International Journal of Creative and Open Research in Engineering and Management
ISSN: 3108-1754 (Online)
Volume 02 Issue 11 November-2026 | Impact Factor: 3.5



AI-Based Autonomous Network Monitoring and Anomaly Detection with Automated Alerts

PONEY JOSHWAA A[1], NITHIYANANDHAM J[2], NAVEENESWAR P G[3], Mrs. MOGANAPRIYA L[4]

1, 2, 3 B.E. Students, Department of Computer Science and Engineering, E.G.S. Pillay Engineering College, Nagapattinam, Tamil Nadu, India

4 Assistant Professor, Department of Computer Science and Engineering, E.G.S. Pillay Engineering College, Nagapattinam, Tamil Nadu, India

How to Cite this Article:

A, P. J., J, N. & G, N. P. (2026). AI-Based Autonomous Network Monitoring and Anomaly Detection with Automated Alerts. International Journal of Creative and Open Research in Engineering and Management, 02(11), 1-9. https://doi.org/10.55041/ijcope.v2i11.054

License:

This article is published under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

© The Author(s). Published by International Journal of Creative and Open Research in Engineering and Management.







https://doi.org/10.55041/ijcope.v2i11.054

Abstract — Detecting anomalous records in network-traffic-like data calls for methods that work without labels and for a way to pass detections to an operator. This paper evaluates a student prototype that combines an Isolation Forest with a reconstruction-based neural detector, merges their binary decisions with a logical OR rule, classifies severity with fixed rules, and builds webhook payloads for an n8n alert workflow. The data are 1,000 synthetic records containing 50 injected anomalies of two kinds. The original implementation scored every record, including records used for fitting, with a fixed reconstruction threshold; re-running it reproduced the saved report exactly, with 286 flagged records and precision 0.175. Under a leakage-free protocol, in which preprocessing, training and thresholds use only a stratified 60% training partition and testing uses an untouched 20% partition, repeated over ten splits, Isolation Forest reached a mean F1-score of 0.82 (precision 0.82, recall 0.85), the reconstruction detector 0.57, and the OR combination 0.75 (recall 0.92, precision 0.64). The OR rule therefore raised recall, mainly for slow-leak-like anomalies, but did not improve the F1-score, and with the original fixed threshold it flagged more than a quarter of normal records. The web interface uses simulated routes, and delivery of alert e-mails was not verified. The results apply to synthetic data only and motivate evaluation on real labelled traffic.

Keywords — anomaly detection; Isolation Forest; reconstruction error; autoencoder; OR fusion; automated alerting

I. INTRODUCTION

Anomaly detection is the task of identifying observations that deviate from the behaviour expected of a system, and it underlies applications such as fault detection, fraud screening and network monitoring [1], [2]. In network monitoring, the volume of records exceeds what an operator can inspect manually, and fixed rules can miss patterns that were not anticipated when the rules were written. Labelled examples of abnormal behaviour are also scarce, which motivates unsupervised detectors that learn what is typical from unlabelled data [1], [2].

Two widely used families of unsupervised detectors rest on different assumptions. Isolation Forest isolates observations by random recursive partitioning and treats those that are isolated after few splits as anomalous [3], [4]. Reconstruction-based detectors, typically autoencoders, learn to reproduce their input through a constrained representation and flag records that are reproduced poorly [5], [6]. Because the two methods can disagree about which records are unusual, combining them is attractive, but a combination also changes the balance between missed anomalies and false alarms, and that balance has to be measured rather than assumed [7]. Detection is also only half of a monitoring workflow: an event must reach a human, and workflow-automation tools such as n8n receive structured events through webhooks and route them to e-mail or other channels [8].

This paper examines a student prototype, AI-Based Anomaly Detection and Automated Alerting System, that implements these ideas on synthetic network-traffic-like records: preprocessing, an Isolation Forest, a reconstruction-based detector, a binary OR rule that combines their decisions, CSV and plot reporting, a rule-based severity classifier, and a webhook payload for an n8n workflow. A web interface built with React and Express accompanies the Python code. The prototype's original report evaluated detectors on the same records used to fit them, used a fixed reconstruction



threshold, and described a score-fusion design that the code does not implement. Whether the combination is actually useful, and which parts of the end-to-end workflow have been demonstrated, were therefore open questions.

II. PROBLEM STATEMENT

The study addresses the difficulty of identifying anomalous network-traffic-like records when labelled examples are scarce and fixed rules may miss unexpected behaviour. It also examines the gap between detecting an event and reliably delivering it to an operator. The prototype's original evaluation scored records that had also been used during fitting, applied a fixed reconstruction threshold, and described a score-fusion design that was not implemented in the inspected code. The present evaluation therefore separates detection performance from alert delivery and distinguishes implemented, simulated, planned and verified components.

Positioning. Hybrid detection and workflow-based alerting are not new, and no novelty claim is made. The question addressed here is narrower: for one concrete, simple pipeline, what does a leakage-free evaluation show about each detector, the OR rule and the threshold, and which parts of the end-to-end workflow are demonstrated?

III. OBJECTIVES

The objectives of this paper are:

The objectives of this paper are: (O1) to document what the code implements, separating implemented, simulated, planned and verified components; (O2) to compare Isolation Forest, the reconstruction detector and their OR combination on the same held-out records under a protocol that avoids preprocessing and threshold leakage; (O3) to quantify how the fixed reconstruction threshold and the OR rule affect false alarms; and (O4) to state precisely what remains unverified. The contributions are limited accordingly: no new detection algorithm is proposed; the contribution is a reproducible, leakage-free evaluation of a specific lightweight pipeline, an exact account of its implementation boundaries, and a released evaluation script.

IV. LITERATURE REVIEW

Isolation-based detection. Isolation Forest builds an ensemble of random trees and uses the average path length needed to isolate a record as its anomaly score, without a distance or density model [3], [4]. Its axis-parallel splits can assign misleading scores in regions away from the data axes, which Extended Isolation Forest addresses with randomly oriented hyperplane cuts [9]; Deep Isolation Forest maps data through randomly initialised neural networks before partitioning in order to isolate “hard” anomalies in high-dimensional or non-linearly separable data [10]. In all variants the proportion of records flagged depends on a score threshold, which in the scikit-learn implementation is derived from a user-supplied contamination fraction [11].

Reconstruction-based detection. Autoencoders learn a low-dimensional representation by reconstructing their input [5]. Sakurada and Yairi applied autoencoders to anomaly detection and compared them with linear and kernel principal component analysis [6], and Kitsune used an ensemble of small autoencoders for unsupervised online network intrusion detection [12]. Surveys stress that the outcome depends on the data, the network design and the threshold applied to the reconstruction error, which is not a calibrated probability of abnormality [1], [2].

Combining detectors and alerting. Outlier-ensemble methods combine the outputs of several base detectors [7]. The combination may operate on continuous scores, which need normalisation, or on binary decisions; a logical OR of binary decisions flags the union of what the components flag and so cannot reduce false positives relative to either component. Automated alerting is a separate stage: n8n exposes webhook endpoints that start workflows [8], but a successful webhook call, a completed workflow run and a delivered e-mail are distinct events. Table I summarises the approaches relevant to this work. Additional hybrid autoencoder–Isolation-Forest studies summarised in the Phase I report [13] were not independently re-verified for this paper and are therefore not cited individually.

Table I: Approaches relevant to the prototype.

Approach	Principle	Strength	Limitation	Relevance here
Isolation Forest [3], [4]	Random trees; short isolation path = anomalous	No labels; scalable; no density model	Axis-parallel splits; flagged share set by contamination	Detector 1 (implemented)
Extended Isolation Forest [9]	Random hyperplane splits	Removes score artefacts along axes	Not in scikit-learn core	Possible upgrade
Deep Isolation Forest [10]	Random neural mappings before isolation	Targets hard, non-linear anomalies	Added complexity	Future work



Autoencoder reconstruction [5], [6]	High reconstruction error = anomalous	Learns non-linear structure; unsupervised	Threshold-sensitive; error is not a probability	Detector 2 (implemented)
Autoencoder ensemble (Kitsune) [12]	Ensemble of small autoencoders, online	Efficient online NIDS	Needs packet-level feature extraction	Context; not reproduced
Outlier ensembles [7]	Combine base-detector outputs	May cover complementary anomalies	Combination rule determines error trade-off	Binary OR (implemented)

Positioning. Hybrid detection and workflow-based alerting are not new, and no novelty claim is made. The question addressed here is narrower: for one concrete, simple pipeline, what does a leakage-free evaluation show about each detector, the OR rule and the threshold, and which parts of the end-to-end workflow are demonstrated?

V. PROPOSED SYSTEM

The prototype combines synthetic-data generation and preprocessing, an Isolation Forest detector, a reconstruction-based detector, binary OR fusion, CSV and plot reporting, rule-based severity classification, and webhook-payload construction for an n8n workflow. A React and Express web demonstrator accompanies the Python pipeline; the two applications are separate in the inspected implementation.

Each record is a feature vector x of dimension d = 6 (five numeric traffic features and one encoded protocol field). A reference label $y \in \{0, 1\}$ (1 = injected anomaly) exists because the data are synthetic; it is used only to evaluate, never to train or tune. A detector outputs a binary decision $\hat{y}$ for each record, and a record with $\hat{y} = 1$ is eligible for an alert. Detection quality and alert delivery are treated as separate questions.

The detector is scikit-learn's IsolationForest [11] with contamination = 0.05 and a fixed random seed, all other settings at library defaults. A record is flagged when the library's predict returns −1. The contamination value sets the decision offset at the 5th percentile of training scores, so it acts as a prior on the anomaly rate; here it equals the true rate of 5% by construction of the generator, which favours this detector relative to a practitioner who does not know the rate.

The model maps the six inputs through hidden layers of 16, 8 and 16 units (ReLU) to six outputs. The source offers a Keras version (linear output, Adam [14], MSE loss, 20 epochs, batch size 32) [15] and, when TensorFlow cannot be imported, a scikit-learn MLPRegressor with the same hidden sizes, max_iter = 100 and a fixed seed. TensorFlow was not installed in the environment used here, so every reported result uses the scikit-learn variant. Note that the narrowest hidden layer (8 units) is wider than the input (6), so the network is over-complete and can in principle approximate the identity map; no ablation of this design was run. The anomaly score is the per-record mean squared reconstruction error

The implemented decision rule is a logical OR of the two binary outputs,

$$\hat{y}OR = \max(\hat{y}IF, \hat{y}AE) \quad (2)$$

which flags the union of the two detectors' positives. The Phase I report and the Review 2 slides describe a two-tier score-fusion rule intended to reduce false positives; that rule is not present in the code, and no result in this paper concerns it. For interpretation only, a logical AND (the intersection) is also reported as an ablation; it is not part of the project.

VI. SYSTEM ARCHITECTURE

B. System overview and implementation status

Figure 1 shows the verified structure. The Python pipeline (A) runs generation, preprocessing, both detectors, OR fusion, reporting and alert-payload construction. The web demonstrator (B) is a separate React and Express application whose routes generate random records and select those whose pre-existing is_anomaly field equals 1; it does not call the Python models. The source repository contains Python (scikit-learn, optional TensorFlow/Keras, pandas, matplotlib, seaborn, requests), TypeScript (React, Express, Vite, nodemailer) and an exported n8n workflow. Table II states the evidence status of each component.

Figure 1: System architecture as verified from the source code. Dashed elements are configured but not shown to work end to end; the dotted line marks the absence of a verified connection between (A) and (B).



Table II: Implementation and verification status of components.

Component	Implemented in source	Verified by execution
Data generator, preprocessing	Yes	Yes (re-run; output identical to saved CSV)
Isolation Forest	Yes	Yes
Reconstruction model	Keras and sklearn MLP variants	sklearn MLP only (no TensorFlow here)
Binary OR fusion	Yes	Yes
Score fusion (two-tier)	No (Phase I/II design only)	Not applicable
CSV report and plot	Yes	Yes (saved report reproduced exactly)
Python webhook payload and severity rules	Yes	Console fallback only; no live webhook
n8n workflow (validate, severity, e-mail)	Exported JSON; placeholder addresses	No
E-mail delivery to a mailbox	SMTP node and nodemailer route	No evidence of receipt
Web UI to Python detector link	No (simulated routes)	Not applicable

C. Dataset

The data come from the project's own generator (data_collector.py, NumPy seed 42). They are synthetic; no real network traffic is involved. Re-running the generator reproduced the CSV in the project archive exactly. Table III summarises the generating process. A second file in the archive, attached_dataset.csv (307 rows, two unlabelled numeric columns, unknown provenance), is used only by a separate script and is excluded from the quantitative evaluation because it has no labels.

Table III: Synthetic dataset (1,000 records; 50 anomalies, 5.0%).

There are no missing values or duplicate rows. Because anomalies are produced by multiplying or replacing a few fields of otherwise ordinary records, they are well separated by construction; the data therefore test the pipeline's mechanics more than its ability to find realistic attacks.

Field	Normal records	Anomaly modification
duration	Gamma(2, 1)	×20 (slow-leak-like)
src_bytes	Lognormal(10, 1)	×5 (DDoS-like)
dst_bytes	Lognormal(8, 1)	×10 (slow-leak-like)
packets	Poisson(50)	×10 (DDoS-like)
cpu_load	Uniform(10, 40)	Uniform(80, 100) (DDoS-like)
protocol_type	tcp / udp / icmp, uniform	unchanged
Counts	950 normal	25 DDoS-like + 25 slow-leak-like

VII. METHODOLOGY

D. Preprocessing

The preprocessing module imputes missing numeric values with the median, label-encodes protocol_type as an integer, selects the five numeric fields plus the encoded protocol, and standardises all six with a z-score scaler. The original script fitted these steps on all 1,000 records before splitting; in the evaluation reported here the split comes first and the encoder and scaler are fitted on the training partition only, with unseen categories mapped to −1. Encoding a nominal field as an ordinal integer imposes an arbitrary order and is retained only for fidelity to the project.

E. Isolation Forest

The detector is scikit-learn's IsolationForest [11] with contamination = 0.05 and a fixed random seed, all other settings at library defaults. A record is flagged when the library's predict returns −1. The contamination value sets the decision offset at the 5th percentile of training scores, so it acts as a prior on the anomaly rate; here it equals the true rate of 5% by construction of the generator, which favours this detector relative to a practitioner who does not know the rate.



F. Reconstruction-based detector

The model maps the six inputs through hidden layers of 16, 8 and 16 units (ReLU) to six outputs. The source offers a Keras version (linear output, Adam [14], MSE loss, 20 epochs, batch size 32) [15] and, when TensorFlow cannot be imported, a scikit-learn MLPRegressor with the same hidden sizes, max_iter = 100 and a fixed seed. TensorFlow was not installed in the environment used here, so every reported result uses the scikit-learn variant. Note that the narrowest hidden layer (8 units) is wider than the input (6), so the network is over-complete and can in principle approximate the identity map; no ablation of this design was run. The anomaly score is the per-record mean squared reconstruction error

$$e(x) = (1/d)\ \Sigma j\ \ (xj - \hat{x}j)2 \qquad (1)$$

and a record is flagged when $e(x) > \tau$. The code fixes $\tau = 0.1$ (the threshold default of predict_autoencoder). Two settings are evaluated: the fixed value ("as implemented") and a training-calibrated value τ = the 95th percentile of e over the training partition, which uses no labels and mirrors the percentile rule used in the project's run_attached_dataset.py. The 95th-percentile choice was fixed before the held-out runs; other percentiles are reported only as a sensitivity analysis and are not used for selection.

G. Binary OR fusion

The implemented decision rule is a logical OR of the two binary outputs,

$$\hat{y}OR = \max(\hat{y}IF, \hat{y}AE) \quad (2)$$

which flags the union of the two detectors' positives. The Phase I report and the Review 2 slides describe a two-tier score-fusion rule intended to reduce false positives; that rule is not present in the code, and no result in this paper concerns it. For interpretation only, a logical AND (the intersection) is also reported as an ablation; it is not part of the project.

H. Reporting and alerting

The reporter writes a CSV containing the input fields and the columns iso_anomaly, ae_anomaly and final_anomaly, and a duration–packets scatter plot. For each flagged record the alert module builds a JSON payload (event, timestamp, severity, description, affected fields, recommendation). Severity is rule-based on raw field values: critical if CPU load ≥ 80, packets ≥ 400 or src_bytes $\geq 100{,}000$; high if CPU load ≥ 60, packets ≥ 200 or src_bytes $\geq 30{,}000$; medium if CPU load ≥ 40 or packets ≥ 100; otherwise low. The payload's anomaly_score is the mean of the two binary flags (0.5 or 1.0 for a flagged record), a vote fraction rather than a calibrated score. A 60-second cooldown suppresses repeated alerts for the same record. If no webhook URL is configured the payload is printed to the console; otherwise it is posted with a shared-secret header.

The exported n8n workflow receives the POST at the path network-anomaly, checks that the event is anomaly_detected and that a severity is present, e-mails only critical and high events through an SMTP node, and returns HTTP 200 for low and medium events without e-mail. In the exported file the webhook node has no authentication configured, so the shared secret sent by the client is not checked by the workflow as exported, and the sender and recipient addresses are placeholders. A bundled mock_n8n.py returns a fixed success message for any POST to the path; its reply is not evidence of workflow execution or e-mail delivery. In the web demonstrator, the run-detection route posts at most one alert per run and the report route sends an HTML report through SMTP or a test account.

I. Evaluation protocol

The 1,000 records are split once per run, stratified by label, into 600 training, 200 validation and 200 test records (30, 10 and 10 anomalies). Encoder, scaler, Isolation Forest, reconstruction network and the percentile threshold use the training partition only. Labels are used only to compute metrics. The validation partition was reserved and not used for tuning; its results are stored with the run records. Each detector and the combinations are scored on the same test records. Precision, recall, F1-score and false-positive rate (FPR) are

$$P = TP/(TP+FP), \quad R = TP/(TP+FN) \qquad (3)$$

$$F1 = 2PR/(P+R), \quad FPR = FP/(FP+TN) \quad (4)$$

with an undefined ratio recorded as not-a-number rather than zero. Because a test partition holds only 10 anomalies, the protocol is repeated for ten seeds (0–9), each of which changes the split and the model initialisation but not the dataset; mean ± sample standard deviation over these runs is reported, together with the confusion counts of the single run with seed 42. The ten runs reuse the same 1,000 records and their test sets overlap, so the spread reflects sensitivity to the split and initialisation, not independent samples of the data. ROC-AUC and PR-AUC are computed from the continuous scores (negated Isolation Forest score and reconstruction error). Environment: Python 3.12.3, scikit-learn 1.8.0, NumPy 2.4.4, pandas 3.0.2; the project pins different versions (scikit-learn 1.4.0, NumPy 1.26.3, pandas 2.2.0). Hardware: [not recorded; authors to supply].



VIII. RESULTS AND DISCUSSION

A. Reproduction of the saved report (in-sample, provisional)

Re-running the project's own code with the scikit-learn variant reproduced the saved anomaly_report.csv exactly: all 1,000 rows of the two detector columns and the fused column agreed (Isolation Forest 49, reconstruction 283, OR 286 flagged). This is consistent with the saved report having been produced by the scikit-learn fallback, although it does not prove which backend the authors used. In that script the detectors were trained on 80% of the records but applied to all 1,000, so 800 of the evaluated records were seen during fitting, and preprocessing was fitted on all records. Table IV compares the flags with the labels. These in-sample figures are descriptive only and are not evidence of generalisation.

The saved OR output flags 286 records (28.6%) for 50 true anomalies, giving precision 0.175; almost all of the false alarms come from the reconstruction detector, whose fixed threshold flags 283 records.

B. Held-out evaluation

Table V gives the held-out results. Isolation Forest alone is the strongest single configuration: mean F1 0.82, precision 0.82, recall 0.85, FPR 0.012, with ROC-AUC 0.991 and PR-AUC 0.927. The reconstruction error separates the classes less well (ROC-AUC 0.920, PR-AUC 0.551); PR-AUC is the more informative figure at 5% prevalence [16].

Fixed threshold. With $\tau = 0.1$, the reconstruction detector flags a mean of 64 of 200 test records (FPR 0.292) and the OR rule inherits this (F1 0.28). Figure 2 shows why: in the seed-42 run 62% of the training records already exceed 0.1, so the value lies inside the bulk of normal reconstruction errors. The networks also did not converge: every one of the eleven training runs stopped at the 100-iteration cap, with final training loss (scikit-learn's half-squared-error objective) between 0.046 and 0.216, so reconstruction errors differ widely between seeds and a single absolute threshold cannot be transferred between runs (FPR s.d. 0.135).

Figure 2: Distribution of training reconstruction error (seed 42) with the code's fixed threshold (0.1) and the 95th-percentile threshold calibrated on the same partition.

Training-calibrated threshold. Calibrating τ on the training partition removes most of the false alarms (reconstruction FPR 0.027), but the detector is then weak on its own (F1 0.57). Combining it with Isolation Forest by OR raises mean recall from 0.85 to 0.92 while lowering mean precision from 0.82 to 0.64; the net effect is a lower mean F1 (0.75 versus 0.82; Figure 3). The OR rule had a higher F1 than Isolation Forest alone in 2 of ten splits, a lower F1 in 7 and a tie in 1; the mean paired difference is 0.072 (s.d. 0.075). No significance test was run, and with ten test anomalies per split these data support only the conclusion that OR fusion did not improve F1 here.

Figure 3: Held-out precision, recall and F1-score (mean ± s.d. over ten stratified splits; reconstruction threshold calibrated on training data).

Complementarity. Pooled over the ten test partitions, Isolation Forest detected 51 of 51 DDoS-like anomalies (100%) but only 34 of 49 slow-leak-like anomalies (69%); the calibrated reconstruction detector detected 26 (51%) and 35 (71%); and the OR rule detected 51 (100%) and 41 (84%). The reconstruction detector therefore adds some slow-leak-like detections that Isolation Forest misses, which is the behaviour the hybrid design anticipates, but at the price of extra false alarms. The AND ablation (Table V) shows the opposite trade: higher precision, lower recall.

Threshold sensitivity. Moving the percentile that sets τ changes the OR result substantially (mean F1 0.55 at the 90th, 0.75 at the 95th, 0.81 at the 97.5th and 0.82 at the 99th percentile), approaching the Isolation Forest result as the reconstruction detector becomes stricter. This sweep is descriptive; it must not be used to select a threshold from test results.

For the seed-42 run (Table VI), the OR rule with the calibrated threshold produced 19 alerts on 200 test records, of which 10 were false; with the fixed threshold it produced 132 alerts, 122 of them false. Alert volume of this order would be burdensome if every flagged record triggered a notification.

Table VI: Confusion counts on the 200-record test partition (seed 42; 10 anomalies).

Method	TP	FP	TN	FN	P	R	F1	FPR
Isolation Forest	43	6	944	7	0.878	0.860	0.869	0.006
Reconstruction	47	236	714	3	0.166	0.940	0.282	0.248
OR fusion	50	236	714	0	0.175	1.000	0.298	0.248



Config.	Method	Precision	Recall	F1-score	FPR	Flagged / 200
Fixed τ = 0.1	Reconstruction	0.16 ± 0.06	0.90 ± 0.08	0.26 ± 0.08	0.292 ± 0.135	64.5 ± 26.1
Fixed τ = 0.1	OR fusion	0.17 ± 0.06	0.97 ± 0.05	0.28 ± 0.09	0.292 ± 0.135	65.2 ± 25.8
Fixed τ = 0.1	AND (ablation)	0.81 ± 0.12	0.78 ± 0.14	0.78 ± 0.07	0.012 ± 0.010	10.0 ± 2.9
Calibrated τ	Isolation Forest	0.82 ± 0.12	0.85 ± 0.13	0.82 ± 0.07	0.012 ± 0.010	10.7 ± 2.7
Calibrated τ	Reconstruction	0.56 ± 0.13	0.61 ± 0.10	0.57 ± 0.07	0.027 ± 0.012	11.3 ± 2.7
Calibrated τ	OR fusion	0.64 ± 0.11	0.92 ± 0.06	0.75 ± 0.06	0.029 ± 0.012	14.7 ± 2.7
Calibrated τ	AND (ablation)	0.78 ± 0.16	0.54 ± 0.08	0.62 ± 0.05	0.010 ± 0.008	7.3 ± 2.3

Config.	Method	TP	FP	TN	FN
Calibrated	IF	9	4	186	1
Calibrated	Recon.	5	10	180	5
Calibrated	OR	9	10	180	1
Fixed 0.1	Recon.	10	122	68	0
Fixed 0.1	OR	10	122	68	0

C. Alerting and web interface

No result is reported for alert delivery because none could be verified. The repository contains no execution log of a live n8n instance, a received e-mail, or a webhook response from a real workflow; the only success message in the archive comes from the mock server, which answers every POST identically. The workflow definition and the Python and Express senders show an intended path, and the payload construction and cooldown logic can be inspected in the code, but webhook invocation, workflow execution, SMTP configuration and receipt in a mailbox remain four separate unverified steps. The web interface is a demonstrator: its upload route ignores the file contents, its records come from a random generator, and its detection route returns the records whose is_anomaly value is already 1.

D. Discussion

The evidence supports three conclusions about this prototype. First, the in-sample precision of the original report (0.175) mostly reflects an uncalibrated reconstruction threshold, not the intrinsic limits of the approach: with a threshold derived from training data the OR rule's precision rises to 0.64, and Isolation Forest alone reaches 0.82. Second, the OR rule's benefit is a modest recall gain concentrated in one anomaly type, bought with a larger drop in precision; whether that is acceptable depends on the relative cost of a missed event and of a false alarm, which was not specified for this project. Third, the project documents describe the fusion as reducing false positives, but a logical OR cannot do so; the stated rationale applies to the unimplemented two-tier rule.

These findings should not be generalised. The data are synthetic and the anomalies are crude multiplicative perturbations; both detectors were configured with a 5% prior equal to the true prevalence; each test partition contains ten anomalies; the ten repetitions share one dataset; and the reconstruction network is under-trained, over-complete and evaluated with a single hyper-parameter setting. Performance on real traffic, other anomaly rates, or a Keras implementation is unknown. The comparison with other algorithms is limited to the three configurations above, so no claim of superiority over published methods is made.

IX. ADVANTAGES AND LIMITATIONS

A. Advantages supported by the evaluation

The verified strengths are the reproducible synthetic-data pipeline, the Isolation Forest held-out result, and the recall increase obtained by OR fusion when the reconstruction threshold is calibrated on training data. The OR combination detected additional slow-leak-like anomalies, although this came with lower precision and a lower mean F1-score than Isolation Forest alone. These findings are specific to the synthetic dataset and evaluation protocol.



B. Limitations

Limitations that apply to the inspected evidence are: synthetic data of uncertain realism; a single dataset; threshold sensitivity; an unconverged reconstruction model whose original backend is inferred, not recorded; a web interface not connected to the models; an n8n workflow without authentication or real recipients and no evidence of delivery; and no environment lock-file or run logs in the archive. Future work should (i) evaluate on public labelled network-traffic data and at several anomaly rates; (ii) train the reconstruction model on normal records only, with a genuine bottleneck, early stopping and a validation-based threshold; (iii) implement and test continuous score fusion against the simple baselines; (iv) add ROC and precision–recall analysis with confidence intervals across independently generated datasets; (v) connect the web interface to the detector through a tested API; and (vi) test the alert path end to end with authentication, retries and a delivery log, and measure alert latency.

X. FUTURE SCOPE

Future work identified in the source includes evaluation on public labelled network-traffic data and at several anomaly rates; training the reconstruction model on normal records only with a genuine bottleneck, early stopping and a validation-based threshold; implementing and testing continuous score fusion; adding ROC and precision–recall analysis with confidence intervals across independently generated datasets; connecting the web interface to the detector through a tested API; and testing the alert path end to end with authentication, retries, delivery logging and alert-latency measurement.

XI. CONCLUSION

The prototype implements Isolation Forest, a reconstruction-error detector, a binary OR combination and a webhook-oriented alert path on synthetic network-traffic-like records. Its saved report was reproduced exactly, and showed 286 flagged records and precision 0.175 because of a fixed, uncalibrated reconstruction threshold. Under a leakage-free held-out protocol repeated over ten splits, Isolation Forest alone gave the best F1-score (0.82); the OR combination raised recall (0.92) through additional slow-leak-like detections but reduced precision and did not improve F1 (0.75). Alert delivery and the link between the web interface and the detector were not verified. The next steps are evaluation on real labelled traffic, a properly trained and calibrated reconstruction model, and a tested end-to-end alert path.

DECLARATIONS

Funding: [authors to state the actual funding status]. Conflicts of interest: [authors to declare; no statement is made here]. Author contributions: [authors to supply; the Review 2 slides list planned Phase I roles, which are not a substitute]. Ethics: the study uses only synthetic data and involves no human participants or personal data. Data and code availability: the data generator, the held-out evaluation script, saved predictions and figures are provided with this submission; a public repository is [URL to be inserted after secrets are removed from the project archive].

ACKNOWLEDGMENT

The authors thank [confirm: Ms. Moganapriya L, Assistant Professor, Department of Computer Science and Engineering, who is named as project guide in the Phase I report] and the Department of Computer Science and Engineering, E.G.S. Pillay Engineering College (Autonomous), Nagapattinam.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Comput. Surv., vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009, doi: 10.1145/1541880.1541882.

[2] G. Pang, C. Shen, L. Cao, and A. van den Hengel, “Deep learning for anomaly detection: A review,” ACM Comput. Surv., vol. 54, no. 2, pp. 1–38, 2021, doi: 10.1145/3439950.

[3] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in Proc. 8th IEEE Int. Conf. Data Mining (ICDM), 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.

[4] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation-based anomaly detection,” ACM Trans. Knowl. Discov. Data, vol. 6, no. 1, pp. 3:1–3:39, Mar. 2012, doi: 10.1145/2133360.2133363.

[5] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, 2006, doi: 10.1126/science.1127647.

[6] M. Sakurada and T. Yairi, “Anomaly detection using autoencoders with nonlinear dimensionality reduction,” in Proc. MLSDA 2014 2nd Workshop Mach. Learn. Sensory Data Anal., 2014, pp. 4–11, doi: 10.1145/2689746.2689747.

[7] C. C. Aggarwal, Outlier Analysis, 2nd ed. Cham, Switzerland: Springer, 2017.



[8] n8n, "Webhook node," n8n Documentation. [Online]. Available: https://docs.n8n.io/integrations/builtin/core-nodes/n8n-nodes-base.webhook/

[9] S. Hariri, M. Carrasco Kind, and R. J. Brunner, "Extended isolation forest," IEEE Trans. Knowl. Data Eng., vol. 33, no. 4, pp. 1479–1489, Apr. 2021, doi: 10.1109/TKDE.2019.2947676.

[10] H. Xu, G. Pang, Y. Wang, and Y. Wang, "Deep isolation forest for anomaly detection," IEEE Trans. Knowl. Data Eng., vol. 35, no. 12, pp. 12591–12604, Dec. 2023, doi: 10.1109/TKDE.2023.3270293.

[11] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[12] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in Proc. Netw. Distrib. Syst. Security Symp. (NDSS), 2018, doi: 10.14722/ndss.2018.23204.

[13] A. Poney Joshwaa, J. Nithiyanandham, and P. G. Naveeneswar, "AI-based autonomous network monitoring and anomaly detection with automated alerts," Phase I Project Report, Dept. Comput. Sci. Eng., E.G.S. Pillay Eng. College (Autonomous), Nagapattinam, India, 2026, unpublished.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. Int. Conf. Learn. Represent. (ICLR), 2015, arXiv:1412.6980.

[15] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, arXiv:1603.04467.

[16] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," PLoS ONE, vol. 10, no. 3, Art. no. e0118432, 2015, doi: 10.1371/journal.pone.0118432.