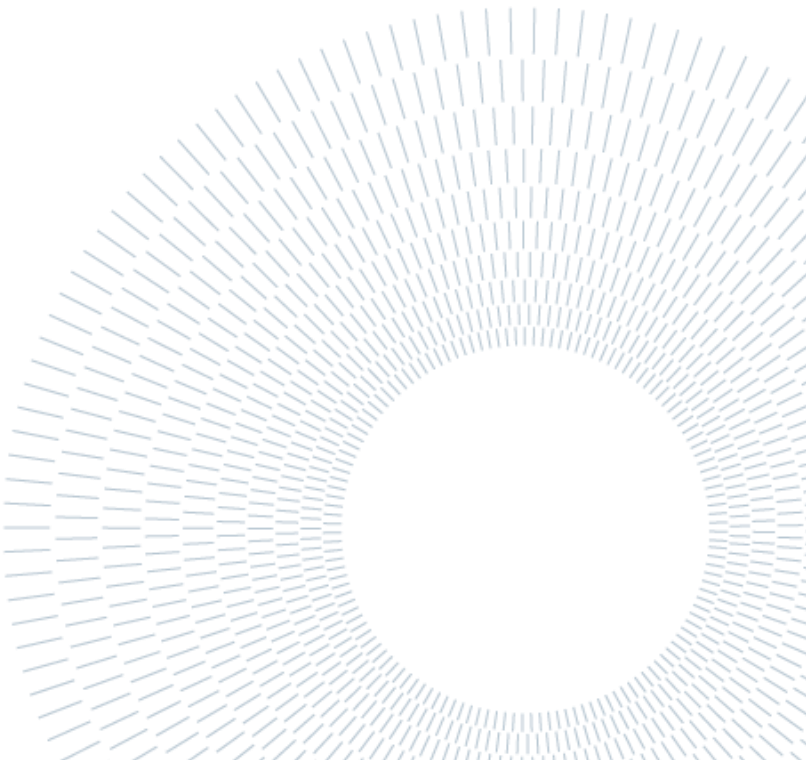
Based on these observations, scenario-based deployment recommendations are provided:

Low attack ratio (<10%): Mean + k·std

Medium attack ratio (10–50%): quantile-based

High attack ratio (>50%): median + MAD

Overall, this figure shows that threshold selection should be adapted to dataset characteristics rather than relying on a single universal strategy.



6 Conclusions and Further Developments

The overall evaluation across all datasets shows that no single threshold-selection strategy consistently outperforms the others in every scenario. The quantile-based method achieves the highest average F1-score, while the mean + k·std approach obtains the largest number of dataset-level wins, indicating strong consistency across different traffic conditions. The median + MAD method, although less competitive on average, demonstrates robustness in high-attack scenarios.

These results suggest that threshold selection should be adapted to traffic characteristics, particularly the attack ratio and score distribution. In practice, mean + k·std is suitable for low-attack environments, the quantile-based method performs well under moderate attack ratios, and median + MAD is more reliable when attack traffic dominates.

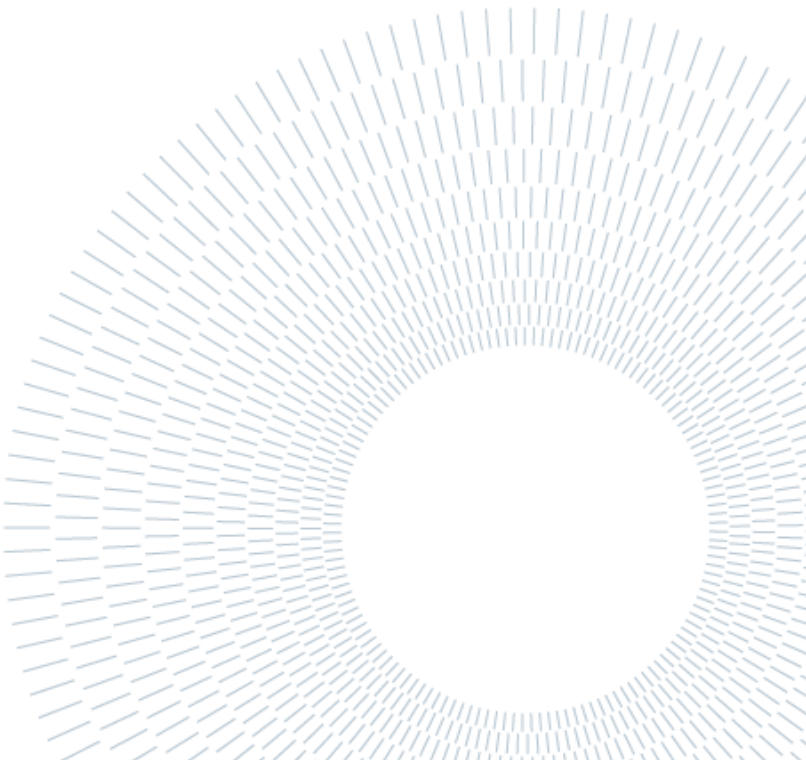
Overall, this study demonstrates that threshold design plays an important role in anomaly-based NIDS, but its effectiveness depends on dataset characteristics rather than a universal rule.

Future work

Future research may explore adaptive threshold-selection mechanisms that dynamically adjust to changing network conditions, rather than relying on fixed statistical rules. For example, online learning or reinforcement-based strategies could be used to update thresholds in real time according to traffic distribution shifts.

Another promising direction is improving feature representation. Since the experiments show that threshold selection alone cannot always compensate for limited anomaly-score separability, future work could investigate more expressive temporal and relational features, lightweight representation learning methods, or feature-selection techniques tailored for resource-constrained NIDS.

In addition, evaluating the proposed threshold-selection strategies in real-world deployment scenarios, such as distributed edge environments or streaming network-monitoring systems, would provide further insight into their robustness and scalability.



References

[1] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, 'Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection', May 27, 2018, *arXiv*: arXiv:1802.09089. doi: 10.48550/arXiv.1802.09089.

[2] E. A. Freeman and G. G. Moisen, 'A comparison of the performance of threshold criteria for binary classification in terms of predicted prevalence and kappa', *Ecological Modelling*, vol. 217, no. 1–2, pp. 48–58, Sept. 2008, doi: 10.1016/j.ecolmodel.2008.05.015.

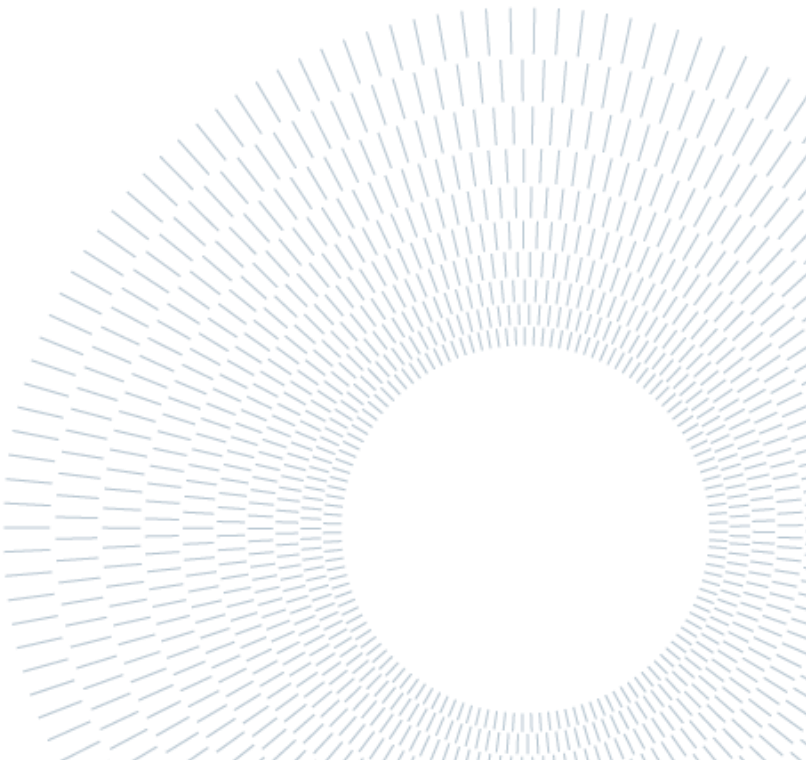
[3] T. A. Deepak, 'XGBoost Classification based Network Intrusion Detection System for Big Data using PySparkling Water', *IJATCSE*, vol. 9, no. 1, pp. 377–382, Feb. 2020, doi: 10.30534/ijatcse/2020/55912020.

[4] Muhammad Arif Faizin 'Optimizing Feature Selection Method in Intrusion Detection System Using Thresholding', *IJIES*, vol. 17, no. 3, pp. 214–226, June 2024, doi: 10.22266/ijies2024.0630.18.

[5] Jaeyeon Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, 'Fast portscan detection using sequential hypothesis testing', in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, Berkeley, CA, USA: IEEE, 2004, pp. 211–225. doi: 10.1109/SECPRI.2004.1301325.

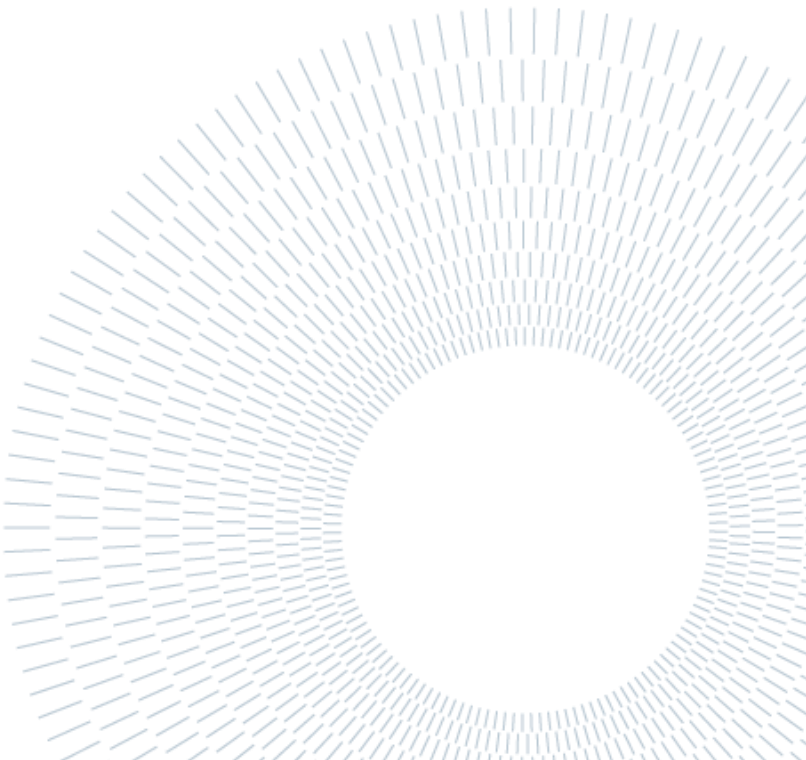
[6] W. Elbakri, M. Md. Siraj, B. A. S. Al-rimy, S. N. Qasem, and T. Al-Hadhrami, 'Adaptive Cloud Intrusion Detection System Based on Pruned Exact Linear Time Technique', *CMC*, vol. 79, no. 3, pp. 3725–3756, 2024, doi: 10.32604/cmc.2024.048105.

[7] A. Komadina, M. Martinić, S. Groš, and Ž. Mihajlović, 'Comparing Threshold Selection Methods for Network Anomaly Detection', *IEEE Access*, vol. 12, pp. 124943–124973, 2024, doi: 10.1109/ACCESS.2024.3452168.



List of Figures

List of Tables



List of Symbols

Variable	Description
$\left\|\left\|l^i\right\|\right\|$	number of neurons in the i-th layer
W^i	weight matrix connecting
b^i	bias vector added to the activations
X	input space
y	output space