Advanced Boosting Techniques for Intrusion Detection

CYSE 420 — Applied Machine Learning in Cybersecurity | Assignment 04
Carl Lochstampfor Jr. | Fall 2026

1. Introduction and Dataset

This assignment implements Gradient Boosting for network intrusion detection, tunes its hyperparameters under cross-validation, and evaluates the result on a held-out test partition using ROC-AUC and a confusion matrix. Random Forest and AdaBoost were also fitted at default settings for context.

The dataset, `Intrusion_data.csv`, is the NSL-KDD training subset: 25,192 connection records described by 41 features and a binary label distinguishing normal traffic from anomalous. The features cover basic connection fields such as `duration` and `src_bytes`, content fields such as `num_failed_logins`, and time-based and host-based traffic statistics such as `serror_rate` and the `dst_host_*` block. Three features are categorical: `protocol_type` (3 levels), `flag` (11 levels), and `service` (66 levels).

Inspection found no missing values, no duplicate records, and no identifier column, so neither the label-leak nor the hidden-duplicate problem affecting file-based malware datasets applies. The distribution is 53.4 percent normal to 46.6 percent anomalous, a majority-class baseline accuracy of 0.5338. Because the classes are near balanced, accuracy is a meaningful headline metric and no imbalance correction such as SMOTE or class weighting was warranted.

2. Method

Preprocessing. All transformations were assembled into a scikit-learn `Pipeline` so that every step estimating parameters from data is fitted only on training data. The three categorical features were one-hot encoded, expanding 41 features to 118; one-hot rather than integer encoding because none has a meaningful ordering. `handle_unknown='ignore'` guards against a rare `service` level appearing during transformation that was absent when the encoder was fitted, a real risk since five of the 66 levels occur fewer than ten times. The 38 numeric features were passed through unscaled: gradient boosting splits on threshold comparisons, so standardization changes nothing despite `src_bytes` spanning far wider magnitudes than the rate features. A zero-threshold variance filter after the encoder removed four constant columns — `land`, `urgent`, `num_outbound_cmds`, and `is_host_login` — leaving 114.

Partitioning. The data was split 70/30 with stratification on the label and a fixed random seed, yielding 17,634 training and 7,558 test records, each holding the 46.6 percent anomaly rate. The split precedes every fitted object in the pipeline, which matters because three components estimate parameters from data — encoder, variance filter, and classifier — and each is refitted inside every cross-validation fold.

Tuning. A grid search over 27 combinations was run with five-fold stratified cross-validation on the training partition only: `learning_rate` at 0.05, 0.1, and 0.2; `n_estimators` at 50, 100, and 200; and `max_depth` at 2, 3, and 5. Learning rate and estimator count are tuned jointly because they trade off directly — a lower rate makes each tree contribute less and needs more trees to compensate. Accuracy, F1, and ROC-AUC were recorded, with ROC-AUC selecting the final model. The search ran 135 fits in 37 seconds. The test partition was used exactly once, after the winning configuration was refitted on the full training set.

3. Results

Tuning produced almost no improvement. Untuned Gradient Boosting at scikit-learn defaults reached a cross-validated ROC-AUC of 0.9996 with a fold-to-fold standard deviation of 0.0002. The best of the 27 tuned configurations — learning rate 0.05, max depth 5, 200 estimators — reached 0.99984, a gain of roughly 0.0002. Across the entire grid ROC-AUC spanned only 0.9961 to 0.9998. The metric is effectively saturated, and the gap between best and worst configurations approaches the noise introduced by fold assignment.

ROC-AUC and F1 disagreed on which configuration was best. The grid selected on ROC-AUC, but the configuration using learning rate 0.20 at the same depth and estimator count ranked only third on that metric while posting the highest F1 (0.9980) and the highest accuracy (0.9981) of any combination tested.

The metrics measure different things: ROC-AUC grades how well the model ranks anomalies above normal records across all thresholds, while F1 grades the decision made at 0.5. F1 also spanned 0.9734 to 0.9980 across the grid, roughly seven times the range of ROC-AUC, making it the more discriminating criterion where both sit near ceiling.

On the held-out test set the tuned model reached accuracy 0.9968, precision 0.9986, recall 0.9946, F1 0.9966, and ROC-AUC 0.9998 — correctly passing 4,030 of 4,035 normal connections and catching 3,504 of 3,523 intrusions, for 5 false positives and 19 false negatives.

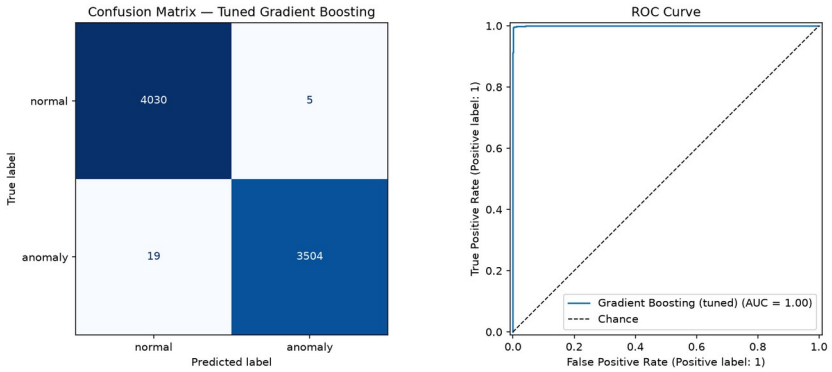

Figure 1. Confusion matrix and ROC curve for the tuned Gradient Boosting model on the held-out test set. The legend rounds to two decimals; the actual ROC-AUC is 0.9998.

The 19 missed intrusions share a common structure. All 19 are TCP connections, and 14 of the 19 carry the flag SF, indicating a normally completed connection. Their services are `http` (4), `ftp_data` (4), `telnet` (3), `smtp` (3), `ftp` (2), and one each of `time`, `X11`, and `finger`. None is a scan or a flood. Attacks with structural signatures are trivially separable: flag `S0`, a SYN with no reply and the signature of port scanning and SYN flooding, appears in 6,929 anomalous records against 80 normal ones. What the model misses are attacks that complete a well-formed session on an ordinary service port and carry their payload inside it — the remote-to-local and user-to-root families.

The failure is not a threshold artifact. Twelve of the 19 received a predicted anomaly probability below 0.11 and none exceeded 0.398; lowering the cutoff to 0.4 would recover one of the nineteen. The model is confidently wrong on these records rather than undecided.

Feature importance is extremely concentrated. `src_bytes` alone accounts for 72.8 percent of total Gini importance. Adding `service_ecr_i` (7.0 percent) and `service_http` (5.3 percent) brings the top three to roughly 85 percent, the top ten reach 96.8 percent, and 49 of the 114 features are never used for a split. A model weighting payload size this heavily separates floods and scans easily and has little signal to bring to a telnet session carrying a few hundred bytes of password guesses, which is exactly the error pattern above. `service_ecr_i` is the ICMP echo reply service associated with the smurf attack, among the most represented attack types here.

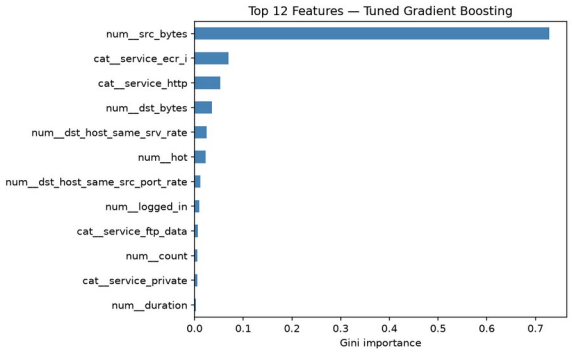

Figure 2. Top twelve features by Gini importance. Note the single dominant predictor.

Against the two comparison models, untuned Random Forest matched the tuned Gradient Boosting model, and AdaBoost did not.

Model	Accuracy	Precision	Recall	F1	ROC-AUC	FP	FN
Gradient Boosting (tuned)	0.9968	0.9986	0.9946	0.9966	0.9998	5	19
Random Forest (default)	0.9971	0.9989	0.9949	0.9969	0.9999	4	18
AdaBoost (default)	0.9774	0.9824	0.9688	0.9756	0.9976	61	110

A one-record difference across 7,558 test samples is noise, not a performance gap: an untuned bagging ensemble matched a tuned boosting ensemble here. AdaBoost is the only genuine separation, with roughly six times the false negatives of either other model. Its weak learners are decision stumps, single-split trees, and this problem requires depth a single split cannot express — every depth-5 configuration in the grid outranked every depth-3 configuration, and the selected model uses depth 5.

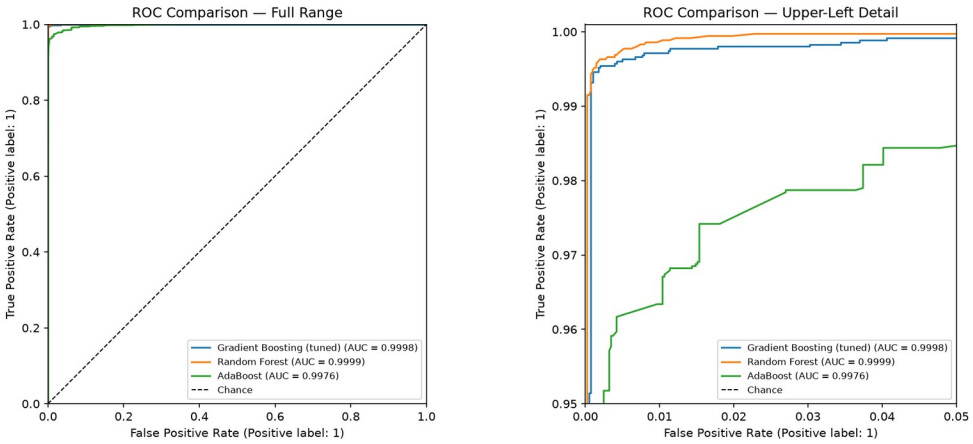

Figure 3. ROC curves for all three ensembles, shown at full range and zoomed to the upper-left corner. The full-range panel provides context; the zoom carries the comparison, where AdaBoost's coarser stepwise curve is clearly separated from the other two.

4. Limitations and Conclusion

NSL-KDD derives from a 1998 DARPA traffic capture and is partly synthetic. A ROC-AUC of 0.9998 therefore characterizes the benchmark rather than deployment performance; production intrusion detection does not achieve these rates on live traffic. The error structure suggests why: the attack classes this model handles effortlessly are those with distinctive packet-level signatures, which conventional signature-based detection already catches. The attacks it misses — those completing a normal-looking session on a normal-looking port — are the ones a machine learning approach was meant to add value against.

Two limitations follow from the method. Tuning moved the metric by 0.0002, so the tuned configuration cannot be claimed as meaningfully superior to the default; and the comparison models were left untuned deliberately, so the table compares a tuned boosting ensemble against untuned baselines rather than their own best configurations. Because importance concentrates in a single field, the model is also plausibly fragile to any shift in payload-size distribution, whether from changing legitimate traffic or deliberate evasion.

The conclusion is that on this dataset the choice of algorithm family matters more than hyperparameter tuning — boosting with sufficient depth and bagging both succeed, boosting restricted to stumps does not — and that the informative result is not the aggregate score but the structure of the errors that remain.