

A Hybrid Light GBM Framework Integrating Feature Selection and Cost-Sensitive Learning for Network Outlier Detection

Asif Ali Syed

Research Scholar, Department of Computer Science, Malwanchal University, Indore

Dr. Manav Thakur

Supervisor, Department of Computer Science, Malwanchal University, Indore

Abstract

Machine learning intrusion detectors are often judged on benchmarks with redundant features, severe class imbalance and evaluation shortcuts. This study builds and evaluates a hybrid framework that combines a correlation filter, a rank-aggregated feature ranking based on mutual information and LightGBM split gain, softened square-root cost-sensitive weighting, and a regularised LightGBM classifier with early stopping. Every design choice was settled on a validation subset carved from the official UNSW-NB15 training partition, and the official test partition of 82,332 flows was used once for final measurement. The pipeline retained 15 of 42 features. Square-root weighting gave the best validation macro-F1 (0.634 against 0.601 without treatment) and outperformed SMOTE and hybrid resampling. On the test partition, the framework reached a ten-class macro-F1 of 0.530, the highest of seven models, and a binary F1 of 0.890 with an AUC of 0.977. Its accuracy was, however, significantly lower than that of five of six baselines under McNemar's test, and its false alarm rate was 25.6%. An ablation study showed that stabilising the learner with regularisation and early stopping contributed 7.0 of the 9.4 points of macro-F1 gained over default LightGBM. Feature selection cut training time by 37.3%. SHAP analysis identified the source time-to-live as the dominant feature, yet removing it changed binary F1 by only 0.001. A nine-point gap between validation and test binary F1 was traced to distribution shift, not duplicate leakage. The framework improves rare-class detection at low training cost, but benchmark results on this dataset must be read with caution.

Keywords: network intrusion detection; LightGBM; feature selection; class imbalance; cost-sensitive learning; UNSW-NB15; SHAP

1. Introduction

1.1 Network Security in a Growing Digital Environment

Modern organisations depend on networks that carry ever larger volumes of traffic between cloud services, remote users and connected devices. Intrusion detection systems (IDS) are the main defensive layer that monitors this traffic and raises alerts when activity departs from expected behaviour (Khraisat et al., 2019). Signature-based systems miss novel attacks, while anomaly-based systems can flag unseen behaviour at the cost of more false alarms.

1.2 Machine Learning as a Basis for Outlier Detection

Machine learning has become the dominant approach to anomaly-based detection, because classifiers can learn the statistical signatures of attack families from labelled flow records

instead of relying on hand-written rules. Surveys document a wide spread of methods, from decision trees and support vector machines to ensembles and deep neural networks (Ahmad et al., 2021; Liu & Lang, 2019). In this setting an attack flow can be viewed as an outlier with respect to normal traffic, and a supervised multi-class model can additionally identify which family it belongs to. Benchmark datasets such as UNSW-NB15 and CICIDS2017 provide common ground for comparing these methods (Moustafa & Slay, 2015; Sharafaldin et al., 2018).

1.3 Redundant Features and Imbalanced Traffic

Two properties of flow data complicate learning. First, flow features are often redundant: byte counters, packet counters and loss counters tend to rise together, and connection-count features computed over overlapping windows are strongly correlated. Redundant features add cost without adding information and can make models harder to interpret, which is why feature selection is regarded as one of the most influential design decisions in IDS research (Li et al., 2017; Thakkar & Lohiya, 2022). Second, attack families are highly imbalanced. In UNSW-NB15 the largest training class outnumbers the smallest by more than 400 to 1, so a classifier can achieve high accuracy while almost never recognising rare attacks. Resampling methods such as SMOTE and algorithm-level cost-sensitive learning are the two main remedies, but their benefits vary by dataset and neither is guaranteed to help (Fernández et al., 2018; Jiang et al., 2020).

1.4 Gradient Boosting for Efficient Detection

Gradient boosting decision trees have emerged as strong learners for tabular data, often matching or outperforming deep networks on heterogeneous, heavy-tailed features (Grinsztajn et al., 2022). XGBoost popularised regularised boosting at scale (Chen & Guestrin, 2016), and LightGBM extended it with histogram-based split finding, leaf-wise tree growth, Gradient-based One-Side Sampling and Exclusive Feature Bundling, which together reduce training cost substantially (Ke et al., 2017). LightGBM also supports per-sample weights and early stopping natively, which makes it a natural base for a pipeline that combines feature reduction with cost-sensitive learning.

1.5 Evaluation Rigour in Security Machine Learning

A growing body of work warns that many reported IDS results are optimistic. Common pitfalls include tuning on test data, leaking information through resampling performed before data splitting, relying on accuracy under class imbalance, and learning artefacts of the testbed instead of attack behaviour (Arp et al., 2022; Engelen et al., 2021). Benchmarks also have known limits in age and realism (Ring et al., 2019; Sarhan et al., 2022). Against this background, this paper presents a hybrid LightGBM framework that integrates a correlation filter, a rank-aggregated hybrid feature ranking, softened cost-sensitive weighting and a regularised, early-stopped learner, and evaluates it under a strict protocol on UNSW-NB15. It reports where the framework wins, ties and loses against six baselines, measures each component's contribution, examines interpretability with SHAP, and quantifies the gap between validation and test performance.

2. Literature Review

2.1 Machine Learning Approaches to Intrusion Detection

Early IDS research relied on single classifiers trained on KDD Cup 99 and NSL-KDD, but systematic reviews show a shift toward ensembles and deep learning, with random forests, gradient boosting, convolutional and recurrent networks all reporting high detection rates (Ahmad et al., 2021; Liu & Lang, 2019). Khraisat et al. (2019) classified IDS techniques into signature-based, anomaly-based and hybrid systems and noted that anomaly-based machine learning methods offer the best prospect of detecting zero-day attacks, while suffering from higher false alarm rates. Apruzzese et al. (2018) argued that false alarms and analyst time remain major barriers to adoption, even when benchmark accuracy appears high.

The choice of benchmark has also evolved. UNSW-NB15 was created to address the outdated traffic and attack mix of KDD-derived datasets, combining realistic normal activity with nine modern attack families (Moustafa & Slay, 2015). It has since been used to evaluate geometric and statistical anomaly detection on large-scale networks (Moustafa et al., 2019). CICIDS2017 added a richer feature set and more recent attacks (Sharafaldin et al., 2018), but later analysis exposed labelling errors and infinite values that can distort reported results (Engelen et al., 2021).

2.2 Feature Selection and Class Imbalance Techniques

Feature selection methods are commonly grouped into filter, wrapper and embedded approaches (Li et al., 2017). Filters such as correlation analysis and mutual information are cheap and model-independent but ignore feature interactions. Wrappers capture interactions at high cost, while embedded methods derive importance from the learner itself, as with tree split gain. Thakkar and Lohiya (2022) reviewed feature selection for IDS and identified hybrid schemes, which combine filter and embedded criteria, as a promising direction for balancing accuracy and cost. Kasongo and Sun (2020) applied a filter-based feature selection method to UNSW-NB15 and reported that a reduced feature set could maintain detection performance while lowering training time, a finding that motivates the selection stage of the present framework.

For class imbalance, data-level methods modify the training set, and algorithm-level methods modify the loss. SMOTE, which generates synthetic minority samples by interpolating between neighbours, remains the most widely used data-level technique, and Fernández et al. (2018) reviewed its many variants together with its known weaknesses on noisy, overlapping and mixed-type data. Jiang et al. (2020) combined one-sided selection undersampling with SMOTE and a deep hierarchical network and reported improved detection on NSL-KDD and UNSW-NB15. Cost-sensitive learning, in contrast, leaves the data unchanged and assigns larger loss weights to rare classes, which keeps training cost low. Few studies compare these families under identical learners and strict leakage control.

2.3 Gradient Boosting, Interpretability and Evaluation Pitfalls

Gradient boosting is now a standard strong baseline for tabular problems. Chen and Guestrin (2016) introduced XGBoost with second-order optimisation and regularisation, and Ke et al.

(2017) showed that LightGBM achieves comparable accuracy with considerably faster training through histogram binning and leaf-wise growth. Grinsztajn et al. (2022) offered broad evidence that tree-based models still outperform deep learning on typical tabular data, largely because they are robust to uninformative and irregularly distributed features. For interpretability, SHAP provides consistent local and global explanations grounded in Shapley values (Lundberg & Lee, 2017).

Methodological critiques have sharpened expectations for evaluation. Arp et al. (2022) identified ten recurring pitfalls in security machine learning, including data snooping, spurious correlations and inappropriate baselines, and found them to be widespread in published work. Engelen et al. (2021) demonstrated how dataset construction errors can lead models to learn artefacts. Sarhan et al. (2022) proposed a standard feature set to enable cross-dataset evaluation and showed that models trained on one dataset often generalise poorly to another. The gap this study addresses lies at the intersection of these strands: few frameworks combine hybrid feature selection, cost-sensitive boosting and interpretability while also applying a strict protocol, reporting unfavourable results and testing whether the model depends on testbed shortcuts.

3. Research Methodology

3.1 Research Design

The study follows a positivist, quantitative and experimental design with a design-science element. The artefact is the hybrid framework, built from components chosen by validation evidence, and the experiments measure it against alternatives under controlled conditions. Each experiment changed one factor at a time, such as the number of features, the imbalance strategy or a single hyperparameter, while the data partition, learner and random seed (42) were held fixed. Four principles governed the work: strict separation of selection and evaluation, no leakage from resampling or ranking, metrics that expose rare-class failure, and honest reporting of negative results (Arp et al., 2022). All experiments ran on a single CPU core with about 4 GB of memory and no GPU, using Python 3.12, LightGBM 4.7, XGBoost 3.4, scikit-learn 1.8, imbalanced-learn 0.14 and SHAP 0.52.

3.2 Dataset, Preprocessing and Partitioning

UNSW-NB15 served as the primary dataset (Moustafa & Slay, 2015; Moustafa et al., 2019). Its official training partition contains 175,341 flows and its testing partition 82,332 flows, each described by 42 predictive features (39 numeric and three categorical: proto, service and state) and labelled as Normal or one of nine attack families: Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode and Worms. Normal accounts for 56,000 training records and Worms for only 130, a ratio of 430.8 to 1. An audit found no missing or infinite values but 67,601 exact duplicate rows (38.6%) in training and 26,387 (32.0%) in testing. Duplicates were retained for comparability with published work, and their effect was measured directly in the robustness analysis.

Preprocessing was minimal. The record identifier was dropped, the three categorical features were ordinal-encoded with an encoder fitted on training data (unseen categories mapped to -1), and no scaling was applied for tree models. The training partition was split once, with

stratification on the ten-class label, into an 80% fitting subset (140,272 records) and a 20% validation subset (35,069 records). All design choices were made on the validation subset, and the official test partition was opened only after the framework had been fixed. A partial single-day CICIDS2017 DDoS extract (225,745 flows) was used only in a supplementary binary experiment, because the complete files were not available in the computing environment.

3.3 Hybrid Feature Selection

Feature selection proceeded in two stages computed on the fitting subset only. A correlation filter first listed all feature pairs with an absolute Pearson correlation above 0.95, in decreasing order, and dropped the second feature of each pair unless it had already been removed. The surviving features were then ranked by two criteria. Mutual information with the class label (estimated on 40,000 records) captures nonlinear dependence but ignores interactions, whereas LightGBM split gain captures interactions but can be dominated by a few features (Ke et al., 2017). Because a product of normalised scores was dominated by extreme gain values, the criteria were combined by rank aggregation: each feature's hybrid score is one minus its normalised mean rank across the two criteria, so a feature ranked first by both scores 1 and one ranked last by both scores 0. The number of retained features k was swept over {5, 10, 15, 20, 25, 30, 32, 42}, and k was set at the knee of the validation macro-F1 curve, the smallest k within about two points of the best score.

3.4 Class Imbalance Handling

Seven configurations were compared using the selected features and the same regularised LightGBM learner: no treatment; fully balanced class weights $w_c = N/(C n_c)$; softened square-root weights equal to the square root of that quantity; random oversampling; SMOTE with five neighbours; random undersampling followed by SMOTE; and undersampling plus SMOTE with square-root weights (Fernández et al., 2018; Jiang et al., 2020). Resampling targets were moderate: each minority class was raised to at least 10% of the largest class, and in the hybrid variant large classes were first capped at 50%. All resampling was applied after splitting, to the fitting subset only, so no synthetic sample was derived from validation or test records. The configuration with the highest validation macro-F1 was selected by a rule fixed in advance.

3.5 The LightGBM Learner and Tuning

LightGBM fits an additive ensemble of regression trees, $F_M(x) = F_0(x) + \eta \sum h_m(x)$, minimising the multi-class cross-entropy over softmax outputs, with each loss term multiplied by the class weight when weighting is used (Ke et al., 2017). A preliminary experiment compared default LightGBM (learning rate 0.1, 500 rounds, no early stopping) with a regularised configuration (learning rate 0.05, at least 50 samples per leaf, L2 weight 1). Early stopping with a patience of 30 rounds and a maximum of 1,000 iterations was then built into the framework. Hyperparameters were tuned by random search over eight configurations drawn from a space covering the number of leaves, maximum depth, learning rate, minimum child samples, column and row sampling fractions and L2 regularisation.

3.6 Experimental Design and Evaluation

The framework was compared with six baselines trained on the same fitting subset with all 42 features and default or near-default settings: a decision tree, a random forest (100 trees), extremely randomised trees (100 trees), logistic regression, a multilayer perceptron (64 and 32 units) and XGBoost (300 trees) (Chen & Guestrin, 2016). A six-step ablation (A0 to A5) added regularisation and early stopping, feature selection, imbalance weighting and tuning in turn. Efficiency was measured by training time at fixed tree counts and inference latency. SHAP values were computed for the binary model on 2,000 test flows (Lundberg & Lee, 2017), followed by a feature-removal test. Robustness was assessed by five-fold stratified cross-validation and a comparison of validation and test scores split by duplicate status.

Macro-averaged F1 was the primary metric, because accuracy and weighted-F1 are dominated by the large classes (Thakkar & Lohiya, 2022). Accuracy, weighted-F1, balanced accuracy, the Matthews correlation coefficient (MCC), the false alarm rate (FAR) and the area under the ROC curve (AUC) were also reported. McNemar's test with continuity correction was applied to paired per-flow correctness, with a Bonferroni-adjusted threshold of 0.0042 for the twelve comparisons.

4. Results and Discussion

4.1 Feature Selection

The correlation filter found 13 pairs above the threshold, including source bytes and source loss ($r = 0.996$), destination bytes and destination loss ($r = 0.997$), and is_ftp_login and ct_ftp_cmd ($r = 1.000$). Removing one member of each pair eliminated ten features and reduced the set from 42 to 32, a 23.8% reduction before any learning. The two criteria disagreed substantially (sttl ranked first by gain but fourteenth by mutual information), which justified combining them. The top 15 features by hybrid score were smean, ct_dst_sport_ltm, dmean, sload, sttl, ct_srv_src, proto, service, dtl, dur, ct_dst_ltm, tcprrt, dpkts, dinpkt and dload, a mixture of size, timing, identity and short-term context features.

Table 1 and Figure 1 show validation performance as k grows. Macro-F1 rose from 0.548 at $k = 5$ to 0.601 at $k = 15$ and then plateaued until all 42 features were used (0.617). Moving from 42 to 15 features therefore cost 1.68 points of validation macro-F1 and 0.33 points of binary F1 while removing 64.3% of the features, so $k = 15$ was chosen as the knee. In a comparison of ranking criteria, mutual information alone was weakest at $k = 10$ (0.555), gain alone reached 0.605 and the hybrid 0.594, while at $k = 20$ all three were within 0.002 of one another. The value of the hybrid ranking thus lies in robustness at small k rather than in a measured gain over split gain alone.

Table 1

Validation Performance Against the Number of Retained Features

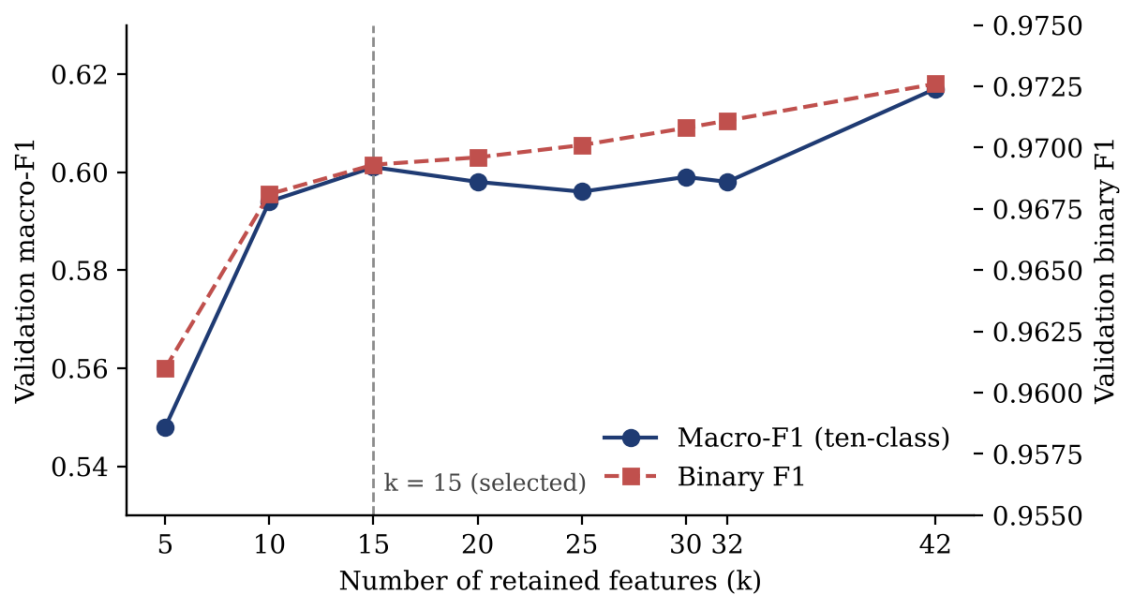
k	Macro-F1	Weighted-F1	Binary F1	Training time (s)	Iterations
5	0.548	0.791	0.9610	49.9	543

k	Macro-F1	Weighted-F1	Binary F1	Training time (s)	Iterations
10	0.594	0.808	0.9681	39.6	353
15	0.601	0.811	0.9693	58.3	449
20	0.598	0.813	0.9696	56.2	395
25	0.596	0.812	0.9701	56.5	330
30	0.599	0.812	0.9708	66.7	397
32	0.598	0.811	0.9711	60.5	357
42	0.617	0.818	0.9726	71.3	372

Note. Values for k = 5 to 32 use the top-ranked features after the correlation filter; k = 42 uses all features. Training time includes early stopping.

Figure 1

Validation Macro-F1 and Binary F1 Against the Number of Retained Features



4.2 Class Imbalance Handling

Table 2 and Figure 2 compare the seven imbalance strategies. The effect of rebalancing was not uniformly positive. Fully balanced class weights lowered macro-F1 from 0.601 to 0.570, even though balanced accuracy rose to 0.613, because extreme weights on a few rare samples traded too much precision for recall. SMOTE alone also slightly lowered macro-F1 to 0.595. Softened square-root weighting gave the best macro-F1 (0.634) and weighted-F1 (0.822) without changing the training set, while the combined undersampling, SMOTE and square-root configuration gave the best balanced accuracy (0.660) but a lower macro-F1 (0.622). Square-root weighting was therefore selected.

No strategy solved the rare-class problem. Shellcode recall rose from 0.612 to as high as 0.771 and Worms from 0.500 to 0.692, but Analysis recall stayed between 0.175 and 0.313 and Backdoor between 0.109 and 0.246. These classes overlap with DoS and Exploits in feature space, and rebalancing cannot create discriminative signal that the features do not contain. SMOTE's weak result is consistent with interpolation blurring overlapping boundaries and producing unrealistic protocol codes (Fernández et al., 2018). Hybrid sampling improved on no treatment (0.615 against 0.601), in line with Jiang et al. (2020), but did not match cost-sensitive weighting.

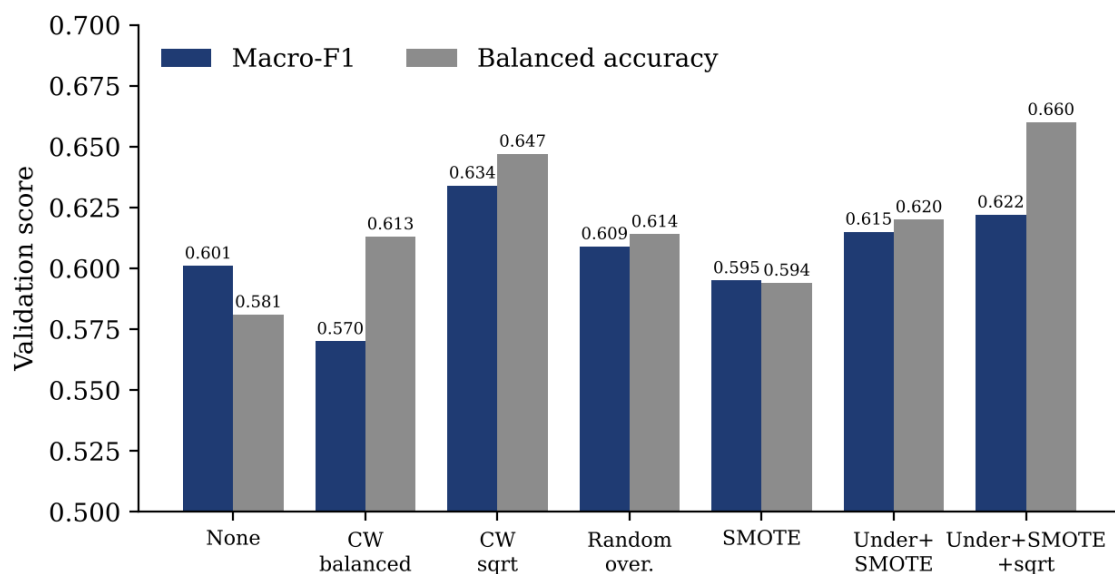
Table 2

Validation Results for Imbalance-Handling Strategies (15 Features)

Strategy	Training samples	Macro-F1	Weighted-F1	Balanced accuracy	Training time (s)
None (baseline)	140,272	0.601	0.811	0.581	56.1
Class weights (balanced)	140,272	0.570	0.794	0.613	82.8
Class weights (square-root)	140,272	0.634	0.822	0.647	64.8
Random oversampling	154,185	0.609	0.810	0.614	45.7
SMOTE	154,185	0.595	0.812	0.594	67.0
Undersampling + SMOTE	117,871	0.615	0.818	0.620	37.6
Undersampling + SMOTE + square-root	117,871	0.622	0.816	0.660	52.0

Figure 2

Validation Macro-F1 and Balanced Accuracy by Imbalance Strategy



4.3 Framework Stabilisation and Tuning

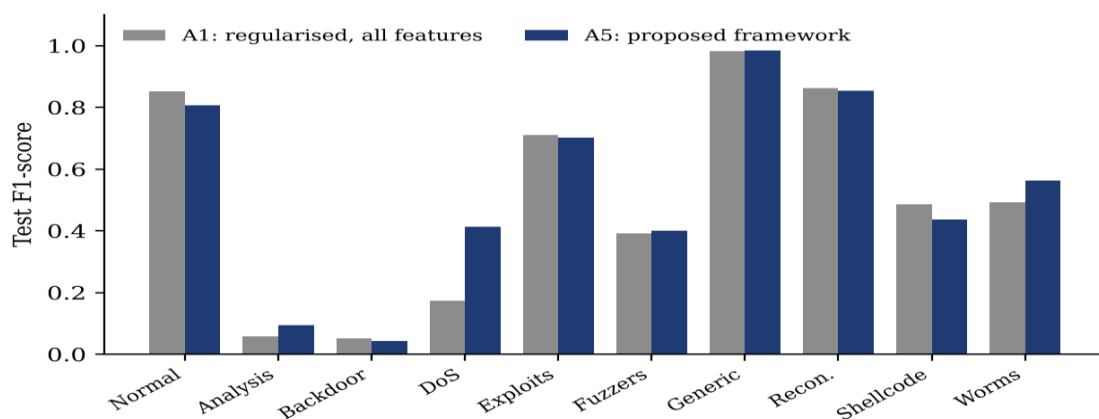
Default LightGBM proved unstable on this data. Its validation log-loss reached a minimum of 0.505 at iteration 30 and then rose steadily to 7.41 by iteration 500, whereas the regularised configuration reached a lower minimum of 0.407 at iteration 372 and remained flat thereafter. A fixed number of boosting rounds is therefore unsafe on UNSW-NB15. The random search produced validation macro-F1 values between 0.628 and 0.637, a spread of only 0.9 points. The selected configuration used 63 leaves, a maximum depth of 12, a learning rate of 0.05, at least 50 samples per leaf and an L2 weight of 1, and stopped at 324 iterations, giving 3,240 trees and a 20.1 MB model. Tuning improved validation macro-F1 by 0.5% in relative terms, less than the cross-validation standard deviation of 0.008, and the sensitivity analysis showed that no single parameter moved macro-F1 by more than 1.2 points.

4.4 Detection Performance on the Test Partition

Evaluated once on the 82,332 test flows, the ten-class framework reached an accuracy of 73.7%, a macro-F1 of 0.530, a weighted-F1 of 0.772, a balanced accuracy of 0.589 and an MCC of 0.671. Figure 3 compares per-class F1 with the regularised all-feature model (A1). Generic was detected almost perfectly (F1 0.984), and Reconnaissance (0.854) and Exploits (0.702) were handled reasonably well. Imbalance weighting raised DoS F1 from 0.173 to 0.414 and Worms from 0.493 to 0.562, but lowered Normal F1 from 0.851 to 0.806 and Exploits recall from 0.844 to 0.653, which is the expected trade-off of cost-sensitive learning. Analysis (0.095) and Backdoor (0.043) remained close to failure, since most of their flows were classified as DoS.

Errors were concentrated in a few confusions. The largest, Normal classified as Fuzzers, involved 9,324 flows, or 43.1% of all 21,628 errors. Normal flows predicted as attacks made up 52.7% of errors, against 4.4% for missed attacks. The binary model reached an accuracy of 86.8%, a precision of 0.823, a recall of 0.969, an F1 of 0.890, an MCC of 0.743 and an AUC of 0.977, missing only 3.1% of attack flows. Its false alarm rate of 25.6%, however, means about one normal flow in four was flagged, a level too high for many operational settings (Apruzzese et al., 2018).

Figure 3 Per-Class Test F1 of the Regularised All-Feature Model (A1) and the Proposed Framework (A5)



4.5 Comparison with Baseline Classifiers

Table 3 and Table 4 compare the framework with the six baselines, and Figure 4 summarises the primary metrics. In the ten-class task the framework achieved the highest macro-F1 (0.530, against 0.512 for XGBoost) and the highest balanced accuracy (0.589). On every other ten-class metric it was weaker: its accuracy ranked sixth of seven, and its weighted-F1 and MCC were level with the decision tree and below XGBoost. McNemar's test confirmed that five of the six baselines were significantly more accurate at the flow level ($p < 0.01$), and only logistic regression was significantly less accurate. With the Bonferroni threshold, the difference against the perceptron ($p = 0.007$) would not be significant. The framework's advantage is therefore confined to metrics that weight rare classes equally, and it was obtained against baselines that received no imbalance treatment.

Table 3

Ten-Class Results on the Official Test Partition

Model	Accuracy	Macro-P	Macro-R	Macro-F1	Weighted-F1	MCC
Decision tree	0.751	0.486	0.526	0.493	0.773	0.673
Random forest	0.763	0.540	0.500	0.471	0.779	0.695
Extra trees	0.757	0.559	0.462	0.453	0.769	0.687
Logistic regression	0.670	0.339	0.356	0.317	0.676	0.581
MLP (64-32)	0.741	0.473	0.462	0.419	0.747	0.669
XGBoost	0.770	0.535	0.553	0.512	0.783	0.701
Proposed framework	0.737	0.520	0.589	0.530	0.772	0.671

Note. Baselines use all 42 features, default or near-default settings and no imbalance treatment. P = precision; R = recall.

In the binary task the framework (F1 0.890) ranked level with the decision tree and the perceptron and behind XGBoost (0.896) and the random forest (0.894). McNemar's test found no significant difference from the decision tree ($p = 0.29$) or the perceptron ($p = 0.25$), while XGBoost and the random forest were significantly better. The binary results therefore place the framework in the top group but do not support a claim of superiority. Tree ensembles outperformed the neural and linear models on macro-F1, consistent with Grinsztajn et al. (2022). Absolute comparison with published figures, such as those of Kasongo and Sun (2020), was avoided because preprocessing choices differ between studies (Arp et al., 2022).

Table 4

Binary Detection Results on the Official Test Partition

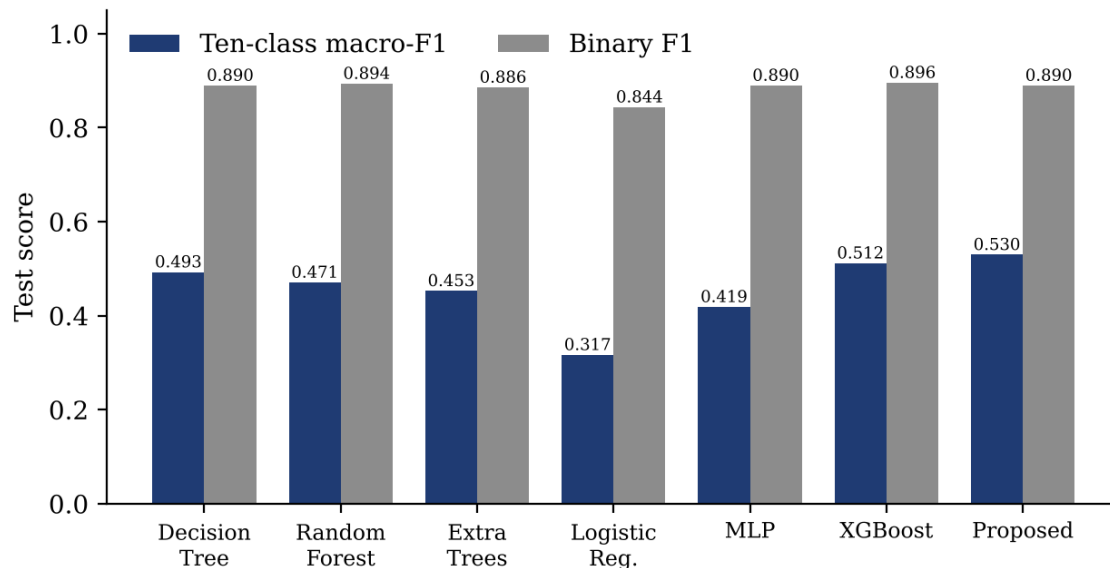
Model	Acc.	Precision	Recall	F1	MCC	FAR	AUC
Decision tree	0.869	0.826	0.965	0.890	0.744	0.248	0.906
Random forest	0.871	0.816	0.987	0.894	0.754	0.272	0.982

Model	Acc.	Precision	Recall	F1	MCC	FAR	AUC
Extra trees	0.860	0.803	0.987	0.886	0.734	0.296	0.979
Logistic regression	0.803	0.749	0.967	0.844	0.625	0.398	0.933
MLP (64-32)	0.867	0.819	0.973	0.890	0.742	0.263	0.975
XGBoost	0.875	0.824	0.983	0.896	0.760	0.257	0.984
Proposed framework	0.868	0.823	0.969	0.890	0.743	0.256	0.977

Note. Acc. = accuracy; FAR = false alarm rate. The proposed binary model uses 15 features; baselines use all 42.

Figure 4

Ten-Class Macro-F1 and Binary F1 of All Models on the Test Partition



4.6 Efficiency

With the number of trees fixed at 150 per class, LightGBM on 15 features trained on the full fitting subset in 12.5 seconds, compared with 19.9 seconds on 42 features, 21.7 seconds for XGBoost and 22.1 seconds for the random forest. Feature selection thus gave a speed-up of 1.59 and a 37.3% reduction in training time. The complete framework, including early stopping, trained in 45.6 seconds, slightly more than XGBoost (41.8 seconds). Inference was the framework's weakest point: the ten-class model needed 370.5 milliseconds per 1,000 flows, about 2,700 flows per second, whereas the binary model processed about 18,900 flows per second. The binary model is therefore the realistic first-stage detector, and sub-minute retraining makes periodic updates feasible.

4.7 Ablation Study

Table 5 and Figure 5 show the contribution of each component on the test partition. By far the largest gain came from stabilising the learner: regularisation and early stopping raised macro-F1 from 0.436 (A0) to 0.506 (A1), a gain of 7.0 points. Imbalance weighting contributed 1.9

points when added to A1 and 1.6 points when added after feature selection. Feature selection alone added 0.5 points and raised accuracy to 77.5%, and tuning added a further 0.2 points. Of the 9.4-point cumulative gain, three quarters came from the first step. The feature selection and tuning gains fall within cross-validation noise, so the main benefit of feature selection is lower cost.

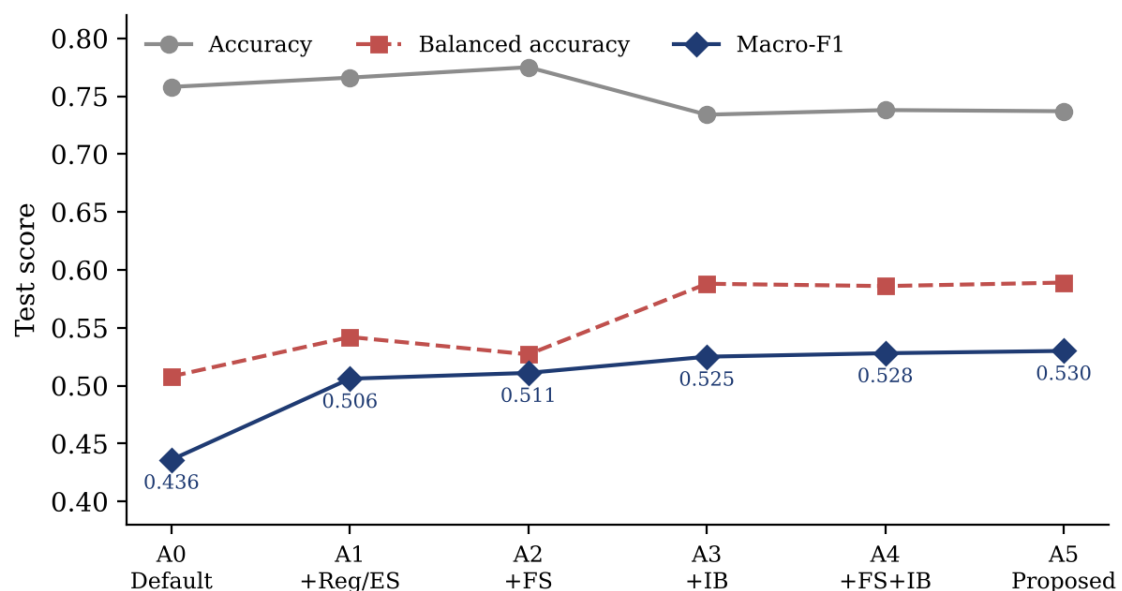
Table 5

Ablation Study on the Official Test Partition (Ten-Class)

Configuration	Accuracy	Balanced accuracy	Macro-F1	MCC	Features
A0 Default LightGBM (100 trees)	0.758	0.508	0.436	0.685	42
A1 + Regularisation and early stopping	0.766	0.542	0.506	0.697	42
A2 A1 + feature selection	0.775	0.527	0.511	0.704	15
A3 A1 + imbalance weighting	0.734	0.588	0.525	0.667	42
A4 A1 + feature selection + weighting	0.738	0.586	0.528	0.672	15
A5 Proposed (A4 + tuning)	0.737	0.589	0.530	0.671	15

Figure 5

Test Accuracy, Balanced Accuracy and Macro-F1 Across Ablation Configurations



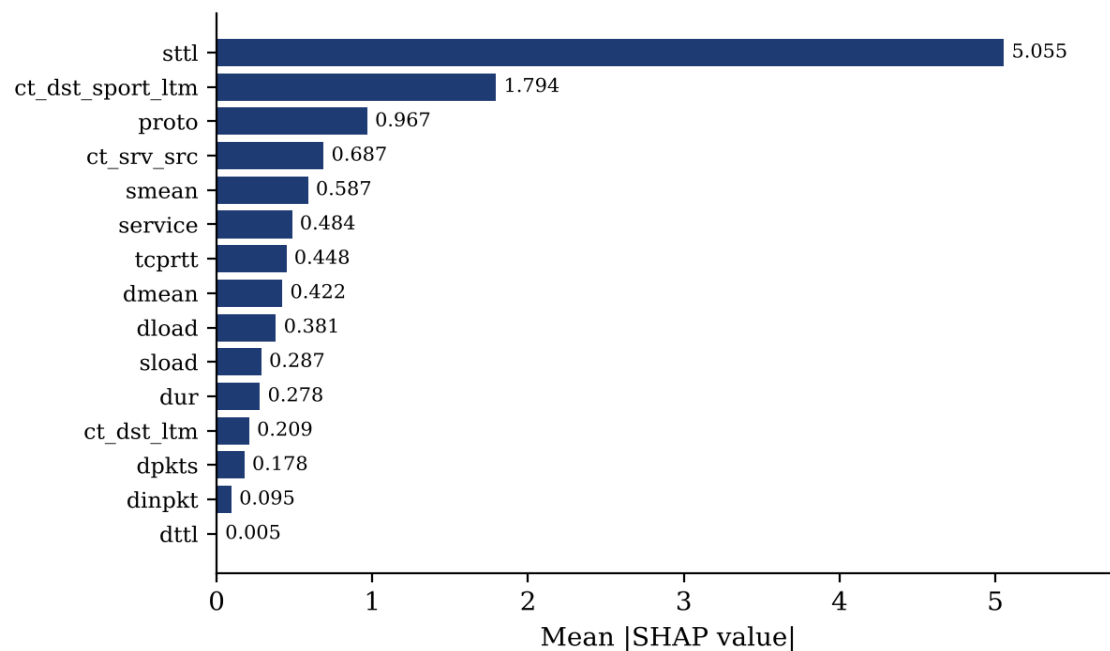
4.8 Interpretability and Robustness

Figure 6 presents the SHAP global importance of the 15 features for the binary model. The source time-to-live, sttl, dominated with a mean absolute SHAP value of 5.06, or 42.6% of the

total, and sttl, ct_dst_sport_ltm and proto together accounted for 65.8%. Because time-to-live values reflect the hosts and routes of the testbed rather than attack behaviour, the model was retrained without sttl. Test F1 changed only from 0.8900 to 0.8886, and removing all three TTL-related features gave 0.8884. The model is therefore not dependent on this single shortcut, although correlated features may encode the same testbed property, and 126 of the 133 protocol values in training appear only in attack flows (Engelen et al., 2021).

Figure 6

Mean Absolute SHAP Value of the 15 Selected Features (Binary Model, 2,000 Test Flows)



Five-fold cross-validation gave a macro-F1 of 0.634 (SD 0.008) and an accuracy of 0.812 (SD 0.002), showing stable behaviour within the training partition. Table 6 shows, however, a substantial gap between validation and test performance. Accuracy fell by 7.7 points, macro-F1 by 10.7 points and binary F1 by 9.0 points. Duplicate leakage did not explain the gap: validation rows with a training duplicate were classified less accurately than rows without one, and the gap persisted on non-duplicate rows. The gap is best explained by distribution shift between the official partitions, in which the Normal share rises from 31.9% to 44.9%. Random validation splits within the training partition are therefore an unreliable guide to test performance on this dataset, which supports the warnings of Arp et al. (2022) and Ring et al. (2019). On the partial CICIDS2017 extract, F1 was 0.9968 with 25 features and 0.9967 with 10, although this DDoS task is close to saturated.

Table 6

Validation and Test Performance of the Proposed Ten-Class Model

Metric	Validation	Test	Gap (points)
Accuracy	0.8141	0.7373	-7.7
Macro-F1	0.6367	0.5295	-10.7
Binary F1 (from ten-class predictions)	0.9676	0.8779	-9.0
Accuracy, rows with a training duplicate	0.7560	0.5663	-19.0
Accuracy, rows without a training duplicate	0.8650	0.7571	-10.8

5. Conclusion

This study built and evaluated a hybrid LightGBM framework that integrates a correlation filter, a rank-aggregated hybrid feature ranking, softened cost-sensitive weighting and a regularised, early-stopped learner for network outlier detection on UNSW-NB15. Under a strict protocol that kept model selection separate from final evaluation, the framework reduced the feature set from 42 to 15, achieved the highest ten-class macro-F1 (0.530) and balanced accuracy (0.589) among seven models, and reached a binary F1 of 0.890 with an AUC of 0.977. Feature selection cut training time by 37.3% at a fixed number of trees, and square-root weighting outperformed SMOTE and hybrid resampling without adding training cost.

The evidence also sets clear limits on these claims. The framework was significantly less accurate than five of six baselines at the flow level, it was not better than XGBoost or the random forest in binary detection, its false alarm rate of 25.6% is too high for direct deployment, and its inference was slow in this implementation. Rare classes such as Analysis and Backdoor remained poorly detected under every strategy. The ablation showed that most of the gain came from stabilising the learner, and a nine-point drop from validation to test exposed a distribution shift that no component of the framework addresses. Practitioners should therefore report macro-averaged and per-class metrics on the official test partition, never run gradient boosting with default fixed rounds on such data, soften rather than fully balance class weights, and test whether dominant explanatory features are true signals or testbed shortcuts. Future work should complete a full CICIDS2017 evaluation on a common feature set to test cross-dataset generalisation (Sarhan et al., 2022), apply the same class weights to strong baselines for a stricter comparison, repeat experiments across multiple random seeds, evaluate SMOTE variants designed for mixed data, and investigate threshold calibration and two-stage cascades to reduce false alarms.

References

1. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1), Article e4150. <https://doi.org/10.1002/ett.4150>
2. Apruzzese, G., Colajanni, M., Ferretti, L., Guido, A., & Marchetti, M. (2018). On the effectiveness of machine and deep learning for cyber security. In 2018 10th International Conference on Cyber Conflict (CyCon) (pp. 371–390). IEEE. <https://doi.org/10.23919/CYCON.2018.8405026>

3. Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., & Rieck, K. (2022). Dos and don'ts of machine learning in computer security. In Proceedings of the 31st USENIX Security Symposium (pp. 3971–3988). USENIX Association.
4. 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>
5. Engelen, G., Rimmer, V., & Joosen, W. (2021). Troubleshooting an intrusion detection dataset: The CICIDS2017 case study. In 2021 IEEE Security and Privacy Workshops (SPW) (pp. 7–12). IEEE. <https://doi.org/10.1109/SPW53761.2021.00009>
6. Fernández, A., García, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. Journal of Artificial Intelligence Research, 61, 863–905. <https://doi.org/10.1613/jair.1.11192>
7. Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? In Advances in Neural Information Processing Systems 35 (pp. 507–520). Curran Associates.
8. Jiang, K., Wang, W., Wang, A., & Wu, H. (2020). Network intrusion detection combined hybrid sampling with deep hierarchical network. IEEE Access, 8, 32464–32476. <https://doi.org/10.1109/ACCESS.2020.2973730>
9. Kasongo, S. M., & Sun, Y. (2020). Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. Journal of Big Data, 7(1), Article 105. <https://doi.org/10.1186/s40537-020-00379-6>
10. 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 30 (pp. 3146–3154). Curran Associates.
11. Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: Techniques, datasets and challenges. Cybersecurity, 2(1), Article 20. <https://doi.org/10.1186/s42400-019-0038-7>
12. Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. ACM Computing Surveys, 50(6), Article 94. <https://doi.org/10.1145/3136625>
13. Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. Applied Sciences, 9(20), Article 4396. <https://doi.org/10.3390/app9204396>
14. Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Advances in Neural Information Processing Systems 30 (pp. 4765–4774). Curran Associates.
15. Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 Military Communications and Information Systems Conference (MilCIS) (pp. 1–6). IEEE. <https://doi.org/10.1109/MilCIS.2015.7348942>
16. Moustafa, N., Slay, J., & Creech, G. (2019). Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. IEEE Transactions on Big Data, 5(4), 481–494. <https://doi.org/10.1109/TBDATA.2017.2715166>
17. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. Computers & Security, 86, 147–167. <https://doi.org/10.1016/j.cose.2019.06.005>
18. Sarhan, M., Layeghy, S., & Portmann, M. (2022). Towards a standard feature set for network intrusion detection system datasets. Mobile Networks and Applications, 27(1), 357–370. <https://doi.org/10.1007/s11036-021-01843-0>
19. Sharafaldin, I., Lashkari, A. H., & 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 (ICISSP 2018) (pp. 108–116). SciTePress. <https://doi.org/10.5220/0006639801080116>
20. Thakkar, A., & Lohiya, R. (2022). A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions. Artificial Intelligence Review, 55(1), 453–563. <https://doi.org/10.1007/s10462-021-10037-9>