2025 Johns Hopkins AI for CyberSecurity Certificate CapStone Assignment

Detecting Network-Centric Attacks from Behavioral Telemetry: A Comparison of Unsupervised, Gradient-Boosted, and Neural Models

Rodger Scoggin CISSP-ISSAP, CCSP

Dir. Security Architecture & Engineering, Pathward N.A.

rscoggin@pathward.com

Abstract---Endpoint anomaly detection is one facet of detective controls and a layer of defense that resides on systems that use and store data that attackers are after. Networks are the mode of transit component for data and the vectors over which attacks take place before they penetrate a host. As a result, modern Network Detection and Response (NDR) systems like DarkTrace and others must use many detection methods to effectively provide this, typically, initial layer of defense. Network focused anomaly detection is one of the methods for detection of malicious activities within an organization and machine learning plays a crucial role in advanced methods of such detection.

In this paper, I assess anomaly detection for network-centric attacks using session-level behavioral telemetry. I train Isolation Forest, Gradient-Boosted Trees (GBT), and a Multi-Layer Perceptron (MLP) against the binary attack_detected label under a forward-in-time split. At low false-alarm operating points suitable for Security Operations Center (SOC) workflows, Gradient Boosting leads on the time-based test slice (Accuracy = 0.894; PD/TPR = 0.765; PFA/FPR = 0.0019). Of the supervised models, only Gradient Boosting satisfied the ≤1% FPR budget; the MLP exceeded it (1.19%). I position this result against current approaches and outline operational tuning for sustained low alert volume [1][3][2].

I. INTRODUCTION

Attackers increasingly hide in normal traffic and payload inspection is often ineffective due to encryption and evasive techniques, however, the behavior remains observable without decrypting the traffic. Session-level telemetry such as login intensity, failure ratios, source reputation, off-hours access, and atypical size/duration of traffic segments expose patterns that separate hostile activity from nominal usage of the network. I chose to evaluate three models (technically, scikit-learn estimators): two supervised classifiers, Gradient Boosting and an MLP, and one unsupervised anomaly detector, namely, Isolation Forest.

Isolation Forest (IF) offers a no-label anomaly baseline, Gradient Boosted (GB) Trees provide a strong tabular classifier with robust performance with a strict false-positive budget while a compact Multi-Layer Perceptron (MLP) serves as a neural comparator. My focus is practical detection at very low False Positive Rates (FPR) to limit SOC fatigue, while aligning with recent work that emphasizes behavior over content when traffic is encrypted [5]. These three were chosen to cover the operational constraints, data shape, and label realities of a SOC. GB produces monotonic scores you can threshold to exact FPR budgets (0.1–1%), runs fast on CPU, and exposes feature importances for sanity checks. It's the strongest, most reliable supervised baseline for tabular NDR. Where trees make axis-aligned splits, an MLP can model smoother contours and comparing with GB it guards against model-specific blind spots. MLP is also easy to threshold and deploy in that training/inference are lightweight. On the other hand, SOCs don't always have clean labels and IF shows what's achievable without supervision.

Other models considered but not chosen:

- Logistic regression / linear SVM: too linear for the interactions that matter here and typically weaker at low FPR.
- kNN / Naive Bayes: can be memory-heavy (important in NDR which needs high line speed throughput) and slow at inference (kNN) or overly simplistic assumptions (NB).
- Random Forest: good, but GB usually yields steeper ROC near the y-axis (better at ≤1% FPR).
- CatBoost/LightGBM/XGBoost: excellent alternatives and close cousins of GB. Chose GB to represent that family of models.
- Autoencoders/OC-SVM/sequence models: useful for future patterns, but they require more tuning, careful calibration and perhaps more features. Possible future work.

II. RELATED WORK

Tree-based ensembles and one-class methods are strong approaches for tabular anomaly detection. Isolation Forest offers a fast, label-free technique for outlier isolation [1]. Gradient-boosted trees are a proven classifier for heterogeneous features and tight operating points [2]. Recent work has been done in deep and self-supervised models over packet/flow sequences, including contrasting learning for encrypted traffic [3][5]. I adopt a tuned Gradient Boosting model as the primary

supervised baseline on behavioral telemetry, use Isolation Forest where labels are scarce, and include a compact MLP as a neural comparator. Again, my emphasis is low-FPR operation suitable for SOC workloads.

III. APPROACH

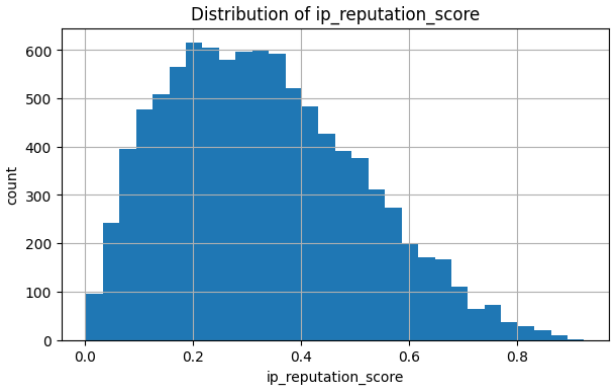
A. Dataset

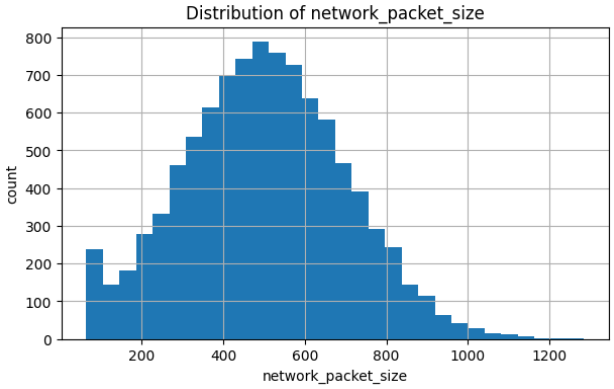
The file I used, cybersecurity_intrusion_data.csv, contains 9,537 records (at time of analysis) of network activity with a unique session ID (session_id) and 10 features used for intrusion detection. It includes attributes like packet size, protocol type, login attempts, session duration, encryption type, IP reputation score,failed logins, browser type, unusual time access, and a binary attack detection label (attack_detected). It is designed for detecting cyber intrusions based on network traffic and user behavior. It was created by Dinesh Naveen Kumar Samudrala in 2025 and posted on Kaggle. [6] At this point in time, for this study, synthetic features were not considered.

Performing Exploratory Data Analysis (EDA), histograms of the numeric variables show distinct shapes that inform the feature engineering and modeling phases. session_duration is strongly right-skewed (heavy tail) with the median about 556 seconds. The 90th and 99th percentiles near 1,796 and 3,660 seconds and a maximum around 7,190 seconds—indicating a small fraction of very long sessions.

network_packet_size is comparatively symmetric around a median near 499 bytes (inter-quartile range roughly 365–635) with a thinner tail up to ~1,285 bytes. login_attempts and failed_logins are a bit right-skewed (medians ~4 and ~1, respectively), with most sessions exhibiting few attempts but a visible tail that shows brute-force pressure. ip_reputation_score concentrates in the lower-to-mid range (median ~0.315; 90th percentile ~0.58; 99th percentile ~0.77) indicating that truly high-risk sources are rarer but present. unusual_time_access is binary and imbalanced toward normal hours (about 1,430 positives vs. 8,107 negatives) making “off-hours” a minority signal. The class label attack_detected is reasonably balanced for supervised learning (0: 5,273; 1: 4,264).

protocol_type is dominated by TCP (~6,624 rows) then UDP (~2,406) and ICMP (~507), consistent with enterprise traffic mixes. encryption_used is mostly AES (~4,706) and DES (~2,865), with a meaningful “None” segment (~1,966), which helps the model discriminate encrypted versus unencrypted behaviors. browser_type is led by Chrome (~5,137), then Firefox (~1,944) and Edge (~1,469), with smaller counts for Safari and Unknown. Together, these histograms suggest a feature space where attacks can manifest as elevated authentication activity, risky source reputation and off-hours access often within common TCP traffic. Signals therefore map naturally to brute-force login patterns (login_attempts, failed_logins), suspicious sources (ip_reputation_score), unusual access times (unusual_time_access), and atypical session sizes/durations (network_packet_size, session_duration) that the models can utilize.





B. Performance Metrics

I report Accuracy, PD (Probability of Detection) / TPR (True Positive Rate), and PFA (Probability of False Alarm) / FPR (False Positive Rate) with emphasis on low-FPR (False Positive Rate) which are operating points suitable for SOC workflows. ROC-AUC (Area Under the Receiver Operating Characteristic Curve) is computed where applicable. The ROC–AUC confidence intervals

are 2,000-rep stratified bootstraps and ROC is zoomed to FPR ≤ 1%. Isolation Forest scores are oriented so higher = more anomalous.

C. Methodology

I preprocess features by separating numeric and categorical fields then applying transformations fit only on the training split to avoid data leakage. The split is 70% train / 30% test. Total samples used 9,537. Numeric variables are scaled (for example, standardized to zero mean and unit variance) to stabilize optimization for models that are sensitive to feature magnitude. Categorical variables (such as protocol and encryption indicators) are encoded into a machine-readable form (one-hot encoding). Using a single "class number" (ordinal/int encoding) would inject a pseudo-ordering and put distance between categories (e.g., UDP < TCP < ICMP) that none of those categories actually have. That can hurt both tree model's splits and the MLP's learning.

Identifier fields like session_id are excluded from learning and any missing values are handled before modeling so downstream estimators receive clean data. They are valueless features thus the identifiers are excluded by column selection + ColumnTransformer. In the feature engineering cell I build an explicit feature_cols list that does not include session_id.

The full preprocessing and modeling steps are composed into a single pipeline so that inference applies the same transformations consistently. I then train three detectors: Isolation Forest (unsupervised anomaly detection that isolates outliers via random splits), Gradient Boosted Trees (GBT - a Gradient Boosting Classifier) and a Multi-Layer Perceptron (MLP) neural network. A time-based split simulates forward-in-time generalization so that models are trained on earlier data, tuned on a later validation slice and evaluated on the most recent test slice. Decision thresholds are calibrated on the validation set to meet a target FPR, typically ≤1 percent, yielding operating points that balance sensitivity and alert volume. For supervised models (GBT, MLP), I select the probability cutoff that attains the desired FPR and for Isolation Forest I tune the anomaly-score threshold analogously. Final reports include PD/TPR, FPR, accuracy, ROC-AUC and confusion matrices at the chosen operating points. [2] [4]

IV. EXPERIMENTAL EVALUATION

A. Experimental Setup

Experiments run on Windows 10 with Visual Studio Code (VSCode) and the Jupyter extension. Python 3.11 executes in a dedicated virtual environment. Core libraries: scikit-learn (Isolation Forest, Gradient BoostingClassifier, MLPClassifier), NumPy, pandas, and matplotlib. Randomness is controlled via NumPy seeds and estimator random_state to reduce variance. The notebook itself is in the file network_anomaly_pipeline_v9.ipynb and output from the notebook can be viewed in the HTML file network_anomaly_pipeline_v9.html. All are included with this paper's submission. The Jupyter kernel executed inside VS Code, enabling cell-by-cell runs, variable inspection, and inline rendering of tables and charts. Project files (the notebook, dataset CSV, and exported metrics) were stored locally and outputs such as metrics\host.csv, metrics\time.csv and per-model predictions were written to a designated outputs directory. Random seeds were fixed via both NumPy and estimator random_state settings to reduce run-to-run variance.

For supervised models, decision thresholds are calibrated on the validation slice to hit target low-FPR operating points suitable for SOC use. For supervised models, decision thresholds are calibrated on the validation slice to define strict FPR operating points suitable for SOC use. Confusion matrices and table metrics shown in this paper are computed at the default 0.5 threshold; calibrated operating points are presented separately in ROC/PR analyses. Confusion matrices shown in Tables II–III reflect default 0.5 thresholds; calibrated cutoffs for strict FPR budgets are reported separately in ROC/PR plots and operating-point summaries.

B. Experimental Procedure

The notebook follows a standard ML workflow:

1. EDA: load the labeled session-telemetry CSV and inspect types/nulls. Review distributions for login_attempts, failed_logins, session_duration, network_packet_size,

ip_reputation_score, protocol_type, browser_type, and unusual_time_access. Confirm that features describe "network-centric" behavior rather than protocol payloads. A key EDA (Exploratory Data Analysis) focus is class balance in the attack_detected label and quick sanity checks for outliers or obviously leaky fields. This phase establishes which variables are numeric vs. categorical and confirms that the features align with "network-centric" behaviors (brute force login attempts, suspicious timing, and source reputation) rather than protocol/port specifics.

2. Preprocessing: split numeric vs. categorical, scale numeric, one-hot encode the categories, drop identifiers (e.g., session_id) and impute missing values. Then the data is split with a time-based split (and a host-agnostic split for comparison).

3. Training: Three models are trained—Isolation Forest (unsupervised baseline), Gradient Boosting Classifier, and MLP Classifier (Multi-Layer Perceptron Classifier)—and thresholds for the supervised models are calibrated to keep FPR/PFA (False Positive Rate/Probability of False Alarm) low.

4. Thresholding: On the validation slice, choose cutoffs to meet FPR budgets (≤1%). For Isolation Forest, tune the anomaly-score threshold in a similar manner.

5. Evaluation/Export: Evaluate compute accuracy, TPR/recall, FPR, ROC-AUC and confusion matrices on both host-based and time-based tests. Then, export metrics_host.csv, metrics_time.csv, and per-model predictions. This setup makes comparisons at realistic operating points reproducible [2][4].

V. RESULTS AND DISCUSSION

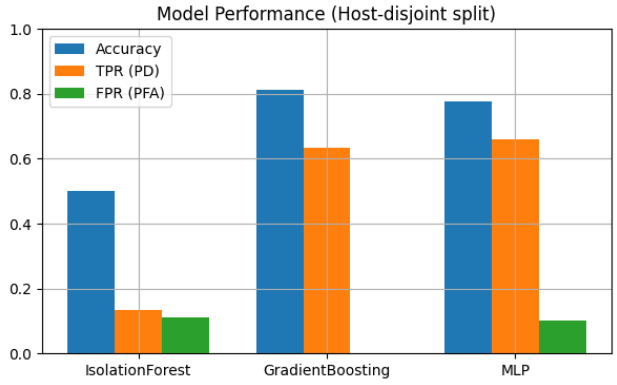


Figure 1 and Table I. Host-Based Test Split Metrics

model	*accuracy*	*tpr*	*fpr*
Isolation Forest	*0.501776*	*0.132673*	*0.110302*
Gradient Boosting	*0.812278*	*0.633663*	*0.000000*
MLP	*0.776256*	*0.659406*	*0.100937*

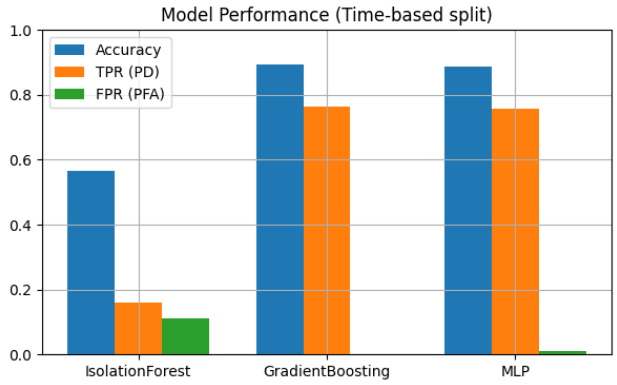


Fig. 2 and Table II. Time-Based Test Split Metrics. Only Gradient Boosting met the ≤1% FPR budget in practice; the MLP's FPR of 1.19% was above the strict target.

model	*accuracy*	*tpr*	*fpr*
Isolation Forest	*0.566737*	*0.161417*	*0.109925*
Gradient Boosting	*0.894479*	*0.764567*	*0.001884*
MLP	*0.886094*	*0.758268*	*0.011935*

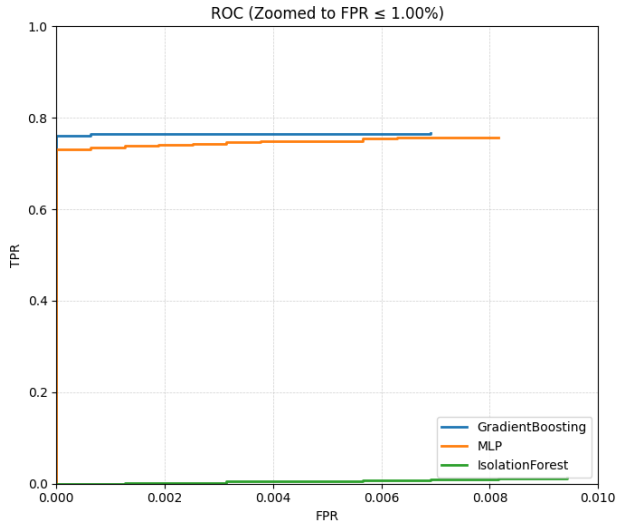


Fig. 3. ROC curves zoomed to the low-PFA regime (FPR ≤ 1%). Gradient Boosting maintains materially higher PD/TPR than MLP at strict false-positive budgets; Isolation Forest underperforms in this regime. Curves computed on the forward-in-time test slice.

Precision	**Recall (TPR)**	**F1**	**F1 (macro)**	**Support (pos)**
Model				
Gradient Boosting	0.9969	0.7646	0.8654	0.8893	1270
MLP	0.9807	0.7583	0.8552	0.8807	1270
Isolation Forest	0.5395	0.1614	0.2485	0.4721	1270

Table III. Precision, Recall, and F1 on the Time-Based Test Slice

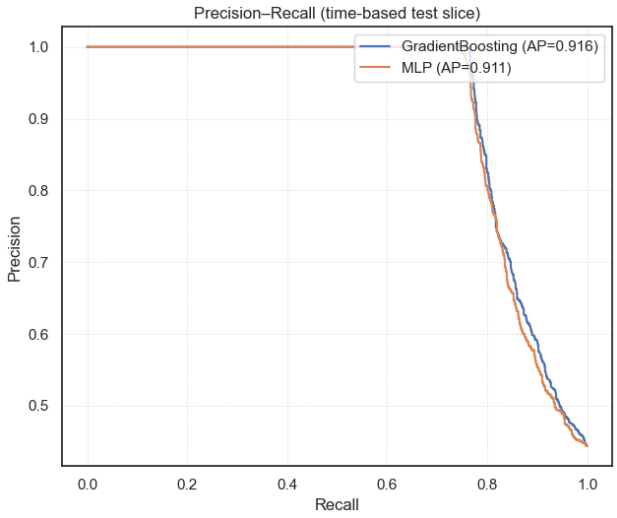


Fig. 4. Precision–Recall curves (time-based test slice); Precision = PPV and Recall = TPR/PD

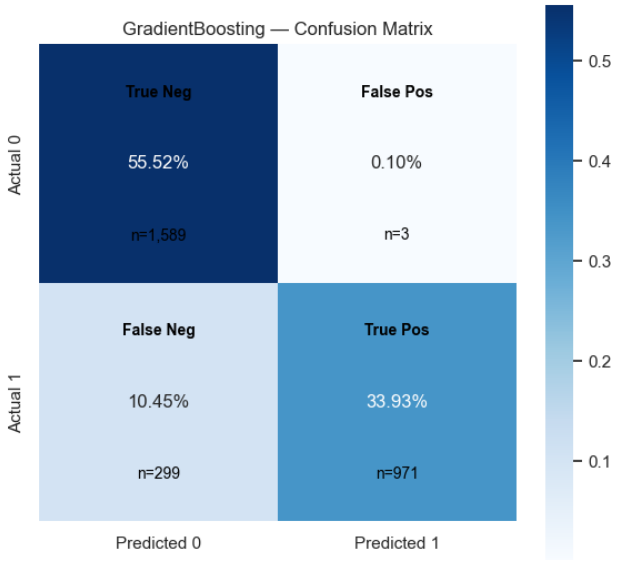


Fig. 5. Gradient Boosting confusion matrix on the time-based test slice at a validation-calibrated threshold (≈0.2% FPR). Counts: TN=1,589, FP=3, FN=299, TP=971; Recall (TPR) ≈ 0.765, Precision (PPV) ≈ 0.997, F1 ≈ 0.865.

Key observations:

On the time-based test slice, Gradient Boosting gives me the best balance for real SOC constraints. Beyond the tabular metrics already shown (Accuracy = 0.894; PD/TPR = 0.765; PFA/FPR = 0.0019), I now report ROC–AUC with 95% CIs and a zoomed ROC (FPR ≤ 1%). Gradient Boosting achieves AUC = 0.889 (95% CI:

0.876–0.903) and sustains ~0.765 PD/TPR across strict false-positive budgets from 0.1% → 1.0% PFA (e.g., TPR ≈ 0.7647 @ 0.1%, 0.7651 @ 0.2%, 0.7655 @ 0.5%, 0.7660 @ 1.0%). In operational terms, those budgets correspond to roughly 5.6, 11.1, 27.8, 55.6 false alarms per 10k sessions at the observed class mix while maintaining high precision (PPV ≈ 0.9984 → 0.9839) as the budget relaxes. The MLP tracks close in rank (AUC = 0.881; 95% CI: 0.866–0.895) and recall (~0.74–0.76 across the same FPR targets) but costs more noise at comparable detection. As a label-free baseline, Isolation Forest under-detects in this feature space (AUC = 0.573; 95% CI: 0.552–0.594) and is essentially non-operational at very low PFA (e.g., TPR ≈ 0.0000–0.0161 from 0.1–1.0%), which is consistent with its behavior on the tabular metrics I already show.

The zoomed ROC makes the same point visually: in the ≤1% PFA window the boosted model's curve sits highest, maintaining materially higher PD than MLP at the same false-positive budgets, while the unsupervised baseline lags. These observations align with the paper's emphasis on low-FPR operation for SOC workflows and the evaluation plan I stated earlier (operating-point reporting, ROC-AUC, and confusion matrices), see Fig 5. They also square with the broader literature cited in this work: Gradient-boosted trees are well-suited to tight operating points on heterogeneous tabular features [2], while Isolation Forest provides a fast label-free comparator [1]; and in encrypted/opaque traffic, behavioral signals tend to yield better operational traction than payload content, especially at strict PFA budgets [5].

In line with the ROC analysis, the precision–recall view reinforces the operational story. On the time-based slice, Gradient Boosting achieves Precision ~0.9969 with Recall ~0.7646, yielding F1 ~0.8654 (macro-F1 ~0.8893; positive support = 1,270). MLP posts similar recall (~0.7583) but with lower precision (~0.9807), resulting in F1 ~0.8552 (macro-F1 ≈ 0.8807). Isolation Forest trails substantially (Precision ~0.5395, Recall ~0.1614, F1 ~0.2485). The PR curves in Fig. 4 visualize this gap: GB maintains high precision across the recall range relevant to a low-PFA workflow, while MLP trades additional false alerts for comparable recall and the label-free baseline under-detects. Looking at Precision (PPV) and Recall (TPR/PD), these results directly translate to analyst experience at strict FPR budgets. This complements the AUC and zoomed-ROC evidence presented above.

VI. CONCLUSIONS AND FUTURE WORKS

Conclusions:

Behavioral telemetry is effective for anomaly-based detection when payloads are encrypted or obfuscated. Under time-based evaluation, Gradient Boosting provides the best PD–FPR balance among the tested models, sustaining very low FPR with useful recall—a practical profile for production SOC workflows.

Operational guidance.
- Calibrate thresholds at multiple operating points (0.1%, 0.5%, 1% FPR) and select per use case.
- Monitor feature and label drift as well as retrain on a calendar or drift trigger.
- Integrate scores with SIEM/SOAR to pair model outputs with simple guardrails (e.g., hard blocks on extreme IP-reputation, surge-based authentication limits).
- Track alert volume and case closure outcomes to refine thresholds and feature importances.

Future work.
1. Sequence models (TCN, Transformer) over rolling session windows to capture temporal dependencies.
2. Self-supervised pretraining to reduce dependence on labels and improve robustness [5].
3. Adaptive thresholding (e.g., extreme-value theory) for tail-aware alerting under rate shifts.
4. Context enrichment: user/device baselines, network geolocation/ASN, and environmental signals to harden against "slow-and-low" tactics.

In conclusion, the results show that well-engineered behavioral features combined with Gradient Boosting can deliver high PD at very low FPR, which is crucial for production SOC workflows. Looking ahead, adding sequence-aware models (TCN/Transformer), user/device-adaptive baselines, and ASN/geolocation-aware enrichment should improve robustness against slow-and-low attacks while preserving the low-alert-volume characteristics demonstrated here.

So, while the Gradient Boosting detector I selected is trained offline on behavioral features and then thresholded for low-PFA operation, its philosophy is aligned with modern NDR. Current NDR platforms prioritize behavior over signatures to surface abnormal

activity in encrypted, heterogeneous traffic. What distinguishes many state-of-the-art platforms (e.g., Darktrace and ExtraHop) is the breadth of wire-data visibility and the learning mode. Darktrace emphasizes self-learning, unsupervised modeling of each entity's "pattern of life" with autonomous response, so detection is expressed as deviations from baselines rather than a supervised risk score [7]. ExtraHop Reveal(x) pairs machine learning with deep L7 protocol decoding and curated detectors to drive investigation workflows and response which are capabilities beyond my tabular feature set [8]. In short, my model is similar in its behavioral orientation and low-false-positive prioritization but dissimilar in scope (single supervised detector vs. multi-sensor NDR stack), learning approach (supervised vs. predominantly self-learning/hybrid), and response (thresholded alerts vs. autonomous triage/containment). For positioning, this makes my detector a focused, lightweight component that can augment or feed an NDR/SIEM pipeline, while full NDR platforms provide end-to-end visibility, correlation, and automated response as well as real-time line-decryption at high speeds and an ability to block malicious traffic upon detection[9].

REFERENCES

[1] F. T. Liu et al., "Isolation Forest" , pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[2] J. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics, 29(5):1189–1232, 2001. DOI: 10.1214/aos/1013203451.

[3] I. Goodfellow et al., "Deep Learning," MIT Press, 2016 (ISBN 978-0262035613).

[4] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection,", IEEE S&P 2010, pp. 305–316. DOI: 10.1109/SP.2010.25.

[5] R. Xie et al., "Enabling robust TLS encrypted traffic classification in the wild via self-supervised contrastive learning," in Proc. USENIX Security Symp., 2023, pp. 1279–1296.

[6] Cybersecurity Intrusion Detection Dataset, Dinesh Naveen Kumar Samudrala, 2025

[7] Darktrace, "**AI Network Security Protection (Darktrace / NETWORK)**."

[8] ExtraHop Networks, "**Powered by the RevealX™ Platform**."

[9] Gartner Peer Insights, "**Best Network Detection and Response Reviews 2025**"