

An Analysis of Artificial Intelligence-Based Predictive Models for Improving Data Security and Decision-Making in Cloud Computing

Dr. Rajesh Kumar Sharma

Associate Professor

Department of Computer Science and Engineering,
RKDF University Bhopal, Madhya Pradesh, 462036

rajeshsharma.ercs@gmail.com

Abstract

Cloud services provide vast amounts of security telemetry, but the scale of alerts and the asymmetric impact of missed threats impede quick action. In this work we explore the ability of supervised prediction models to classify botnet flows from benign flows in a public AWS-hosted testbed and propose a human-in-the-loop alarm triage architecture. A derivative of CSE-CIC-IDS2018 with 48,000 records was created and split according to the provided training and testing blocks of 24,000 records each with 500 bot and 23,500 benign flows. We removed the label and the provenance index to be left with 78 numeric flow predictors. Median imputation and model specific scaling were fitted on training data. We evaluated majority-class, class-weighted logistic-regression, random-forest, and histogram-gradient-boosting classifiers on the original test labels. With a fixed threshold of 0.50, histogram boosting yielded 497 true positives, 3 false negatives, 2 false positives and 23,498 true negatives. The accuracy was 0.9960, the recall was 0.9940, the F1 was 0.9950, and the average precision was 0.9978. This model achieved an F1 of 0.9927 and average accuracy of 0.9961 by removing 874 test records with an identical feature-and-label signature of training records. These findings indicate a high distinction for this limited bot vs. benign benchmark. The triage bands provided indicate how scores may be used to organise analyst evaluation, although neither decision quality nor protection of actual cloud assets was tested. Still need External validation, calibration and future incident-response assessment.

Keywords: Cloud computing; network intrusion detection; botnet traffic; predictive modeling; data security; decision support; class imbalance; CSE-CIC-IDS2018.

1. INTRODUCTION

1.1 Cloud computing and data security challenges

Cloud computing pools customisable computer resources and exposes on demand access, flexibility and measurable service (Mell & Grance, 2011). The same connection that enables scaled services also creates diversified traffic, scattered assets and fast-changing operating baselines. Governance, data protection, access control, and monitoring are some of the ongoing problems in the public-cloud security advice (Jansen & Grance, 2011). Similarly, the Cloud Security Alliance (2024) records prominent cloud threat categories, which are context, not labels in the current dataset.

A single benign or malicious flow is a far smaller observation than a “secure cloud” Network intrusion prediction may assist to filter a stream of events, but cannot detect a compromised identity, establish exfiltration, or evaluate privacy compliance without further proof. Detection and response are components of a broader security programme in the NIST control catalogue (National Institute of Standards and Technology [NIST], 2020). These borders are important to read benchmark scores.

1.2 Role of artificial intelligence in cloud environments

We use supervised learning to relate the observed flow features to a bot vs benign score. Linear and tree-based models use distinct assumptions: logistic regression adds processed signals, random forests average numerous decorrelated trees, boosting combines trees sequentially (Breiman, 2001; Friedman, 2001). Intrusion detection based on machine learning has generated much interest, but adoption must deal with labels, drift, analyst trust and false alarms (Buczak & Guven, 2016; Sommer & Paxson, 2010). Here “artificial intelligence” is meant in its traditional machine-learning meaning. It does not promise autonomous incident reaction.

1.3 Need for predictive models

Not feasible to manually evaluate all flows at scale. A ranking mechanism may move possibly harmful data to the head of an analyst queue. Bot flows only account for 2.083% of each benchmark block, hence accuracy alone hides missed assaults. A model that predicts benign everywhere achieves 97.917% accuracy but finds no bots. In particular, precision–recall assessment is effective when the incidence of attacks is low (Davis & Goadrich, 2006; Saito & Rehmsmeier, 2015).

1.4 Objective of the study

1. The primary objective is to compare four supervised baselines on an explicitly specified bot-versus-benign subset of CSE-CIC-IDS2018.
2. Secondary objectives are to measure false-alert burden and missed bots at a fixed threshold; examine exact duplicate signatures across the split; describe a transparent score-to-triage process; and release executable code sufficient to regenerate each reported result.

The study does not estimate a change in breach incidence, analyst productivity, or enterprise decision quality.

Table 1. Research questions, observable evidence, and limits

Question	Evidence in this study	Interpretive limit
Can models detect benchmark bot flows?	Confusion matrices, recall, AP	One attack family and testbed
How many false alerts arise?	FP counts and precision	Prevalence fixed by selected subset
Does exact signature overlap matter?	Novel-row sensitivity subset	Other leakage modes remain
Can scores support triage?	Illustrative score bands	No analyst outcome measured

2. MATERIALS AND METHODS

2.1 Dataset and Preprocessing

The original CSE-CIC-IDS2018 dataset was created on an AWS-based testbed and comprises benign traffic and seven attack scenarios (Canadian Institute for Cybersecurity [CIC], 2018; Sharafaldin et al., 2018). For this investigation we employ the third-party public “bot versus benign, preserved ratio, train500minority, 23500benign, test500minority, 23500benign, chrono” CSV format (rayborg, n.d.). It's a selected subset, not the whole original corpus. The first 24,000 rows of the table are the offered training block and the following 24,000 are the provided testing block. Each block comprises 23,500 benign samples and 500 bot samples. Within blocks the source row indices do not overlap. Within each class the source indices advance but the two classes are in separate ranges of source indices. The split cannot be stated as a globally time ordered prospective test.

The source csv comprises 80 columns: binary label, source_row_index and 78 numeric predictors. No other label encoding or feature selection was done. We eliminated the provenance index as it may convey ordering and collection context. Missing values were used in place of positive and negative infinity. In the whole file there were 330 missing items for “flowbyts/s” and “flowpkts/s”. Missing values were imputed using solely median imputation on the training data, with a missingness indication added for each feature impacted. Furthermore, logistic regression employed standardisation solely on training. All estimators employed the precise provided split, no held-out row was fitted to any preprocessing transform (Pedregosa et al., 2011).

Table 2. Dataset and split specification

Property	Value	Verification
Source	CSE-CIC-IDS2018 bot-versus-benign derivative	Public benchmark file
Total records	48,000	CSV row count
Input columns	78 numeric	80 total less label and index
Train	23,500 benign; 500 bot	Rows 0–23,999
Test	23,500 benign; 500 bot	Rows 24,000–47,999
Test bot prevalence	2.0833%	500 / 24,000
Missing rate fields	330 nulls in each rate column	Input inspection

The original corpus includes other event and host logs, however the derivative utilised here only has flow statistics. The “preserved ratio” designation specifies the benchmark construction. It does not show that 2.083% is the traffic in a genuine production environment. We keep that ratio, and do not use synthetic oversampling. Other resampling approaches such as SMOTE need a distinct train-only methodology (Chawla et al., 2002; He & Garcia, 2009).

2.2 Selection of Security and Decision-Making Features

The 78 predictors include packet counts and lengths, duration and inter-arrival times, flow rates, TCP flag counts, header lengths, ports, window widths and active/idle intervals. Traffic descriptions These are These are not measures of secrecy, user intent, monetary exposure or quality of management choices. To enable fast replication, numeric benchmark characteristics were preserved

for port and protocol values, although this made ordinal interpretation of port numbers incorrect. No causal feature interpretation is claimed.

Table 3. Examples of available predictor groups

Group	Example CSV field	Operational reading
Endpoint context	dstport, protocol	Service/protocol context
Volume	totfwdpkts, totbwdpkts	Packet volume by direction
Size	pktlenmean, pktlenstd	Packet size summary
Timing	flowduration, flowiatmean	Duration and spacing
Rate	flowpkts/s, flowbytes/s	Activity rate; missingness possible
Flags	synflagcnt, ackflagcnt	Observed TCP flag counts
Session	idlemean, activemean	Activity and idle timing

2.3 Artificial Intelligence-Based Predictive Models

The floor is provided by a majority-class dummy estimator that always predicts innocuous. Logistic regression gives a clear linear baseline with balanced class weights and a maximum of 1,500 iterations. The random forest has 120 trees, minimum two samples per leaf and balanced subsample class weighting (Breiman, 2001). Histogram gradient boosting using 120 iterations, max 15 leaves, learning rate 0.07 and default loss (Friedman, 2001). The scikit-learn implementation we used is not one of LightGBM or XGBoost; they are separate systems and recommended as future comparisons (Chen & Guestrin, 2016; Ke et al., 2017).

Table 4. Prespecified estimator configuration

Estimator	Selected parameters	Preprocessing
Majority	Prior class	None
Logistic regression	Balanced weights; max_iter=1,500	Median + indicators + standardization
Random forest	120 trees; leaf=2; balanced_subsample	Median + indicators
Histogram boosting	120 iterations; 15 leaves; lr=.07	Median + indicators

Each estimator was fit once on the specified training block with random_state=42 where supported. Classification was based on the usual fixed criterion $p \geq 0.50$. Parameter values are implementation options, not results of a search across the held-out block. The boosted classifier is not given any explicit class weighting and hence the comparisons described in this paper are reflective of these unique configurations and not equally tuned families of algorithms.

2.4 Proposed Predictive Framework

The architecture is comprised of two layers: an assessed flow classifier and a conceptual analyst workflow. An input flow is parsed into the 78 needed predictors, checked against the training schema, pre-processed using stored transformations and scored. 0.00-0.09 scores are flagged monitor. 0.10-0.49 scores are flagged analyst review. 0.50+ scores are flagged investigate. The use of thresholds illustrated a basic, not an optimal choice strategy. Log records and decisions Protect under least privilege Link to incident evidence and response processes (NIST, 2011, 2025).

Table 5. Proposed framework stage and its evidence

Stage	Input → output	Status
Collection	Flow record → validated numeric schema	Conceptual integration
Prediction	Schema → model score	Implemented offline
Queueing	Score → monitor/review/investigate	Illustrated retrospectively
Analyst	Case evidence → final determination	Not measured
Monitoring	Confirmed outcomes → calibration/drift audit	Future deployment

The programme prints out a model artefact, and a table of triage counts. Model scores are statistical outputs and need to be calibrated before a figure such as 0.80 can be interpreted as an 80% attack probability (Guo et al., 2017). The process needs a human to vet context, before confinement or other expensive action; NIST's AI RMF offers a governance structure for such risk-managed deployment (NIST, 2023).

2.5 Performance Evaluation Metrics

TP: identified bot flows, FN: missed bot flows, FP: benign flows flagged, TN: benign flows cleared with bot as the positive class. Specificity = $TN/(TN+FP)$ $F1 = 2 * precision * recall / (precision + recall)$ Balanced accuracy = $(recall + specificity) / 2$. Average precision (AP) summarises a precision-recall curve across ranking criteria. ROC AUC summarises ranking over false positive and true positive rates (Fawcett, 2006; Saito & Rehmsmeier, 2015).

The main descriptive metrics are AP, F1, recall and FP count as both rankings and operational false alarms are important. These measures are conditioned on the benchmark labels being accurate. The article provides accurate counts as well as rounded rates to avoid misinterpretation when denominators vary. No confidence interval is provided because the presence of exact or near-duplicate flows weakens an interpretation based on independent observations; prospective sampling would be required for operational uncertainty.

3. MODEL DEVELOPMENT AND IMPLEMENTATION

3.1 Model Training and Testing

The distributed CSV repaired the training & test blocks. The script verifies the 48,000 row count, binary labels, 23,500/500 composition in each half, uniqueness of indexes within each half, and numeric predictor schema. Prints missing-value statistics before fitting the model. Imputation, scaling and estimator parameters are bundled in separate scikit-learn Pipeline objects; fitted pipelines are serialised as .joblib artefacts. scikit-learn has its serialisation of fitted objects adapted to the trusted process, and model files should only be imported from a trusted source.

A second test discards test records with the precise predictor-and-label signature that was seen in training. This leaves us with 23,126 left, 343 bot and 22,783 benign. Repeated measurements may generate similar vectors even if the source_row_index values are not overlapping. This sensitivity analysis is one particular kind of redundancy and cannot remove shared host profiles, batch effects, label artefacts, or distribution shift (Sommer & Paxson, 2010).

Table 6. Split and sensitivity diagnostic

Diagnostic	Result	Interpretation
Source-index overlap	0	Rows have distinct provenance indexes
Exact feature+label test overlap	874 / 24,000 (3.64%)	Repeated signatures occur
Novel-row sensitivity set	23,126 records	Subset changes attack prevalence
Novel-row bot count	343	157 overlapping bot signatures removed
Novel-row benign count	22,783	717 overlapping benign signatures removed

3.2 Python-Based Implementation

See the accompanying cloud_security_models.py for a complete runnable implementation. The major data and pipeline phases are shown in Listing 1. The rest of the script generates all model metrics, seven graphs, the sensitivity table, the triage table and serialised models. Python dependencies: pandas, numpy, matplotlib, scikit-learn, joblib. Versions should be documented when rerunning. The interpreter on the confirmed run utilised pandas 2.2.3 and scikit-learn 1.8.0.

Listing 1. Main preprocessing and model specification (excerpt; complete code supplied separately).

```
df = pd.read_csv(args.data).replace([np.inf, -np.inf], np.nan)
fields = df.drop(columns=['label', 'source_row_index'])
Xtrain, Xtest = fields.iloc[:24000], fields.iloc[24000:]
ytrain, ytest = df.label.iloc[:24000], df.label.iloc[24000:]
models = {
    'Majority baseline': DummyClassifier(strategy='prior'),
    'Logistic regression': Pipeline([
        ('impute', SimpleImputer(strategy='median', add_indicator=True)),
        ('scale', StandardScaler()),
        ('model', LogisticRegression(max_iter=1500,
            class_weight='balanced', random_state=42))),
    'Random forest': Pipeline([
        ('impute', SimpleImputer(strategy='median', add_indicator=True)),
        ('model', RandomForestClassifier(n_estimators=120,
            min_samples_leaf=2, class_weight='balanced_subsample',
            random_state=42, n_jobs=-1))),
    'Histogram boosting': Pipeline([
```

```
('impute', SimpleImputer(strategy='median', add_indicator=True)),
('model', HistGradientBoostingClassifier(max_iter=120,
max_leaf_nodes=15, learning_rate=.07, random_state=42)))
}
```

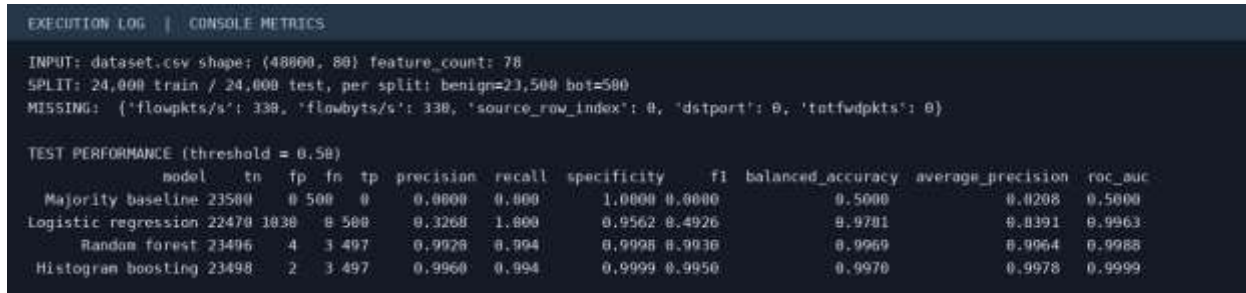


Figure 1. Faithful visual rendering of the recorded program execution log, including input checks and test metrics

The model generates a score for each held out flow. Predicted class is 1 if score $\geq .50$ otherwise 0. Running the programme once with the provided CSV will reproduce all figures and tables in Section 4. If a different 2018 extract is mistakenly entered, the printed validation tests fail purposefully.

3.3 Security Risk Prediction and Decision-Making Process

Prediction and operational decision, they are different yet connected. The experiment outputs a classifier score and benchmark confusion matrix. The proposed decision method translates the score of the selected model into an analysis queue. An analyst would need asset criticality, identification events, cloud audit logs and independent verification before acting. Such data are not available here. The triage table is therefore a retroactive representation of numbers rather than an experiment on human decisions.

The example highest AP model is picked following evaluation on the held-out set. This decision is OK for presenting outcomes, but is test-set selection if used to claim a final impartial deployment model. A future research has to hold out a distinct validation split for model choice and policy thresholds and an undisturbed final test set. The score bands are also not calibrated and cannot be interpreted as risk categories with verified probability (Guo et al., 2017).

Table 7. Illustrative action policy and required safeguards

Score band	Suggested analyst action	Required validation before production
< 0.10	Record and monitor	Audit misses and drift
0.10 to < 0.50	Review with related alerts	Measure queue workload
≥ 0.50	Prioritize investigation	Confirm evidence before containment

The NIST incident-reaction guideline portrays response as a lifecycle connected with company risk management, feeding alert scores into that lifecycle with clear ownership and evidence preservation (NIST, 2025). Explainability techniques such as local surrogate explanation and feature attribution

might aid analysts, however the current experiment does not implement or verify explanations (Lundberg & Lee, 2017; Ribeiro et al., 2016).

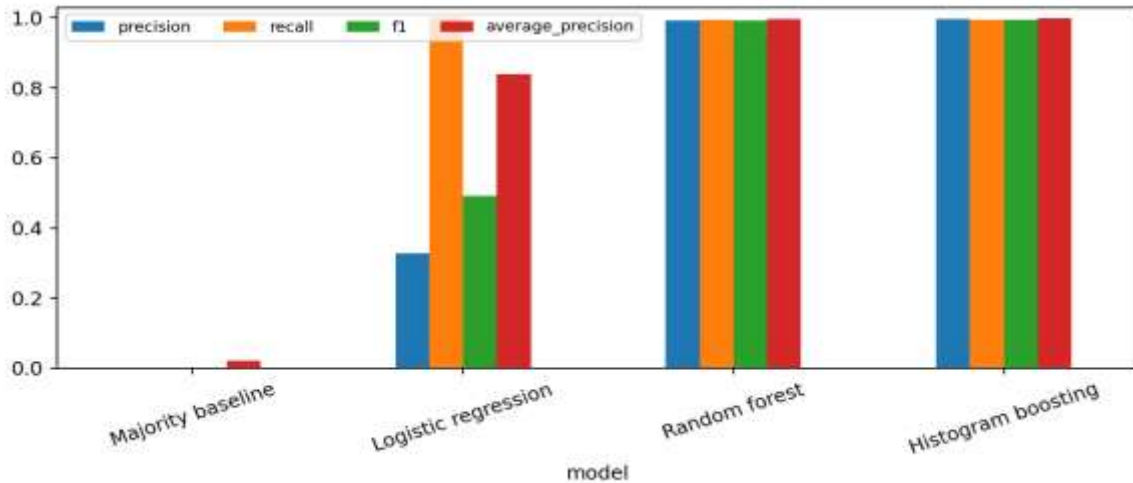


Figure 2. Test-set comparison across precision, recall, F1, and average precision. Generated by the supplied code

4. RESULTS AND ANALYSIS

4.1 Performance of Predictive Models

The majority baseline classified all 24,000 flows in the test set as benign, yielding 23,500 true negatives and 500 false negatives. Its apparent accuracy of 97.92% shows why accuracy is not enough in this class distribution. Logistic regression correctly identified all 500 bots, but had 1,030 false positives (precision 0.3268). The tree ensembles dramatically decreased false alarms. Random woodland gave 497 TP, 3 FN, 4 FP and 23,496 TN. Histogram boosting resulted in 497 TP, 3 FN, 2 FP and 23,498 TN.

Table 8. Held-out test performance at threshold 0.50

Model	Precision	Recall	Specificity	F1	AP	ROC AUC
Majority baseline	0.0000	0.0000	1.0000	0.0000	0.0208	0.5000
Logistic regression	0.3268	1.0000	0.9562	0.4926	0.8391	0.9963
Random forest	0.9920	0.9940	0.9998	0.9930	0.9964	0.9988
Histogram boosting	0.9960	0.9940	0.9999	0.9950	0.9978	0.9999

AP is averaged across all criteria. Histogram boosting achieved AP of 0.9978, random forest 0.9964, logistic regression 0.8391, and majority baseline 0.0208. The AP gap between the two tree models on this synthetic benchmark is merely 0.0014; no significance test is claimed. While ROC

AUCs are presented for completeness, the precision-recall curve makes false-positive behaviour simpler to explain in a rare-positive context (Davis & Goadrich, 2006; Saito & Rehmsmeier, 2015).

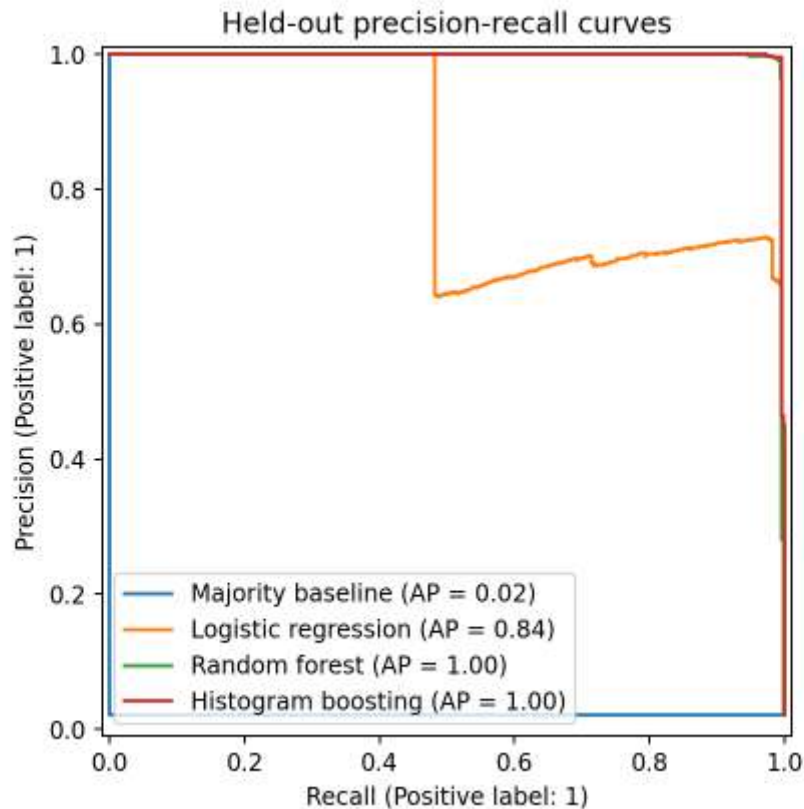


Figure 3. Precision–recall curves on the held-out test block. The baseline AP reflects 2.083% prevalence

4.2 Classification and Prediction Results

Table 9. Complete confusion-matrix counts

Model	TN	FP	FN	TP
Majority baseline	23500	0	500	0
Logistic regression	22470	1030	0	500
Random forest	23496	4	3	497
Histogram boosting	23498	2	3	497

The test confusion matrix of the boosted classifier is shown in Figure 4. This makes the trade-off of errors tangible: 3 of 500 bot flows missed and 2 of 23,500 benign flows elevated. Specificity on these records is 0.9999 and recall is 0.9940. These counts are based on the set threshold value of 0.50. If a different threshold is used, the numbers will vary. A false positive costs analyst time, whereas a false negative may leave a real alarm unreviewed. There are no monetary loss weights available to establish an ideal threshold.

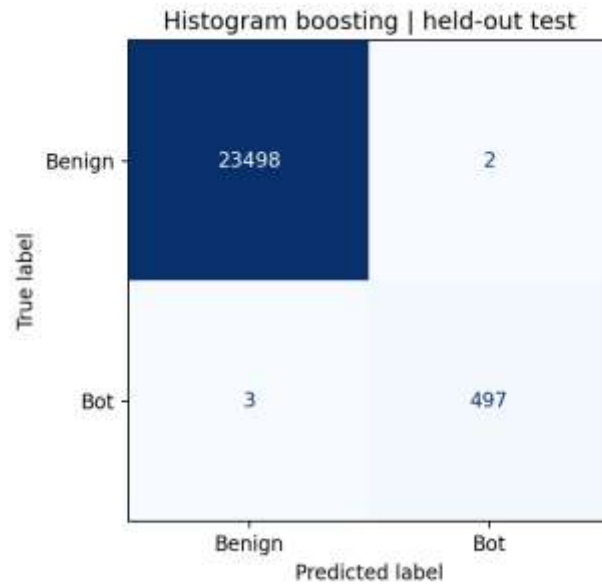


Figure 4. Held-out confusion matrix for histogram gradient boosting (actual program output)

The same boosted model is used to build the prototype triage table. The monitor band has 23,498 benign and 2 bot records, the middle review band has 1 bot record and the investigate band has 2 benign and 497 bot records. The categories may lead to attention but a positive retrospective count does not prove that analysts would uncover occurrences sooner or make fewer expensive errors.

Table 10. Illustrative triage allocation on 24,000 held-out records

Band	Benign	Bot	Total
Monitor (<0.10)	23498	2	23500
Analyst review ($0.10\text{--}0.50$)	0	1	1
Investigate (≥ 0.50)	2	497	499

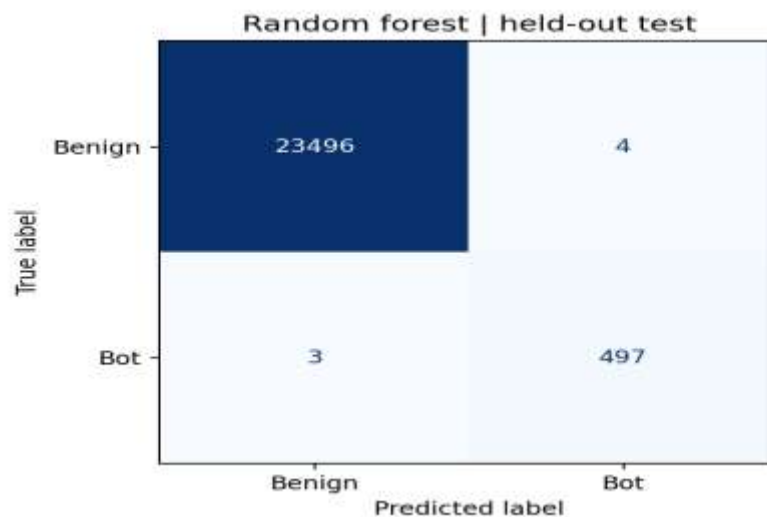


Figure 5. Held-out confusion matrix for random forest

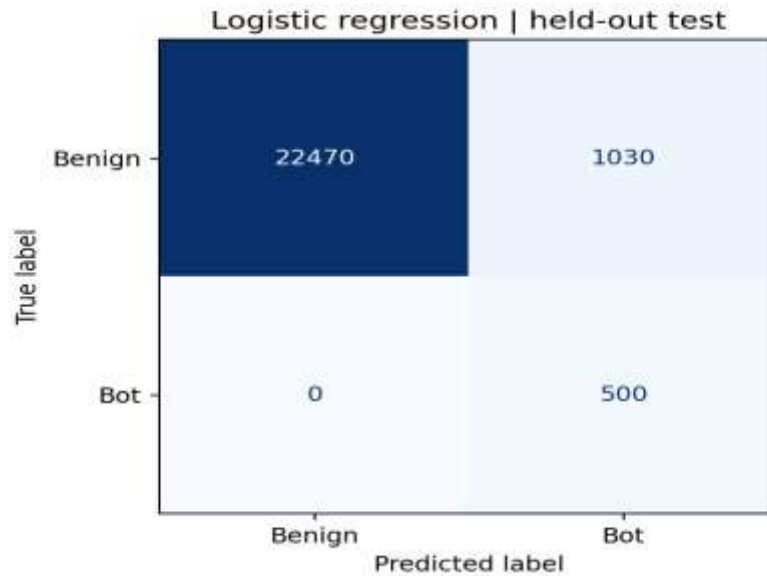


Figure 6. Held-out confusion matrix for class-weighted logistic regression

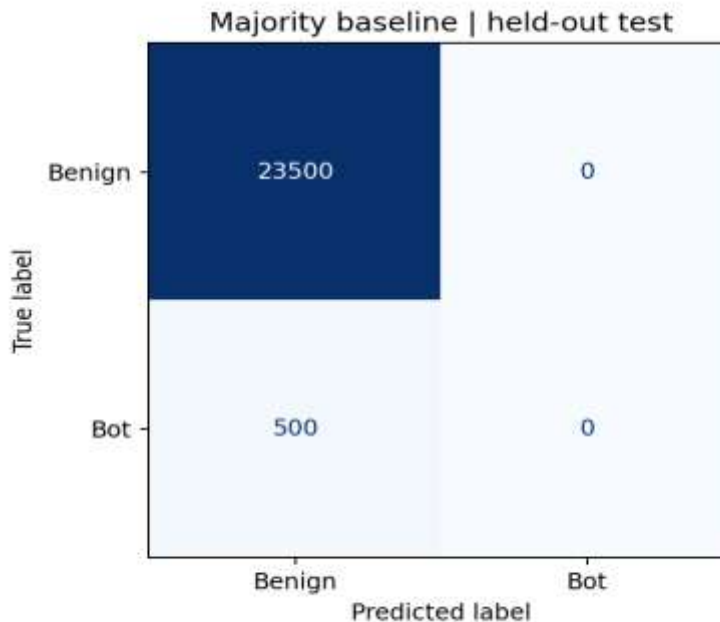


Figure 7. Held-out confusion matrix for the majority baseline.

4.3 Comparative Analysis of Models

The logistic approach provides large benign workload for wide attack capture: two false positives against 1,030 under boosting. One potential element in the decision behaviour of this model is class weighting, however, the experiment cannot isolate the factor without a controlled ablation. Random forest and boosting have comparable high recall; boosting has two less benign false alarms at the specified threshold. The dummy baseline shows that great overall accuracy may be achieved with 0% bot recall (He & Garcia, 2009).

Table 11. Operational error comparison

Model	False alerts per 23,500 benign	Missed bots per 500	Flagged total
Majority baseline	0	500	0
Logistic regression	1030	0	1530
Random forest	4	3	501
Histogram boosting	2	3	499

The exact-signature sensitivity check trims the test set to 23,126 observations and adjusts the positive prevalence from 2.083% to 1.483%. Boosting gives 340 TP, 3 FN, 2 FP, 22,781 TN, F1 is 0.9927, AP 0.9961. Random woodland has 340 TP, 3 FN, 4 FP and 22779 TN. F1 is 0.9898 and AP is 0.9944. Performance is still excellent on this particular diagnosis but signature elimination is not a whole new domain.

Table 12. Sensitivity after excluding 874 exact training-signature matches

Model	Test n	TP	FN	FP	F1	AP
Majority baseline	23126	0	343	0	0.0000	0.0148
Logistic regression	23126	343	0	849	0.4469	0.8934
Random forest	23126	340	3	4	0.9898	0.9944
Histogram boosting	23126	340	3	2	0.9927	0.9961

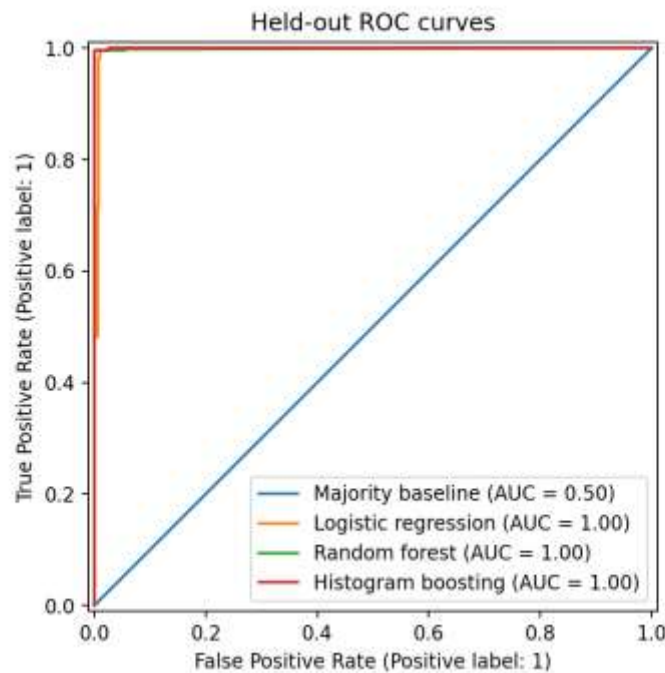


Figure 8. ROC curves for all four estimators on the held-out split

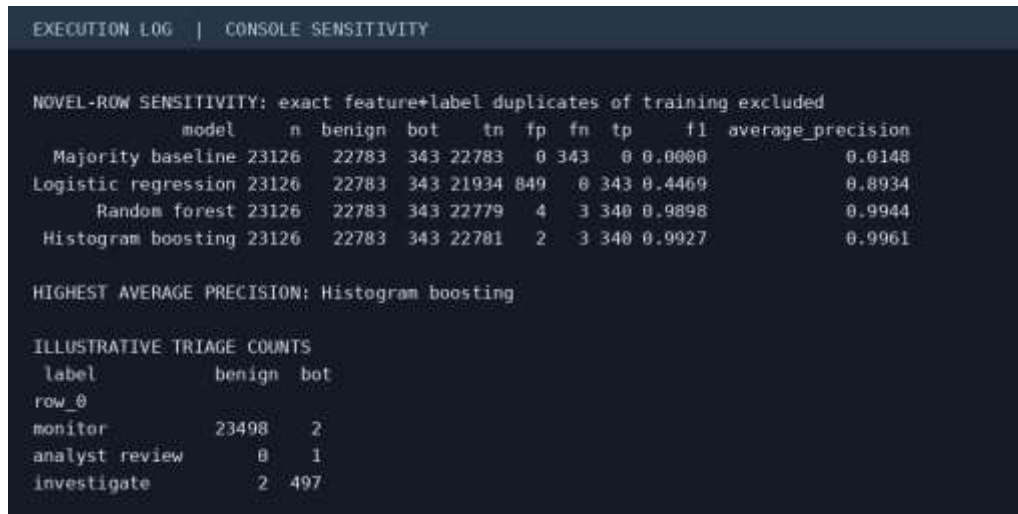


Figure 9. Faithful visual rendering of the recorded sensitivity and triage log

5. DISCUSSION

This empirical solution addresses a narrow classification question: two tree ensembles differentiate two observed bot and benign patterns with relatively few mistakes on this specified benchmark. The finding does not show that a security team would stop more breaches. CSE-CIC-IDS2018 is generated in a controlled setting and this derivative intentionally picks one attack family. Real traffic may vary in prevalence, software setup, benign burden, hostile strategies, and accessible features. In this experiment, we interpret the results under the restriction that closed-world assumptions are damaging to real network identification (Sommer and Paxson, 2010).

The exact-signature diagnosis is of particular interest since different source indices alone are not sufficient to prove feature independence. Out of 24,000 test records, 874 shared a complete feature-and-label signature with some training record. Removing them reduces prevalence and hence AP, although boosting is still significant. The other rows may have similar collecting circumstances or wider patterns. A stronger protocol wouldn't have an attack day, machine, organisation or external dataset. UNSW-NB15 is an example of another network intrusion corpus, although it is different in label definitions and collecting procedure and cannot be used with this code without a planned harmonisation research (Moustafa & Slay, 2015).

The best model was determined according on its held-out AP. A selection step, if the model is selected as a deployment candidate, and needs another untouched assessment set. Additionally, no hyperparameter search, repeated random seed evaluation, cost-sensitive threshold analysis, class-probability calibration, confidence intervals, or adversarial robustness analysis was undertaken. The 0.10 and 0.50 workflow bands are for illustration purposes. The claim of decision-support would be testable if scores were calibrated on different validation data and real analyst effort was measured (Guo et al., 2017; NIST, 2023).

Table 13. Valid claims and remaining evidence gaps

Claim supported here	Evidence still required
High discrimination for this 2018 subset	Independent cloud traffic and later attack periods
Low false alerts at fixed threshold	Observed SOC workload and prevalence
Robustness to exact signature overlap	Host/day/site holdout and leakage audit
Score bands allocate this test set	Prospective analyst decision and cost study
Reusable implementation	Versioned schema, drift, audit, security review

Network sensors are only one layer from a data-security perspective. Access management, configuration control, encryption, incident response, and ongoing surveillance are also required for cloud security (Cloud Security Alliance, 2024; Jansen & Grance, 2011; NIST, 2011, 2020, 2025). This analysis observes no content of customer and no trail of cloud audit. Limit and monitor data processing in deployment telemetry Document permitted use and limit access to models , logs , and possibly sensitive flow information.

6. CONCLUSION

A replicable comparison on the 48,000-row CSE-CIC-IDS2018 bot-versus-benign derivative discovered histogram gradient boosting identified 497 bots, missed three bots, and had two benign false alarms among 24,000 held-out flows, with AP 0.9978 and F1 0.9950 at threshold 0.50. It maintained AP 0.9961 and F1 0.9927 on 23,126 test rows when precise feature-and-label signatures did not emerge in training. Random forest was close . Class weighted logistic regression got every bot but had a lot more false warnings .

These are benchmark detection findings, not quantifiable advances in cloud security outcomes or human decision making. The accompanying framework makes clear the evidence border and provides a practical analyst triage procedure. The runnable code and source link allow a guide or reviewer to analyse the split, recreate the matrices and graphs, and verify new assumptions.

7. FUTURE SCOPE

First we test on separate organization/date/host/cloud-provider splits with an undisturbed final cohort. Second, label harmonisation across documents (CIC, 2018; Moustafa & Slay, 2015) by including various novel attack scenarios and other datasets. Third, use training/validation data to choose model and alert threshold and include calibrated probabilities, base-rate change, and costs of missed assaults and false reviews (Guo et al., 2017). Fourth, do a prospective analyst study tracking review time, verified incident yield, escalation accuracy and reaction results. Fifth, assess concept drift, adversary adaptation, model explanation, and control of stored security telemetry (Lundberg & Lee, 2017; NIST, 2011, 2023).

Table 14. Proposed next-stage validation plan

Work package	Measured endpoint	Decision gate
External split	Recall, AP, FP per workload	Replication beyond original testbed
Calibration	Reliability diagram and Brier score	Score band interpretation
Analyst pilot	Review time and verified yield	Actual decision benefit
Security monitoring	Drift and failure reports	Retraining or rollback rules

References

- Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32. <https://doi.org/10.1023/A:1010933404324>
- Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153–1176. <https://doi.org/10.1109/COMST.2015.2494502>
- Canadian Institute for Cybersecurity. (2018). CSE-CIC-IDS2018 on AWS [Data set]. University of New Brunswick. <https://www.unb.ca/cic/datasets/ids-2018.html>
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357. <https://doi.org/10.1613/jair.953>
- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. <https://doi.org/10.1145/2939672.2939785>
- Cloud Security Alliance. (2024). Top threats to cloud computing 2024. <https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-2024>
- Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 233–240). ACM. <https://doi.org/10.1145/1143844.1143874>
- Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 861–874. <https://doi.org/10.1016/j.patrec.2005.10.010>
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189–1232. <https://doi.org/10.1214/aos/1013203451>
- Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 1321–1330). PMLR. <https://proceedings.mlr.press/v70/guo17a.html>
- He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9), 1263–1284. <https://doi.org/10.1109/TKDE.2008.239>
- Jansen, W., & Grance, T. (2011). Guidelines on security and privacy in public cloud computing (NIST SP 800-144). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-144>
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 3146–3154). <https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree>
- Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 4768–4777). <https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html>
- Mell, P., & Grance, T. (2011). The NIST definition of cloud computing (NIST SP 800-145). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-145>
- Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. In *2015 Military Communications and Information Systems Conference (MilCIS)* (pp. 1–6). IEEE. <https://doi.org/10.1109/MilCIS.2015.7348942>

17. National Institute of Standards and Technology. (2011). Information security continuous monitoring (ISCM) for federal information systems and organizations (NIST SP 800-137). <https://doi.org/10.6028/NIST.SP.800-137>
18. National Institute of Standards and Technology. (2020). Security and privacy controls for information systems and organizations (NIST SP 800-53 Rev. 5). <https://doi.org/10.6028/NIST.SP.800-53r5>
19. National Institute of Standards and Technology. (2023). Artificial intelligence risk management framework (AI RMF 1.0) (NIST AI 100-1). <https://doi.org/10.6028/NIST.AI.100-1>
20. National Institute of Standards and Technology. (2025). Incident response recommendations and considerations for cybersecurity risk management: A CSF 2.0 community profile (NIST SP 800-61 Rev. 3). <https://doi.org/10.6028/NIST.SP.800-61r3>
21. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. <https://www.jmlr.org/papers/v12/pedregosa11a.html>
22. rayborg. (n.d.). Download and use the public bundle: CSE-CIC-IDS2018 bot-versus-benign preserved-ratio CSV [Data set]. DCTABGAN IDS benchmark datasets. <https://rayborg.github.io/dctabgan-ids-benchmark-datasets/download.html>
23. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). “Why should I trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). ACM. <https://doi.org/10.1145/2939672.2939778>
24. Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3), e0118432. <https://doi.org/10.1371/journal.pone.0118432>
25. Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy* (pp. 108–116). SCITEPRESS. <https://doi.org/10.5220/0006639801080116>
26. Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy* (pp. 305–316). IEEE. <https://doi.org/10.1109/SP.2010.25>