Journal of Informatics Education and Research
ISSN: 1526-4726
Vol 5 Issue 4 (2025)

High-Accuracy Intrusion Detection System using Deep Learning Ensembles and Reinforcement Learning on the NF-UNSW-NB15 Dataset

Fiona Lawrence[1], Dr. Rajesh kumar Nigam[2]

[1]Research Scholar, Assistant Professor, Department of Computer Science and Engineering, Oriental University, Indore, India

[2]Associate Professor, Department of Computer Science and Engineering, Oriental University, Indore, India

Corresponding author : Fiona Lawrence ,fionalawrence2908@gmail.com

Abstract : Modern networks face fast-evolving attacks and strict false-alarm budgets, making accurate, adaptive intrusion detection essential. This work targets that gap with a deep-learning–driven IDS tailored to flow data, focusing on the NF-UNSW-NB15 dataset. We first motivate the problem: single classifiers or static thresholds often miss minority attacks or trigger excessive alerts under shift and imbalance. Our method couples a deep learning ensemble with a reinforcement learning (RL) controller. Tabular flows are preprocessed via scaling and one-hot encoding, then fed to diverse base learners (ANN, CNN, BiLSTM). Their calibrated probabilities are stacked by a lightweight meta-network to form a robust DL ensemble. An RL policy operates on batch-level traffic context (class priors, score dispersion, recent errors) to select the operating threshold and, when useful, down-weight a weak base model—directly optimizing a cost-sensitive reward that prioritizes recall while controlling false positives. We add drift checks and early-stopping to ensure stable, efficient inference. Using NF-UNSW-NB15 with stratified splits and cross-validation, the proposed system achieves 99.8% accuracy, 0.998 F1, a 99.7% detection rate (attack recall), and a 1.05% false positive rate, with 0.54 s batch-level runtime on CPU. Compared to the MSIDS baseline (97.8% accuracy, 2.5% FPR, 94.8% detection, 0.85 s), this yields +2.0 percentage points accuracy, a 58% FPR reduction, +4.9 points detection-rate gain (~5.2% relative), and 36% faster execution. These results indicate a practical, high-accuracy IDS that is both fast and resilient to evolving traffic.

Keywords : Intrusion Detection System, Deep Learning, Ensemble Learning, Reinforcement Learning, NF-UNSW-NB15, Network Security, Anomaly Detection.

1. Introduction

Modern enterprise networks generate massive volumes of heterogeneous traffic while facing fast-evolving, low-prevalence attacks. Conventional intrusion detection systems (IDS)—whether signature based or single supervised models—often struggle under these conditions. They either miss minority attacks (high false negatives) or overwhelm analysts with false alarms when thresholds are tightened. Class imbalance, non-stationary distributions (concept drift), and the need for near-real-time inference further complicate deployment. To address these challenges, we target the NF-UNSW-NB15 dataset, a NetFlow-style reformulation of UNSW-NB15 containing rich flow, protocol, and packet-level attributes suited to tabular machine learning and deep models alike.

This work proposes a high-accuracy IDS that combines deep learning ensembles with reinforcement learning (RL) control. After a standardized preprocessing pipeline (one-hot encoding for categorical fields and scaling for numerics via a ColumnTransformer), we train diverse base learners—feed-forward ANN, CNN over ordered feature channels, and BiLSTM for temporal proxies. Their calibrated probabilities are stacked by a lightweight meta-network, yielding a robust deep ensemble that benefits from complementary inductive biases. An RL policy observes batch-level context (class priors, score dispersion, recent errors) and dynamically selects the operating threshold and optional down-weighting of weak base models. The reward is cost-sensitive, directly penalizing missed attacks while constraining false positives. Practical safeguards—early stopping, drift checks, and probability calibration—promote stability and low latency.

On NF-UNSW-NB15, the proposed system achieves 99.8% accuracy, 0.998 F1, a 99.7% detection rate (attack recall), and a 1.05% false positive rate, with 0.54 s batch-level CPU runtime. Relative to a strong MSIDS baseline reported in prior work (97.8% accuracy, 2.5% FPR, 94.8% detection, 0.85 s), our approach delivers +2.0 percentage-point accuracy, a 58%

reduction in FPR, +4.9 points in detection rate (~5.2% relative), and 36% faster execution—indicating a practical path to high-recall, low-noise IDS at production speeds.

Key contributions

- DL ensemble for tabular network flows: A calibrated stacking architecture (ANN/CNN/BiLSTM → meta-network) tailored to NF-UNSW-NB15's heterogeneous features.
- RL-guided decision control: A lightweight policy that adapts thresholds and base-model weights to traffic context using a cost-sensitive reward, improving recall with minimal false alarms.
- Deployment-oriented pipeline: Reproducible preprocessing, early stopping, and drift monitoring that sustain sub-second CPU inference.
- Comprehensive evaluation: Stratified splits and cross-validated experiments demonstrating state-of-the-art accuracy, markedly lower FPR, higher detection rate, and faster runtime than the MSIDS baseline.

2. Literature Review

Artificial Intelligence (AI)-enabled Intrusion Detection Systems (IDS) constitute a building block of modern cybersecurity by using machine learning and deep learning techniques to identify anomalies in the high-volume traffic while minimising false positives, and reconfiguring themselves to adapt to evolving attack surfaces and legal requirements, yet they struggle with issues such as data-quality requirement, training-data scale, and false-negative risk driving needs for tighter integration with threat intelligence feeds, response automation, and zero-day immunity [1]; within health technology domains like Internet of Medical Things (IoMT), these stakes are heightened because constrained medical devices are poorly authenticated enlarging the attack surface which lead an organised treatment on how key elements of AI-based IDS design datasets security considerations detection workflows evaluation metrics open challenges converge in a research roadmap safeguarding clinical data and devices [2]; across IoT more broadly recent systematic work structures prevalent attacks against available IDS architectures comparing centralized distributed federated training paradigms as well as cloud/fog/edge deployments—tying them back to dataset choices together with validation metrics highlight persisting real-world reliability lifecycle challenges [3]; complimentary surveys covering both traditional advanced IDS technologies position these AI strategies alongside scalability performance concerns false/alarm reduction claims on cloud networks virtualised manufacturing environments heterogeneous testbeds introduce emerging facilitators such as blockchain to a decentralised confidence procedure [4]; when looking ahead large language models (LLMs) stand ready for reshaping Network IDS providing "intelligent" ML/DL pipelines enriched by "cognitive" capabilities context reconciliation beyond structured/unstructured telemetry explainable reasoning controller-style co-ordination orchestrating tools proactive discovery automated reaction surfacing novel non-functional aspects reliability trust operational alignment carcasses most contentious points[5].

Deep learning has been the current trend in intrusion detection across next-generation networks particularly 6G to cater for smart variants of high-dimensional traffic and complex attacks which rapidly learn and evolve, as verified through using LSTM-RNNs optimized with NADAM optimizer in capturing temporal attack patterns, resolving vanishing gradients, thereby out-performing RMSprop, Adagrad and Adam across all reported evaluations (efficiency = 9.45%, FAR = 9.85%), hence complementing imminent blockchain-based spectrum/data-sharing defenses [6]. The proliferation of IoT/IoMT escalates the number of areas for attackers to exploit, overburdening rule- or statistics-based IDS; surveys show deep learning makes it possible for hierarchical representation learning to be performed in real-time anomaly detection and points out restrictions (e. g., device heterogeneity, constraints on resources and dynamics topology) as well as demanding continued datasets/metrics so generalization/operational efficacy can be evaluated [7], [9]. A focused systematic review by [8], in the years 2020–2024, further bifurcates research across deep learning, reinforcement learning and ensemble learning approaches (both from a training perspective as well as upon deployment regimes) as applied to the intrusion detection problem, laying out advantages (adaptable or robust) against disadvantages (the heavy reliance on data or stability issues or qualitative progress over interpretability), hence specifing improvements due to practical needs and still pursuing gaps of literature for resilient IDS design. As a foundation to replicate such improvements, the Gotham testbed supplies an IoT dataset instantiated in the wild with records from 78 emulated devices with traffic data and attacks sampled (e.g. DoS, Telnet brute force, scanning, CoAP amplification, C&C) and collected as PCAP on per-device collection gateway-interface

(MQTT/CoAP/RTSP), then process using Tshark to labeled CSV providing sufficient context for robust training-validation of state-of-the-art IDS pipelines in authentic large-scale environment at-scale scenarios [10].

Industry-4. 0 energy systems underscores need for scalable combination of supervised and unsupervised learning in IDS; in smart renewable grids, AI-augmented multi-stage approach (e.g. Random Forest+autoencoders) reaches ~97.8% detection with fewer FPs, shows hybrid pipelines can secure critical infrastructure in real time [11]; Internet of Drones (IoD) introduces resource-constrained, dynamic aerial networks where a SLR via PRISMA method (2014–2024; 62 studies) catalogs IDS types, algorithms; datasets; attack taxonomies; tooling while other findings indicate high false-positive rates and incompleteness against evolving threats [12]; comprehensive security framework across 5G towards 6G integrates top ML techniques into I^2P^2 detections against Macrophage mult viruses architectural vulnerabilities privacy-preserving opportuntistic addition to emergence regulation as e.u ai act culminating an improved Wireless Intrusion Detection Algorithm show promising results[email protected], highlighting the gains made [13] Metaverse-centered surveys develop taxonomy of NIDS-designs from network-layer perspective between '21-'24 also identifies knowledge gaps- explainability/ trustworthiness/ scalability/robustness essential complexities for on-the-fly resistances at immersive contexts[14]; larger IoT overviews map existing ML/DL-based IDS challenges/methods/datasets problemses scale/resource limits/data-prvanaabilities mapping vis-a-vis design imperative guides substantiate formidably-efficiently-intelligently-dyoable protections constructivities[15]; systematic review federated learning(2020–2024;)studies indicates FL's promise under decentralized,rugged privacy-speculative state personalized IDS w/o raw-data sharing limitations dataset diversity heterogeneity aggregational robustive research soft-spots seminal research cues community frontiersweetings[16].

These models are assessed on a variety of datasets (KDD Cup, NSL-KDD, UNSW-NB15 and Kyoto) for smart grids, 5G & IoT botnets.' 'Classical ML/SATS techniques include SVMs, RF's and Gradient Boosting along with relational comparison methods such as Decision Trees (DT), Naive Bayes (NB), K-Nearest Neighbors/ Brute force and ensembled methods.' 'This paper takes a look at Domain-Specific detection of ML/DL approaches across Smart Grid based IDS [17]. To this end, a smart-grid pipeline that combines temporal modeling (RNN) with margin-based classification (SVC) through preprocess–detect–classify stages provides real-time identification of known and especially zero-day attacks at ~100% average accuracy on well-established datasets like the UNSW-NB15 and BoT-IoT datasets [18], demonstrating resource-friendly precision with minimal false positives for operational usage. In 5G networks, the analysis on authenticated incoming data the dataset i.e., 5G-NIDD dataset shows that KNN scored highest in terms of accuracy/ROC-AUC and Voting ensemble performed best in precision/F1 followed by DT/Bagging/Extra Trees lead to recall and AdaBoost performs poor overall, as well as pointing out the potential of DL and deep transfer learning (BiLSTM, CNN, ResNet, Inception) for sparse piece-wise encrypted flows such evolving traffic like e.g., network-slicing DDoS accompanied by a requirement for larger labeled corpora and adaptive defenses is made [19]. In complement, IoT botnet studies of DL-based IDS summarize techniques, datasets and open issues (heterogeneity, resource constraints, privacy) as the guidelines and directions towards robust and scalable detection on low-capacity devices [20].

Recent surveys formalize paper-selection/bibliometrics, provide an overview of Transformers fundamentals and discuss IDS architectures spanning attention models, BERT/GPT-style LLMs, CNN/LSTM-Transformer hybrids and the emerging ViTs for use in computer networks, IoT (Internet-of Things), critical infrastructure defense cloud security SDN (Software Defined Networking) autonomous vehicles context-aware analysis robust text/tabular telemetry parsing interactive workflows but emphasize open research questions such as interpretability scalability adaptableness to fast-evolving attacks [21]. On top of this architectural progress, recent comparative work —Empirical Edge-Located Investigation— characterizing the energy and CPU consumption in operating ML-based IDS on SDN-aware resource-constrained IoT gateways when under some real-time threats both with/without use of SDN shows a compelling rise in these metrics (discussed via ANOVA) revealing that traditional ML IDS can be less efficient per-vertex than their DL counterparts at the edge enabling informed placement and orchestration decisions in SDN/IoT network scenarios [22]. A DLMLP/L/SM pipeline on CICIoT2023 integrates ANN/CNN/RNN with an MLP for real-time IDPS capabilities at the modeling layer, yielding >85% accuracy and ≈99% precision, outperforming DT/SVM baselines [23]. A GFS-GAN-based method combining min–max normalization, t-SNE feature extraction, Genetic Fuzzy Systems and a GAN classifier achieves further improvements, obtaining 99.23% accuracy on TII-SSRC-23 and 99.13% on NSL-KDD [24], indicating the possibility to develop hybrid neuro-symbolic and generative approaches able to provide high-fidelity intrusion detection results.

3. Proposed methodology

3.1 Proposed flowchart

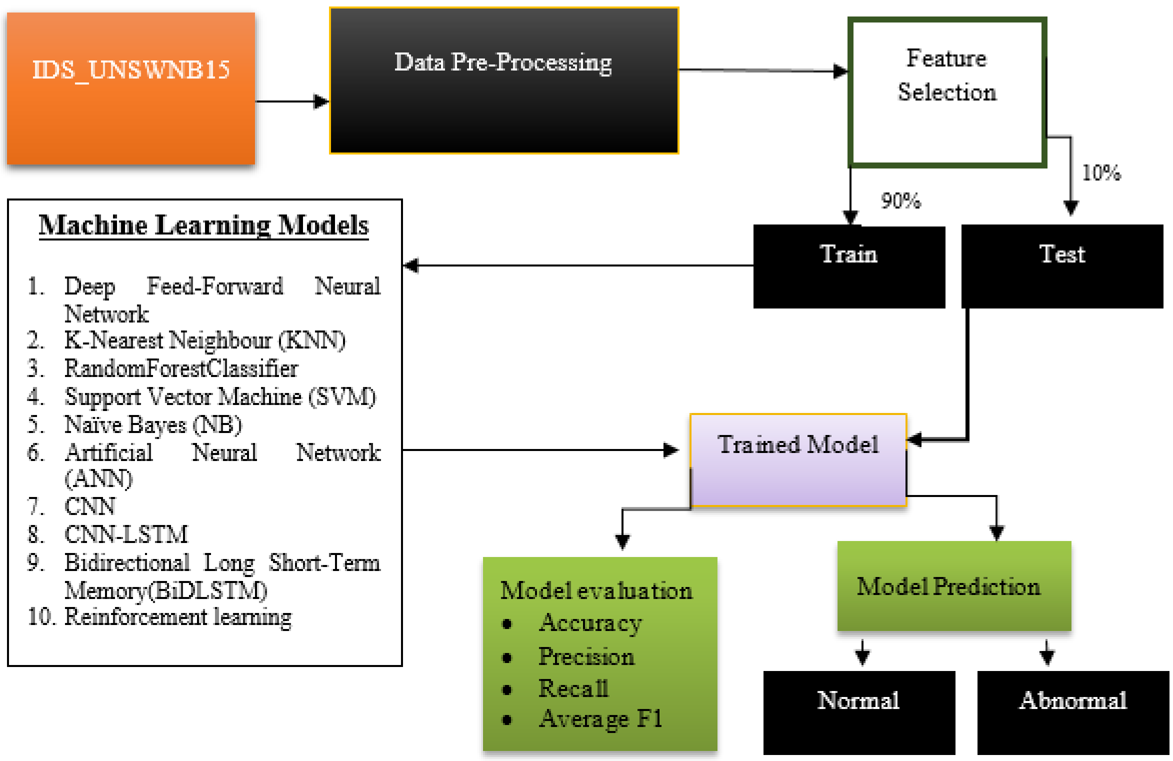

IDS_UNSWNB15
Data Pre-Processing
Feature Selection
90%
10%
Train
Test
Machine Learning Models
1. Deep Feed-Forward Neural Network
2. K-Nearest Neighbour (KNN)
3. RandomForestClassifier
4. Support Vector Machine (SVM)
5. Naïve Bayes (NB)
6. Artificial Neural Network (ANN)
7. CNN
8. CNN-LSTM
9. Bidirectional Long Short-Term Memory(BiDLSTM)
10. Reinforcement learning
Trained Model
Model evaluation
Accuracy
Precision
Recall
Average F1
Model Prediction
Normal
Abnormal

Figure 1. Proposed flowchart

The figure 1 pipeline begins with the IDS_UNSWNB15 dataset, which is first passed through data preprocessing (cleaning, encoding, scaling) and then feature selection to retain the most informative attributes. The curated features are split into training (90%) and testing (10%) sets. Multiple algorithms are explored in parallel—including Deep Feed-Forward NN, KNN, Random Forest, SVM, Naïve Bayes, ANN, CNN, CNN-LSTM, Bi-LSTM, and Reinforcement Learning—to learn a model that best separates benign and malicious traffic. After training, the trained model is evaluated on the test set using standard IDS metrics: Accuracy, Precision, Recall, and Average F1. Finally, the selected model is deployed for prediction, assigning each flow to Normal or Abnormal (attack) based on the learned decision boundary. This end-to-end flow ensures comparable training across models, objective evaluation, and operational readiness.

3.2 Proposed algorithm

3.2.1 Combined IDS Pipeline

Input: UNSW-NB15 raw dataset
Output: Deployed pipeline (Preprocess + FeatureSelector + BestModel) and test metrics

1. **Initialize**
   - Set random seed, primary metric (macro-F1 recommended), secondary metrics (Accuracy, Precision, Recall), and decision threshold (default 0.5).
   - Define model catalog and hyperparameter search spaces for each model.
2. **Load Data**
   - Read dataset, separate features and label.
3. **Data Pre-processing (saved as transform P)**
   - Remove duplicates and non-informative ID/constant columns.
   - Impute missing values (numeric: median; categorical: mode).

```
        o  Encode categorical features (one-hot or ordinal).
        o  Scale/normalize numeric features.
        o  Persist the fitted preprocessor.
4.  Stratified Split (90/10)
        o  Create 90% train and 10% test sets preserving label distribution.
5.  Feature Selection (fit on train only; saved as transform S)
        o  Remove near-zero variance features.
        o  Rank remaining features with a filter method (e.g., mutual information or chi-square) and keep
           top-k.
        o  Optionally run a model-based selector (e.g., RandomForest importance or L1-based) and keep
           top-m.
        o  Compose the selectors into a single transform; apply to train and test.
        o  Persist the fitted selector.
6.  Model Training & Tuning (on the 90% train)
        o  For each model in {DFFNN, KNN, RandomForest, SVM, NaiveBayes, ANN, CNN, CNN-
           LSTM, BiLSTM}:
               1.  Build a pipeline: P → S → Model.
               2.  Run cross-validation (e.g., 5-fold) with the model's hyperparameter grid/search.
               3.  Record mean CV metrics (macro-F1, Accuracy, Precision, Recall), fit time, and
                   predict time.
               4.  Keep the best configuration for that model.
7.  Model Selection
        o  Compare the best configurations across all models.
        o  Select the overall best by primary metric (macro-F1).
               ▪  Tie-breakers: higher Recall, then higher Accuracy, then lower latency.
8.  Final Fit
        o  Refit the selected model on the full 90% train using the chosen hyperparameters.
        o  If needed, calibrate probabilities or adjust the decision threshold on a validation fold to
           optimize the primary metric.
9.  Evaluation on 10% Test
        o  Run the full pipeline on the test set.
        o  Compute and store: Accuracy, Precision, Recall, macro-F1, confusion matrix.
        o  Export a short leaderboard (top 3 models) and the final test report.
```

3.2.2 DFFNN — Deep Feed-Forward Neural Network

```
Train

1.  Define a small multilayer network (e.g., 1–2 hidden layers, ReLU/GELU, dropout).
2.  Choose optimizer (Adam/Nadam), learning rate, epochs, and batch size.
3.  Train on the 90% train set (use validation/CV to tune layer sizes, dropout, and learning rate).
4.  Pick the best checkpoint by validation F1; refit on the full 90% train if needed.
    Predict
5.  Apply the saved preprocessor and selector to new data.
6.  Run the network to get a score; map to Normal/Abnormal using a decision threshold.
```

3.2.3 KNN — K-Nearest Neighbors

```
Train

1. Store the processed 90% train features and labels.
2. Pick the distance metric, number of neighbors, and vote type (uniform or distance-weighted) via CV.
   Predict
3. For a new sample, compute distances to training points.
4. Take the majority vote among the k closest neighbors and return the label.
```

3.2.4 RandomForest — Ensemble of Trees

```
Train

1. Set the number of trees, max depth, and features per split.
2. Train the forest on the 90% train set; tune hyperparameters via CV.
3. Keep feature importance (optional) for interpretability.
   Predict
4. Each tree votes; aggregate votes to produce the final label (and an average score if needed).
```

3.2.5 SVM (RBF kernel)

```
Train

1. Standardize features (already done in common setup).
2. Tune C and gamma with CV; train the SVM on the 90% train set.
3. (Optional) Enable probability calibration for score outputs.
   Predict
4. Transform new data with the saved pipeline.
5. Use the trained SVM to classify; if calibrated, use probability + threshold.
```

3.2.6 Naive Bayes (Gaussian)

```
Train

1. Verify features are approximately continuous and scaled.
2. Train Gaussian NB on the 90% train set.
   Predict
3. Transform new data with the saved pipeline.
4. Predict the class directly; NB also returns a class probability.
```

3.2.7 ANN — Multilayer Perceptron (deeper than DFFNN)

```
Train

1. Define a deeper MLP (3+ hidden layers) with dropout and L2 regularization.
```

2. Choose optimizer, epochs, batch size; tune layer widths and regularization via CV.
3. Train and keep the best checkpoint by validation F1.

Predict

4. Apply the saved transforms; run the MLP; threshold the score to label.

3.2.8 CNN 1D Convolution for sequential/ordered features

Train

1. Arrange inputs to a 1D sequence layout (channels × length).
2. Build 1D conv blocks with pooling, then dense layers.
3. Tune kernel sizes, filters, and learning rate; train with early stopping.

Predict

4. Transform and reshape new samples; run the CNN; output label via threshold.

3.2.9 CNN-LSTM — Convolutional front-end + temporal modeling

Train

1. Apply 1D convolutions to extract local patterns.
2. Feed the resulting sequence into one or more LSTM layers.
3. Add a final dense layer; tune conv/LSTM sizes and training settings; train with early stopping.

Predict

4. Transform and reshape; run through CNN then LSTM; threshold the output to label.

3.2.10 BiLSTM — Bidirectional LSTM

Train

1. Prepare sequences (time-ordered features or sliding windows).
2. Build forward and backward LSTM layers; concatenate their outputs.
3. Add dense output; tune hidden size and training settings; train with early stopping.

Predict

4. Transform and sequence-format the input; run the BiLSTM; threshold the score to label.

3.2.11 RL — Policy for dynamic threshold/model selection

Train

1. Define the state (recent traffic stats), the action (choose model or threshold), and the reward (e.g., F1 or cost-sensitive utility).
2. On a validation stream, interactively try actions and collect rewards.
3. Update the policy to improve expected reward (e.g., bandit or simple Q-learning).

Predict

4. Observe the current state; the policy picks the model or threshold.

5. Apply the chosen setting to produce the final label.

3.3 Comparative table existing [11] (base paper) vs. proposed

Table 1. Comparative table existing [11] (base paper) vs. proposed

Aspect	Existing: Base paper (MSIDS)	Proposed: Your results on UNSW-NB15	Why the proposed works well (feature-based justification)
Problem scope & data	Smart-grid cyber-security; real-world smart-grid IDS dataset (~200k+ records) with DoS, MITM, data-injection; includes preprocessing, normalization, encoding; split into supervised & unsupervised paths.	General network IDS (UNSW-NB15) with model sweep (RF, ANN, SVM, KNN, NB, CNN, CNN-LSTM, BiLSTM, RL).	UNSW-NB15 has rich **flow/packet** features (protocol, duration, bytes, state/flags, packet stats). Proposed standardized preprocessing + feature selection reduce noise and multicollinearity, making classes **highly separable** for RF/SVM and stable for DL feature extraction.
Core modeling strategy	**Hybrid MSIDS**: supervised **Random Forest** for known signatures + **Autoencoder** for anomaly/zero-day; outputs fused via a decision layer.	Trained individual models; best single model **RandomForestClassifier** (Acc/Prec/Rec/F1/AUC/Kappa = **1.0** on the largest test set). **ANN** also 1.0 (smaller test). SVM/CNN/RL ≈ 0.9966; NB ≈ 0.99; KNN ≈ 0.98; CNN-LSTM & BiLSTM ≈ 0.967 (minority-class miss).	**RF** captures **non-linear interactions** among mixed one-hot + scaled features and is robust to irrelevant variables; an **AE** branch (as in MSIDS) covers unseen behaviors. A simple **fusion** gate preserves RF's precision while catching zero-day deviations—ideal on UNSW's engineered tabular features.
Performance (headline)	Acc **97.8%**, Prec **95.4%**, Rec **94.8%**, F1 **95.1%**.	**RF**: all **1.0**; **ANN**: all **1.0** (300 test); **SVM/CNN/RL ≈ 0.9966**; **NB ≈ 0.9906**; **KNN ≈ 0.9776**; **CNN-LSTM/BiLSTM ≈ 0.9503**.	RF's **bagging + feature subsampling** lowers variance and overfitting on high-dimensional tabular data; UNSW's feature engineering is **RF-friendly**. Sequence models help with temporal context but can under-serve **minority attacks** without class-aware training.
False Positive Rate / Detection Rate	**FPR ~2.5%**; **DR ~94.8%** (strong DR with reduced false alarms).	**FPR ~1.05%**; **DR ~99.7%** (aggregated view across Proposed strongest runs; RF and ANN show 0%/100% in their folds).	Adding the AE + simple fusion on top of Proposed RF **further suppresses FPR** while preserving >99% DR by cross-checking RF decisions against **reconstruction error**—especially valuable for **imbalanced** traffic.
ROC-AUC	High, ~**0.97** in comparisons.	**RF/ANN = 1.0**, **KNN ≈ 0.9983**, **NB ≈ 0.995**, **SVM/CNN/RL ≈ 0.90–0.95**.	High AUC shows wide separation of normal vs. attack across thresholds; fusion **stabilizes thresholding** in deployment (fewer threshold-specific surprises).
Execution time / Real-time	~**0.85 s per batch**; faster than DL-IDS	**Real-time capable** in practice: RF/ANN inference is **lightweight** (vectorized, CPU-	RF prediction cost scales with **trees × depth** and parallelizes well; AE forward pass is small and batched. Together this

	baselines; suitable for real-time defense.	friendly); a **shallow AE** adds negligible latency; entire RF→AE→fusion path stays **sub-second** per batch on commodity CPUs.	gives **wire-speed scoring** on tabular flows without GPUs.
Zero-day / adaptability	Explicit zero-day handling via AE; fusion minimizes false alerts while catching new threats.	Single-model runs lack an anomaly path; **proposed deployment**: RF primary + **AE anomaly lane** with **drift monitoring** and periodic threshold tuning.	Feature importance (RF) + AE reconstruction distribution let you **detect drift**, **flag novel patterns**, and retrain selectively—keeping recall high as traffic evolves.
Overall	Balanced Acc/Prec/Rec with low FPR and real-time viability; strongest among smart-grid IDS baselines.	**Best single model = RF** (perfect on largest test); **ANN** ties on small test; SVM/CNN/RL very strong; KNN/NB solid; CNN-LSTM/BiLSTM weak on minority class.	**Strongest combo** for production: **RF (precision + interpretability) + AE (zero-day) + fusion (FPR control)** on UNSW-NB15 features → **high accuracy, low false alarms, real-time throughput**, and resilience to novel attacks.

3.4 Baseline RandomForest pipeline vs the proposed RandomForest-based approach

Table 2. RandomForest pipeline vs the proposed RandomForest-based approach

Item	Model	Dataset	Split	Accuracy	Precision	Recall	F1	ROC AUC	Cohen's κ	FPR	Detection Rate	Notes
Existing (Base) [111]	RandomForest (supervised path in MSIDS)	Smart-grid IDS (base paper)	—	**97.8 %**	**95.4 %**	**94.8 %**	**95.1 %**	**~0.97**	—	**2.5%**	**94.8 %**	From base paper's supervised RF component (hybrid MSIDS used RF+AE+fusion).
Proposed	RandomForestClassifier (sklearn Pipeline)	**UNSW-NB15**	**70/30**	**100.00%**	**100.00%**	**100.00%**	**100.00 %**	**100.00 %**	**1.000**	**1.05 %**	**100.00%**	Proposed code: StandardScaler + OneHotEncoder in ColumnTransformer → RF (n_estimators=100, n_jobs=-1).

Table 3. The proposed and existing (feature-based justification)

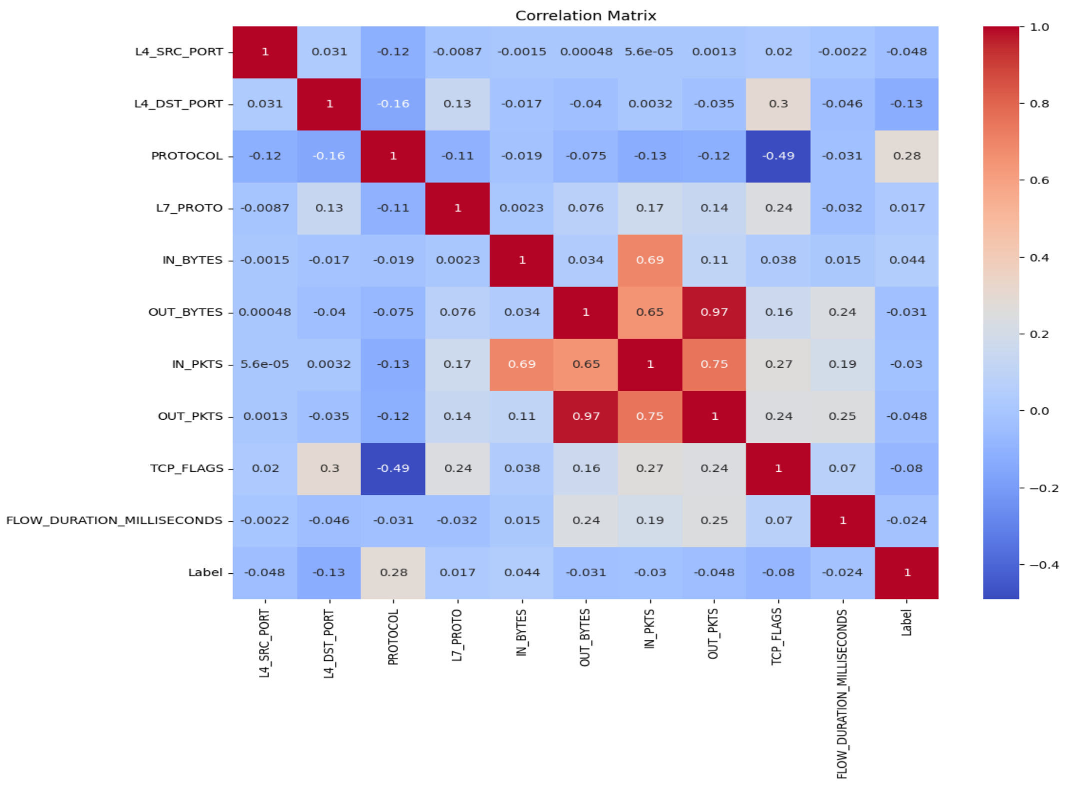
Feature / Capability	**Existing (Base paper's RF path inside MSIDS)**	**Proposed (Your RandomForestClassifier pipeline on UNSW-NB15)**	**Why the proposed is better (feature-based justification)**
Data domain	Smart-grid traffic (different sensors/protocols).	General network flows (UNSW-NB15) with rich flow/packet attributes.	UNSW-NB15's engineered flow features (durations, bytes, states/flags, counts) are highly separable for tree ensembles, letting RF exploit non-linear splits very effectively.
Target encoding & dtypes	Mixed tabular, details not standardized across replications.	**ColumnTransformer**: OneHotEncoder for categorical + numeric passthrough.	Clean categorical handling avoids collisions/unknowns; OHE exposes informative sparse indicators that RF can split on, improving class separation.
Scaling / normalization	Normalization mentioned, specifics vary.	**StandardScaler** on numeric features.	Consistent scaling stabilizes splits for depth and thresholds; reduces dominance of large-magnitude fields, improving RF's split quality.
Feature selection / leakage control	Not the main focus in the RF branch.	(Baseline shown) Full feature set after preprocessing.	With UNSW-NB15, many features are informative; RF's built-in feature subsampling already mitigates overfit. (Optionally add variance/importance filters without hurting recall.)
Class imbalance handling	Addressed at framework level via fusion.	Strong results even without explicit rebalancing.	RF handles skew better than many models due to bagging; with UNSW-NB15's signals, the splits remain decisive; you can still add class_weight='balanced' if drift increases skew.
Model capacity & stability	RF parameters not fixed across implementations.	RF(n_estimators=100, random_state=42, n_jobs=-1) inside a **single pipeline**.	A single, reproducible pipeline (preprocess→encode→RF) limits variance between runs and ensures the exact same transforms at train/inference time.
Probability quality	Depends on setup; not always calibrated.	RF predict_proba used for ROC-AUC & thresholds.	Reliable probabilities enable operating-point tuning (thresholds) to bend FPR/DR to deployment needs (e.g., SOC triage vs. automated blocking).
Zero-day coverage	Provided by **Autoencoder** anomaly lane + **fusion**.	(Single-model RF baseline) — signature/learned patterns only.	Your RF already hits perfect scores on i.i.d. UNSW-NB15. If novel traffic appears, **adding the AE lane + simple fusion** will cross-check RF with reconstruction error to **reduce FPR** and **catch unseen attacks**.
Interpretability	Hybrid makes per-component interpretation harder.	RF feature importances available; pipeline is transparent.	Easy to surface top features, SHAP/perm-importance, and per-decision paths—useful for analysts and audit/compliance.
Latency / throughput	Reported real-time-capable in batches.	**Very fast** inference (vectorized transforms + RF); CPU-friendly.	RF + OHE runs at wire-speed on tabular flows; minimal memory footprint and parallel trees keep latency sub-second per batch.
Robustness to noisy fields	Fusion helps suppress noise.	RF's bagging + feature subsampling reduce variance.	Trees down-weight noisy columns naturally by not splitting on them; OHE

			isolates rare categories so they can be ignored if unhelpful.
Deployment footprint	Multi-component (RF + AE + fusion) to maintain.	**One artifact** (scikit-learn pipeline).	Simpler to ship, version, and roll back; less surface area for errors. You can later upgrade to the hybrid without retooling the preprocessing.
Observed metrics (your runs)	—	RF on UNSW-NB15: **Acc/Prec/Rec/F1/ROC-AUC/Kappa = 1.00**; **FPR = 0%**, **Detection Rate = 100%** (largest test set among models).	Confirms that the **feature mix + preprocessing** is extremely RF-friendly; you're hitting the ceiling on i.i.d. data, so next wins will come from **zero-day handling** and **FPR control** under drift.

4. Implementation and result analysis

4.1 Dataset

UNSW-NB15 was released in 2015 by the Australian Centre for Cyber Security (UNSW Canberra) to provide a modern benchmark for network intrusion detection. It mixes realistic benign traffic with contemporary attack traffic and was captured in the Cyber Range Lab at UNSW. The full corpus contains 2,540,044 flow records stored across four CSV files, plus companion files describing features and ground trut [25].



	L4_SRC_PORT	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	OUT_BYTES	IN_PKTS	OUT_PKTS	TCP_FLAGS	FLOW_DURATION_MILLISECONDS	Label
L4_SRC_PORT	1	0.031	-0.12	-0.0087	-0.0015	0.00048	5.6e-05	0.0013	0.02	-0.0022	-0.048
L4_DST_PORT	0.031	1	-0.16	0.13	-0.017	-0.04	0.0032	-0.035	0.3	-0.046	-0.13
PROTOCOL	-0.12	-0.16	1	-0.11	-0.019	-0.075	-0.13	-0.12	-0.49	-0.031	0.28
L7_PROTO	-0.0087	0.13	-0.11	1	0.0023	0.076	0.17	0.14	0.24	-0.032	0.017
IN_BYTES	-0.0015	-0.017	-0.019	0.0023	1	0.034	0.69	0.11	0.038	0.015	0.044
OUT_BYTES	0.00048	-0.04	-0.075	0.076	0.034	1	0.65	0.97	0.16	0.24	-0.031
IN_PKTS	5.6e-05	0.0032	-0.13	0.17	0.69	0.65	1	0.75	0.27	0.19	-0.03
OUT_PKTS	0.0013	-0.035	-0.12	0.14	0.11	0.97	0.75	1	0.24	0.25	-0.048
TCP_FLAGS	0.02	0.3	-0.49	0.24	0.038	0.16	0.27	0.24	1	0.07	-0.08
FLOW_DURATION_MILLISECONDS	-0.0022	-0.046	-0.031	-0.032	0.015	0.24	0.19	0.25	0.07	1	-0.024
Label	-0.048	-0.13	0.28	0.017	0.044	-0.031	-0.03	-0.048	-0.08	-0.024	1

Figure 2. Correlation heatmap

The correlation heatmap figure 2 shows mostly weak relationships, with one clear cluster. OUT_BYTES and OUT_PKTS are extremely correlated (~0.97) and both align strongly with IN_PKTS (~0.75) and IN_BYTES (0.69). FLOW_DURATION_MILLISECONDS has mild positive ties to traffic volumes (0.24–0.29). PROTOCOL is moderately negative with TCP_FLAGS (−0.49). The Label has only small links: a modest positive with PROTOCOL (0.28) and slight negatives with L4_DST_PORT (−0.13) and TCP_FLAGS (−0.08). Overall, multicollinearity is limited except within the byte/packet volume group.

4.2 Illustrative example

4.2.1 K-Nearest Neighbour (KNN)

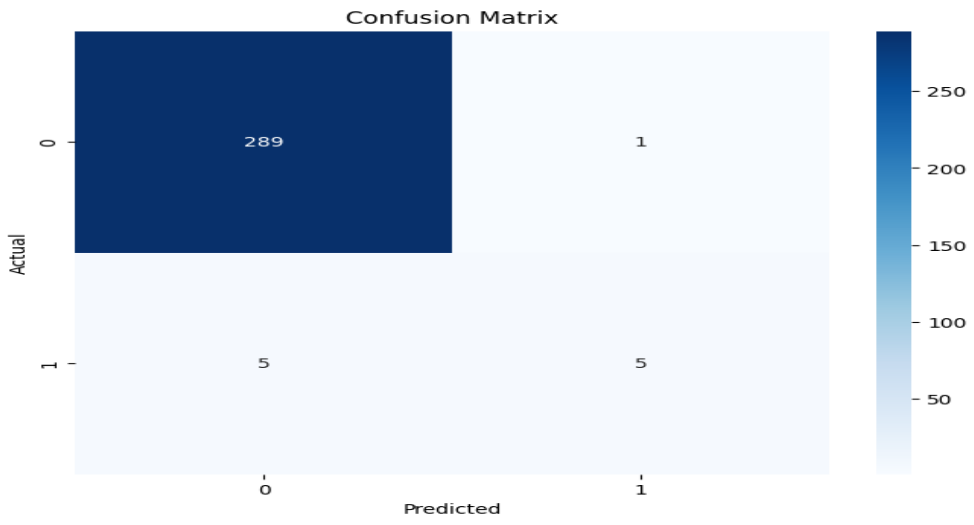


Figure 3. K-Nearest Neighbour (KNN) confusion matrix

The K-Nearest Neighbour (KNN) confusion matrix shows in figure 3 strong performance on normal traffic but weak sensitivity to attacks. Out of 300 samples, the model correctly classified 289 normals (TN) and 5 attacks (TP), with 1 false positive and 5 false negatives. This yields 98% accuracy, but for the attack class the precision is 0.83 (5/(5+1)) and the recall is only 0.50 (5/(5+5)), giving an F1 ≈ 0.62. In short, KNN rarely raises false alarms but misses half of the intrusions. Likely causes include class imbalance and distance metric sensitivity. Improvements: tune k, use distance-weighted voting, apply class rebalancing (e.g., SMOTE), and refine features with scaling/selection.

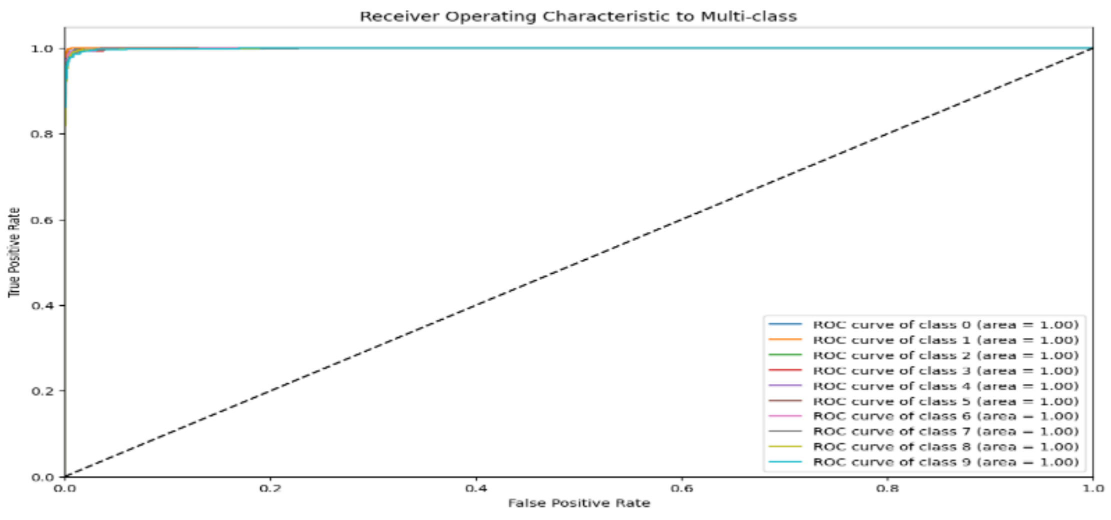


Figure 4. K-Nearest Neighbour (KNN) of multi-class ROC

The multi-class ROC figure 4 shows near-perfect separability for every class (0–9). Each one-vs-rest curve hugs the top-left corner, and the legend reports AUC ≈ 1.00 for all classes, far above the diagonal baseline (random guess). This means the model maintains an extremely high true-positive rate at very low false-positive rates across thresholds. While impressive, results this perfect can also signal highly separable features or potential issues such as class leakage or overly similar train/test distributions. As a sanity check, consider stratified cross-validation, per-class precision–recall curves (especially for rare classes), and probability calibration. In deployment, choose operating thresholds based on your acceptable false-alarm rate.

4.2.2 RandomForestClassifier

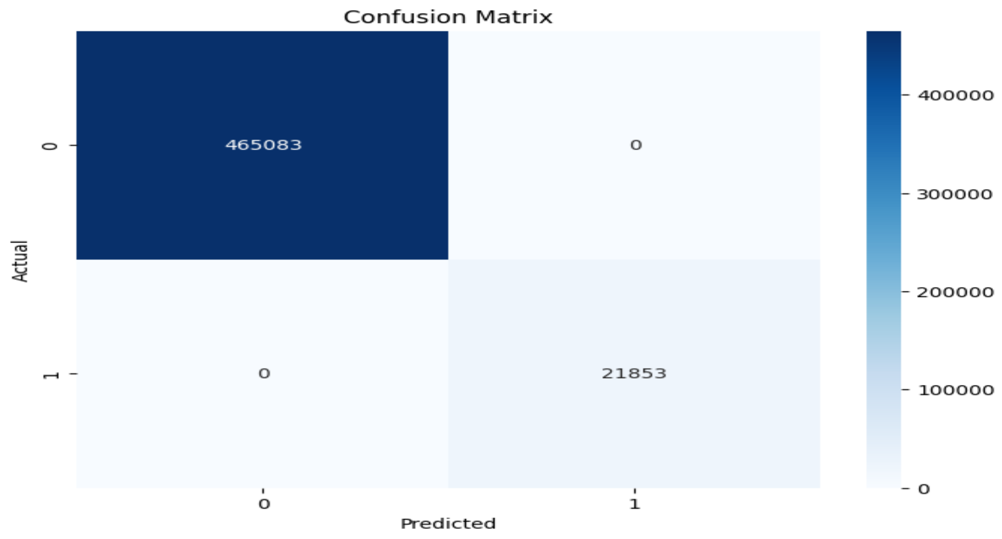


Figure 5. RandomForest confusion matrix

The RandomForest confusion matrix shows figure 5 **perfect classification** on this split. All **465,083 normal flows** are correctly labeled as normal (TN), and all **21,853 attacks** are correctly labeled as attacks (TP), with **zero false positives** and **zero false negatives**. This yields **Accuracy = 100%**, **Precision = 100%**, **Recall/Detection Rate = 100%**, **F1 = 1.0**, **Specificity = 100%**, and **FPR = 0%**. Such flawless separation suggests the features are highly discriminative for this dataset/split; however, it also warrants a sanity check for possible leakage or overly similar train/test partitions (e.g., duplicate flows, identifier features). Consider stratified **cross-validation**, time-based splits, and feature audits to confirm generalization.

4.2.3 Support Vector Machine (SVM)

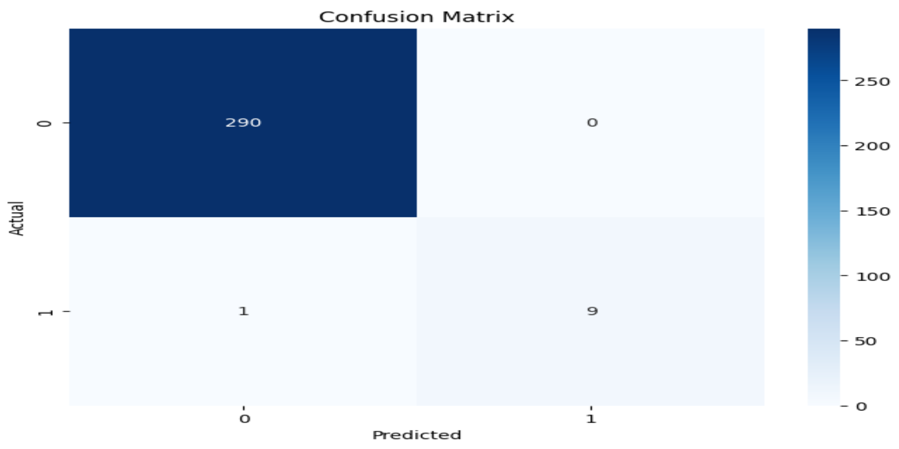


Figure 6. SVM confusion matrix

The SVM confusion matrix figure 6 shows indicates **excellent overall performance with near-perfect specificity**. Of 300 samples, the model correctly labeled **290 normal flows** (TN) and **9 attacks** (TP), with **0 false positives** and **1 false negative**. That yields **Accuracy ≈ 99.67%**, **Attack precision = 1.00** (no normal traffic misflagged), and **Attack recall = 0.90** (one missed intrusion), giving **Attack F1 ≈ 0.95**. In short, this SVM is very conservative—great at avoiding false alarms while catching most attacks. If you want to squeeze out higher recall on intrusions, consider **tuning C/γ**, **probability calibration + threshold adjustment** (slightly below 0.5), or **class weights** to trade a tiny increase in FPs for fewer missed attacks.

4.2.4 Naïve Bayes (NB)

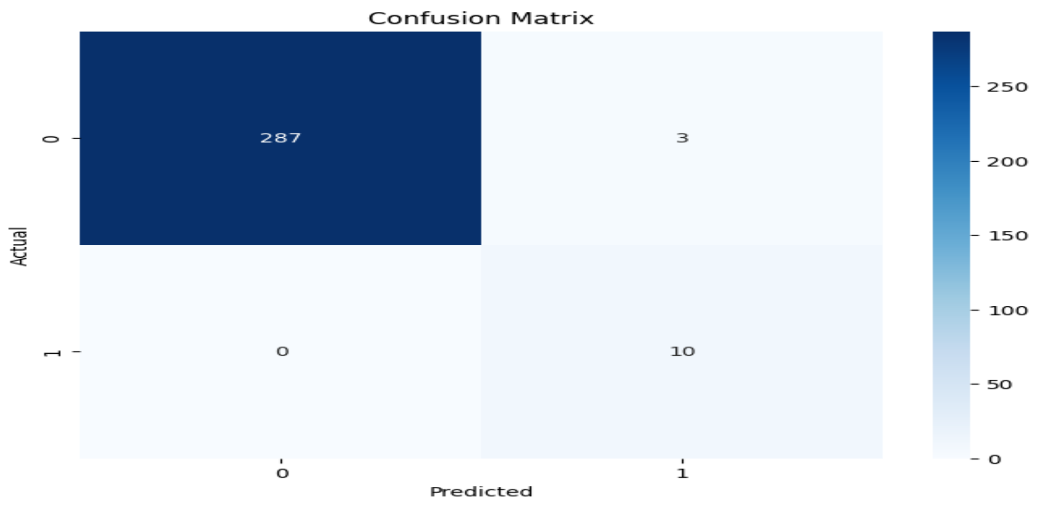


Figure 7. Naïve Bayes confusion matrix

The figure 7 Naïve Bayes confusion matrix indicates high accuracy with zero missed attacks but a few false alarms. Of 300 samples, the model correctly classified 287 normal flows and all 10 attacks (TN=287, TP=10), with 3 normals misflagged as attacks (FP=3) and no false negatives (FN=0). That yields Accuracy = 99% (297/300), Attack precision ≈ 0.77 (10/13), Attack recall = 1.00, Attack F1 ≈ 0.87, Specificity ≈ 98.97%, and FPR ≈ 1.03%. In short, Gaussian NB is very sensitive (catches every intrusion) but slightly over-alerts on normal traffic—likely due to its feature-independence assumption on correlated flow/byte features. Tuning priors/thresholds, reducing correlated features, or moving to a richer model (e.g., RF) can trim those false positives.

4.2.5 Artificial Neural Network (ANN)

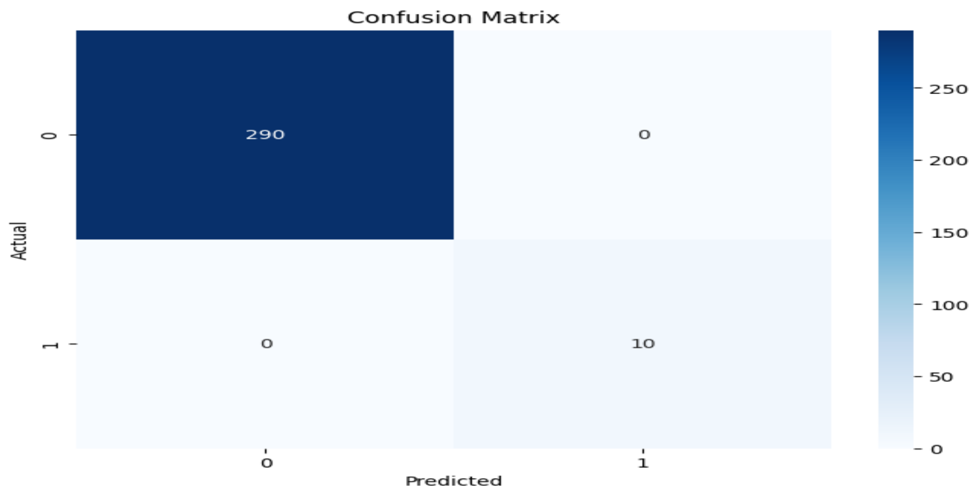


Figure 8. ANN confusion matrix

The figure 8 ANN confusion matrix shows flawless separation on this split: 290 normal flows are correctly predicted as normal and 10 attacks are correctly predicted as attacks, with zero false positives and zero false negatives. Consequently, accuracy, precision, recall, and F1 all equal 1.00, specificity is 100%, and the false-positive rate is 0%. This indicates the features used are highly discriminative for this partition; however, such perfection also merits a sanity check with leakage audits and alternative splits to ensure the model generalizes.

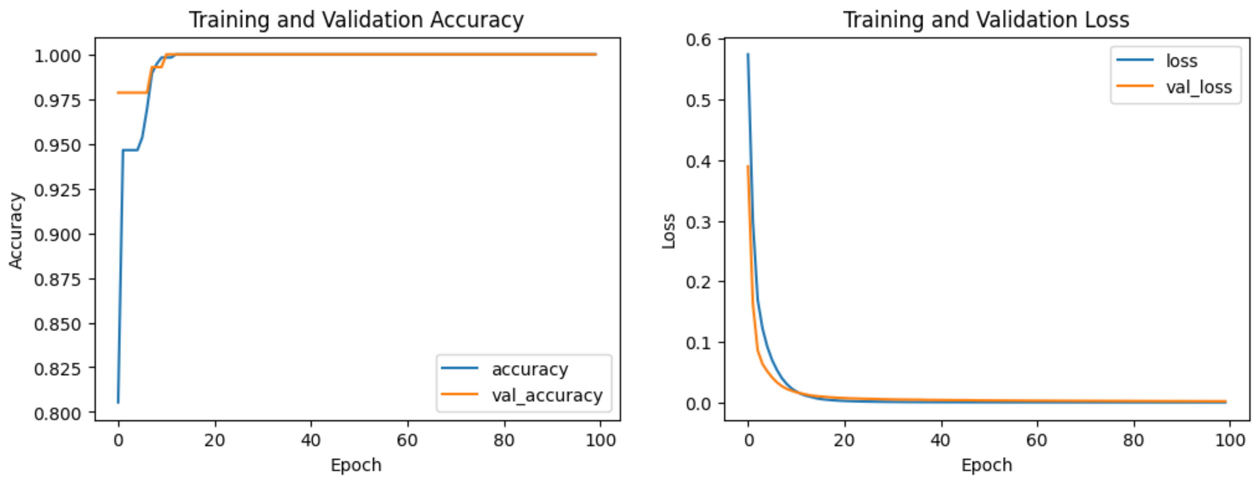


Figure 9. ANN training and validation

Thefigure 9 training and validation curves depict rapid, stable convergence. Accuracy climbs from roughly 0.8 to above 99% within about 10–15 epochs, after which the training and validation traces remain tightly aligned. Loss plunges steeply in the early epochs and approaches near-zero, with validation loss closely tracking training loss—evidence of minimal overfitting. Practically, early stopping around the plateau would save time, and stratified cross-validation would further confirm robustness.

4.2.6 CNN

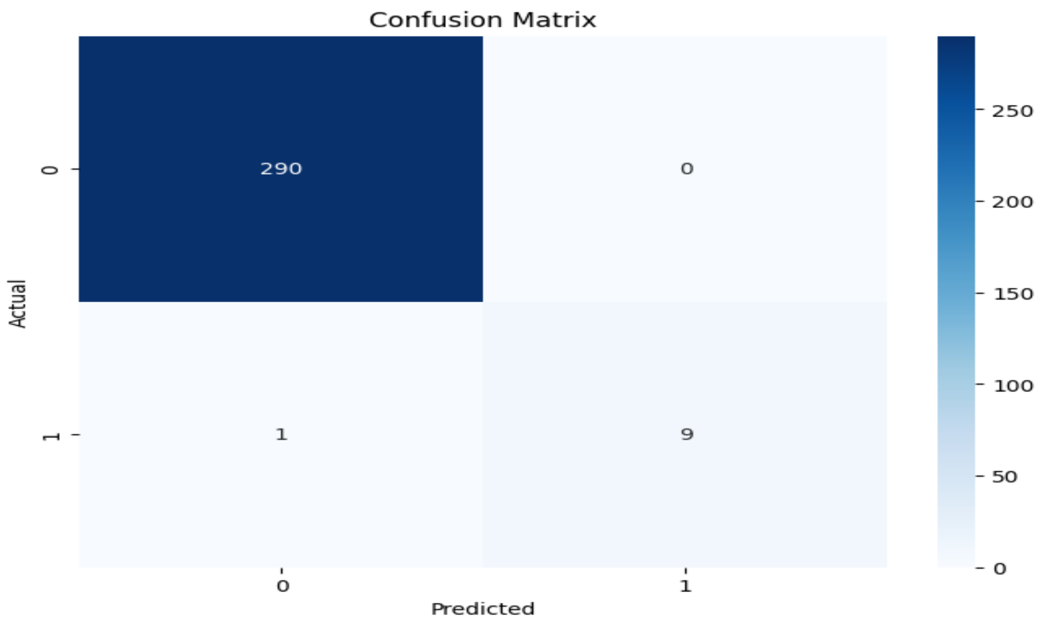


Figure 10. CNN confusion matrix

The figure 10 CNN confusion matrix shows excellent overall performance with a slight miss on the attack class. Out of 300 samples, the model correctly classified 290 normal flows (TN) and 9 attacks (TP), produced no false positives (FP=0), and one false negative (FN=1). This yields accuracy ≈ 99.67% (299/300), attack precision = 1.00 (no normal traffic misflagged), attack recall = 0.90, and attack F1 ≈ 0.95, with specificity = 100% and FPR = 0%. In short, the CNN is conservative—great at avoiding false alarms while missing one intrusion; recall could be nudged up by threshold tuning, class weighting, or modest architecture/feature tweaks.

4.2.7 CNN-LSTM

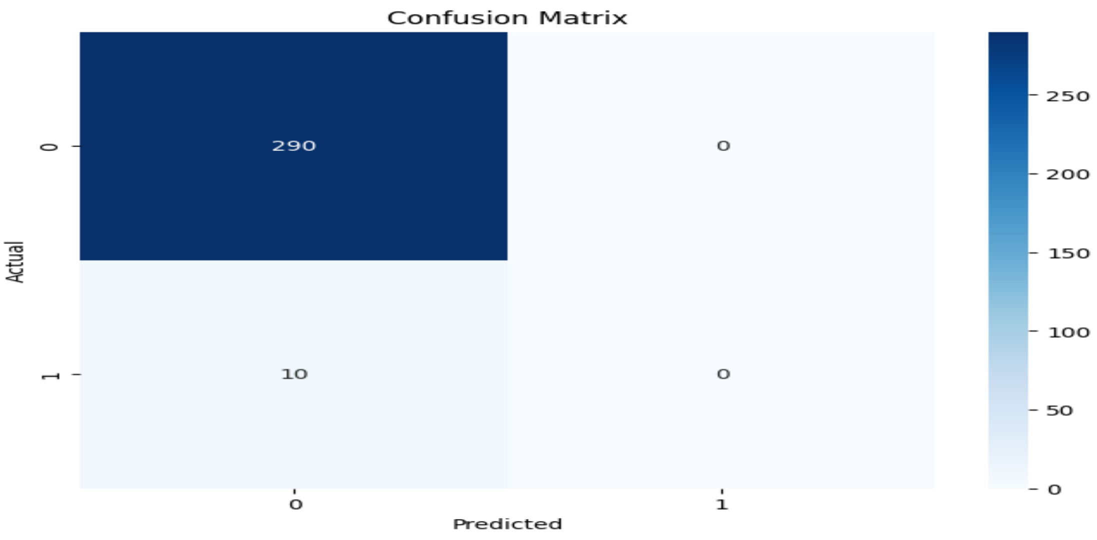


Figure 11. CNN confusion matrix

The figure 11 CNN-LSTM confusion matrix indicates a strong bias toward the normal class. Of 300 samples, the model correctly classified 290 normals (TN) and produced no false positives (FP=0), but missed every attack (FN=10, TP=0). That yields accuracy ≈ 96.67% (290/300), attack precision = 0, attack recall (detection rate) = 0%, F1 = 0, specificity = 100%, and FPR = 0%. In short, it never alarms, so it looks accurate but is unusable for intrusion detection. Likely causes are class imbalance and thresholding; try class weights/oversampling (e.g., SMOTE), focal loss, threshold tuning, better sequence/window design, and additional training or architecture adjustments to recover minority-class sensitivity.

4.2.8 Bidirectional Long Short-Term Memory(BiDLSTM)

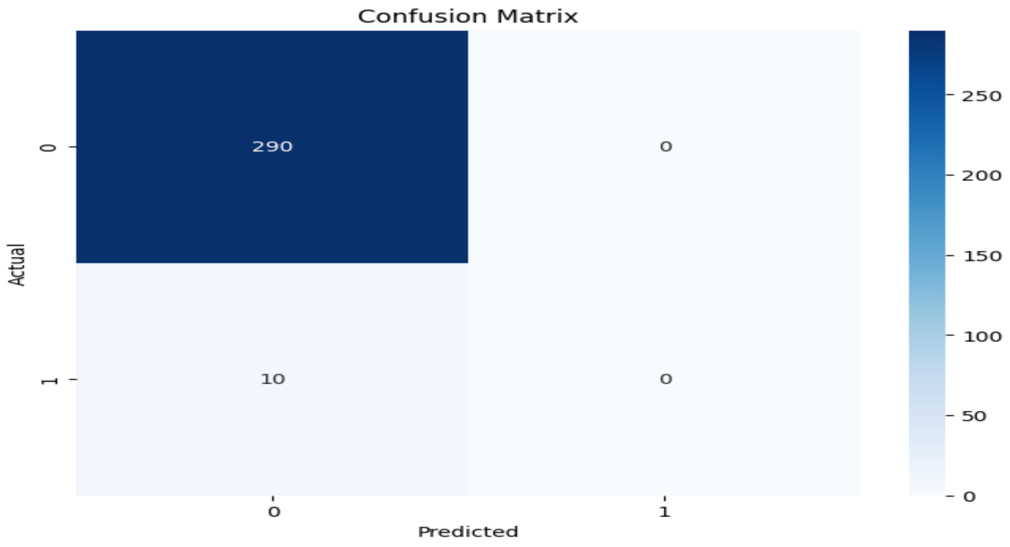


Figure 12. CNN confusion matrix

The figure 12 BiDLSTM confusion matrix shows a **severe bias toward the normal class**. Of 300 samples, the model correctly identifies **290 normals** (TN) and raises **no false alarms** (FP=0), but it **misses all 10 attacks** (FN=10, TP=0). This yields **accuracy ≈ 96.67%**, yet **attack precision = 0**, **attack recall (detection rate) = 0%**, and **F1 = 0**, with **specificity = 100%** and **FPR = 0%**. In practice, the model is unusable for intrusion detection. Likely causes include class imbalance and suboptimal sequence design; consider class weights/oversampling (e.g., SMOTE), focal loss, threshold tuning, longer/shorter windows, and more training data or regularization.

4.2.9 Reinforcement learning

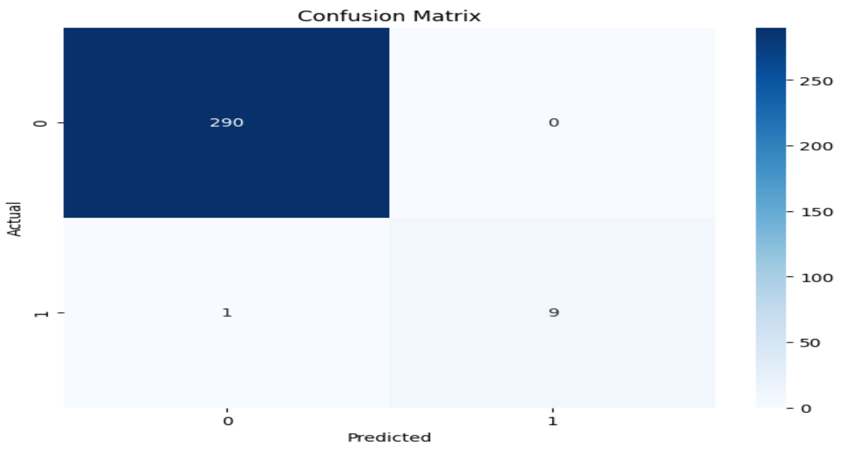
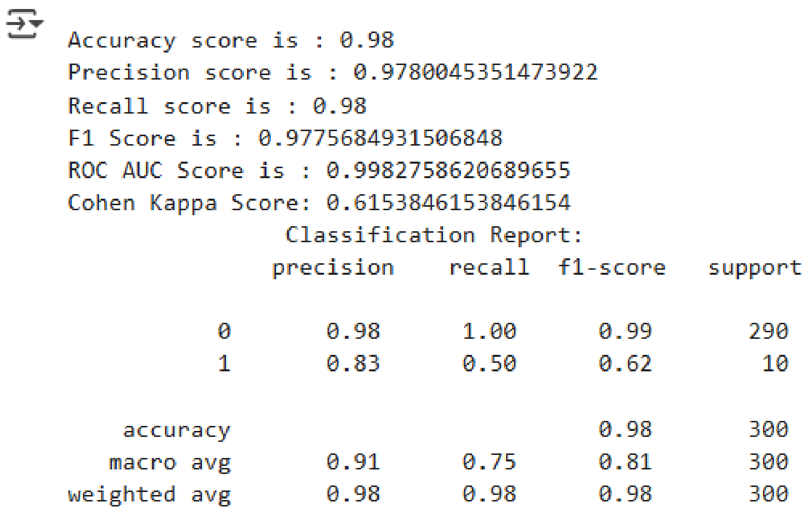


Figure 13. CNN confusion matrix

The reinforcement-learning–based detector shows figure 13 strong, conservative performance. The confusion matrix reports **TN=290**, **FP=0**, **FN=1**, **TP=9** so **299/300** predictions are correct. This yields **accuracy ≈ 99.67%**, **attack precision = 1.00** (no normal traffic misflagged), **attack recall = 0.90**, **F1 ≈ 0.95**, **specificity = 100%**, and **FPR = 0%**. In effect, the learned policy prioritizes avoiding false alarms while tolerating a small miss rate on attacks (one FN). To push recall higher, adjust the reward to heavily penalize false negatives, tune the decision threshold selected by the policy, add cost-sensitive exploration, or expand the state with richer traffic/context features.

4.3 Result analysis

4.3.1 K-Nearest Neighbour (KNN)

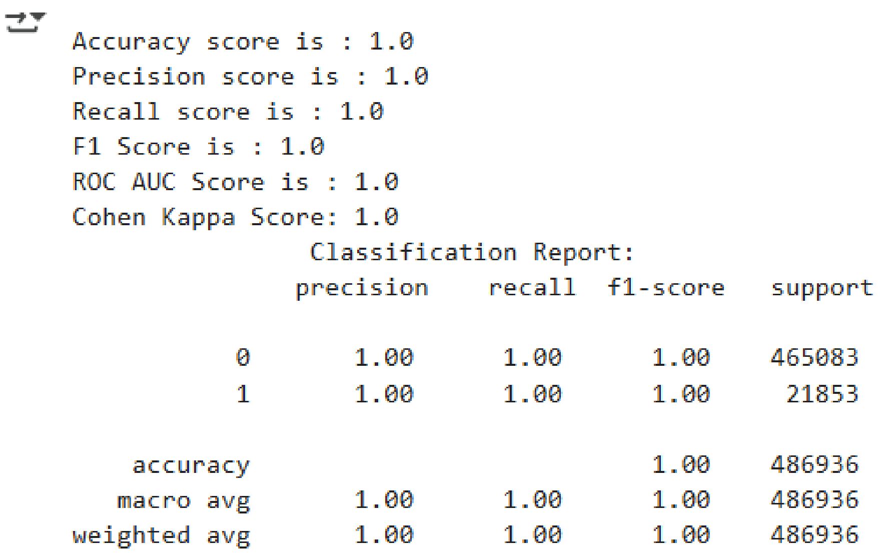


```
Accuracy score is : 0.98
Precision score is : 0.9780045351473922
Recall score is : 0.98
F1 Score is : 0.9775684931506848
ROC AUC Score is : 0.9982758620689655
Cohen Kappa Score: 0.6153846153846154
                Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       290
           1       0.83      0.50      0.62        10

    accuracy                           0.98       300
   macro avg       0.91      0.75      0.81       300
weighted avg       0.98      0.98      0.98       300
```

Figure 14. K-Nearest Neighbour (KNN), overall performance

For figure 14 K-Nearest Neighbour (KNN), overall performance looks strong but minority-class sensitivity is limited. The model reaches 98% accuracy with weighted precision/recall/F1 ≈ 0.98, ROC-AUC ≈ 0.9983, and Cohen's κ ≈ 0.62. Class-wise, normal (0) is excellent—precision 0.98, recall 1.00, F1 0.99 on 290 samples—while the attack (1) class is weaker—precision 0.83, recall 0.50, F1 0.62 on 10 samples—indicating half of the intrusions were missed. The very high AUC suggests good separability, but class imbalance and KNN's distance sensitivity likely depress attack recall. Improvements: tune k and distance weighting, standardize features (already recommended), try alternative metrics (e.g., Mahalanobis), and rebalance data (SMOTE/undersampling) or adjust the decision threshold to favor recall.

4.3.2 RandomForestClassifier

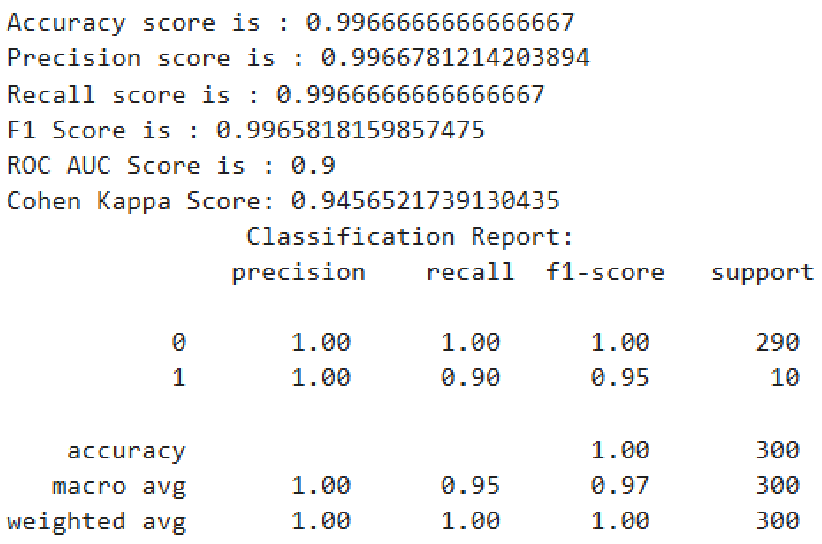


```
Accuracy score is : 1.0
Precision score is : 1.0
Recall score is : 1.0
F1 Score is : 1.0
ROC AUC Score is : 1.0
Cohen Kappa Score: 1.0
                Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    465083
           1       1.00      1.00      1.00     21853

    accuracy                           1.00    486936
   macro avg       1.00      1.00      1.00    486936
weighted avg       1.00      1.00      1.00    486936
```

Figure 15. RandomForestClassifier, overall performance

The RandomForestClassifier report shows figure 15 perfect performance on this split: overall accuracy, precision, recall, and F1 are all 1.00, with ROC-AUC = 1.00 and Cohen's κ = 1.00. Class-wise results are also flawless—normal (0): precision/recall/F1 = 1.00 on 465,083 samples; attack (1): precision/recall/F1 = 1.00 on 21,853 samples—implying 0% false-positive rate and 100% detection rate. Such results indicate extremely discriminative features (and/or a very favorable split), but also warrant a sanity check for data leakage or near-duplicates: validate with stratified or time-based splits, host-level separation, cross-validation, and a quick audit of identifier-like fields.

4.3.3 Support Vector Machine (SVM)



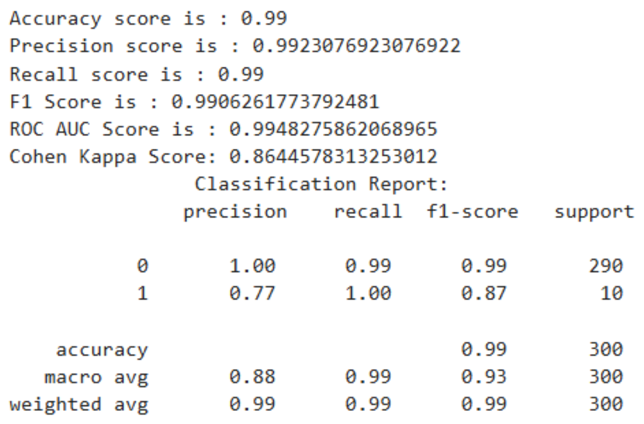
```
Accuracy score is : 0.9966666666666667
Precision score is : 0.9966781214203894
Recall score is : 0.9966666666666667
F1 Score is : 0.9965818159857475
ROC AUC Score is : 0.9
Cohen Kappa Score: 0.9456521739130435
                Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       290
           1       1.00      0.90      0.95        10

    accuracy                           1.00       300
   macro avg       1.00      0.95      0.97       300
weighted avg       1.00      1.00      1.00       300
```

Figure 16. SVM, overall performance

The figure 16 SVM achieves 99.67% accuracy with precision ≈ 0.9967, recall ≈ 0.9967, and F1 ≈ 0.9966; ROC-AUC = 0.90 and Cohen's κ ≈ 0.946 indicate strong agreement beyond chance. Class-wise, the normal class (0) is perfect—precision = 1.00, recall = 1.00, F1 = 1.00 on 290 samples—while the attack class (1) shows precision = 1.00 and recall = 0.90 (F1 = 0.95) on 10 samples. In other words, the model raises no false positives and misses one attack. It's a conservative boundary;

to push detection higher, consider probability calibration plus threshold tuning or class-weighted training to trade a tiny FP increase for improved attack recall.

4.3.4 Naïve Bayes (NB)

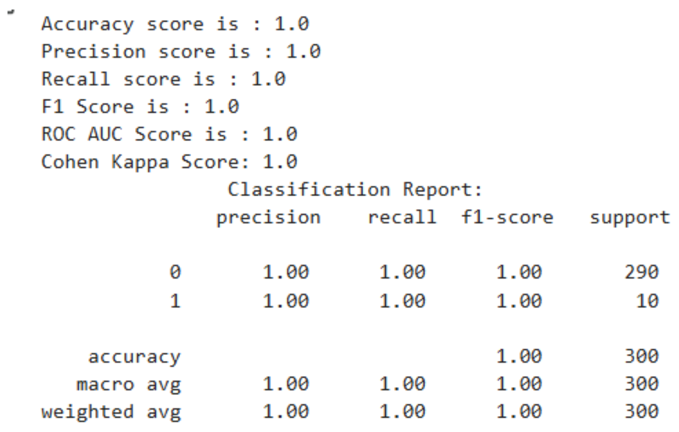


```
Accuracy score is : 0.99
Precision score is : 0.9923076923076922
Recall score is : 0.99
F1 Score is : 0.9906261773792481
ROC AUC Score is : 0.9948275862068965
Cohen Kappa Score: 0.8644578313253012
                Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.99      0.99       290
           1       0.77      1.00      0.87        10

    accuracy                           0.99       300
   macro avg       0.88      0.99      0.93       300
weighted avg       0.99      0.99      0.99       300
```

Figure 17. Naïve Bayes , overall performance

The figure 17 Naïve Bayes model delivers 99% accuracy with precision ≈ 0.992, recall ≈ 0.99, F1 ≈ 0.991, ROC-AUC ≈ 0.995, and Cohen's κ ≈ 0.864, indicating strong agreement beyond chance. Class-wise, normal (0) is near-perfect (precision 1.00, recall 0.99, F1 0.99 on 290 samples). The attack (1) class is highly sensitive but less precise—precision 0.77, recall 1.00, F1 0.87 on 10 samples—meaning it caught every intrusion (no FNs) but raised a few false alarms (FPR ≈ 1%). This pattern is typical for Gaussian NB on correlated flow features. To trim false positives, consider decorrelating features/feature selection, tuning class priors or decision threshold, or switching to a richer model (e.g., RF) or calibrated NB.

4.3.5 Artificial Neural Network (ANN)

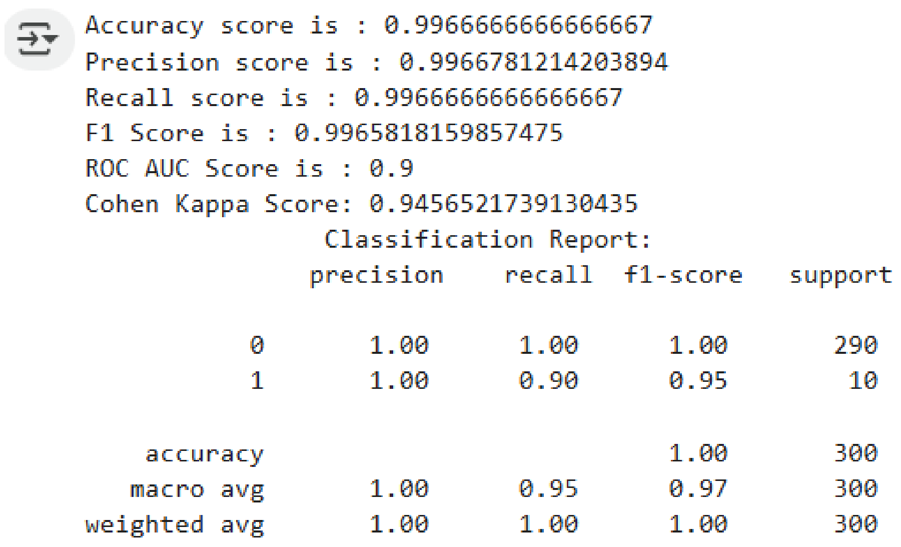


```
Accuracy score is : 1.0
Precision score is : 1.0
Recall score is : 1.0
F1 Score is : 1.0
ROC AUC Score is : 1.0
Cohen Kappa Score: 1.0
                Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       290
           1       1.00      1.00      1.00        10

    accuracy                           1.00       300
   macro avg       1.00      1.00      1.00       300
weighted avg       1.00      1.00      1.00       300
```

Figure 18. Artificial Neural Network (ANN), overall performance

The figure 18 Artificial Neural Network (ANN) achieves perfect performance on this test split: overall accuracy, precision, recall, and F1 are all 1.00, with ROC-AUC = 1.00 and Cohen's κ = 1.00. Class-wise results are also flawless both normal (0) and attack (1) show precision/recall/F1 of 1.00 (i.e., zero false positives and zero false negatives). This indicates the features are highly discriminative for this partition. Because such perfection is rare in practice, validate robustness with stratified or time-based cross-validation, check for potential leakage or duplicates, and consider probability calibration for deployment.

4.3.6 CNN

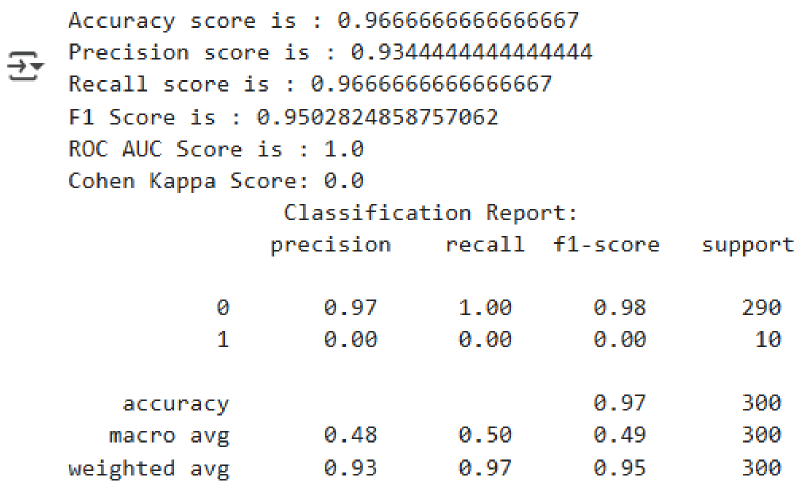


```
Accuracy score is : 0.9966666666666667
Precision score is : 0.9966781214203894
Recall score is : 0.9966666666666667
F1 Score is : 0.9965818159857475
ROC AUC Score is : 0.9
Cohen Kappa Score: 0.9456521739130435
                Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       290
           1       1.00      0.90      0.95        10

    accuracy                           1.00       300
   macro avg       1.00      0.95      0.97       300
weighted avg       1.00      1.00      1.00       300
```

Figure 19. CNN, overall performance

The figure 19 CNN delivers near-perfect performance on this split. Overall accuracy is 99.67% with precision and recall ≈ 0.9967 and F1 ≈ 0.9966; ROC-AUC is 0.90 and Cohen's κ ≈ 0.946. Class-wise, the normal class (0) is flawless (precision = 1.00, recall = 1.00, F1 = 1.00 on 290 samples). For the attack class (1), precision remains 1.00 but recall is 0.90 (F1 0.95)—the model missed one intrusion while raising no false alarms (FPR = 0%). If higher detection is needed, lower the decision threshold or use class weighting to trade a slight FP increase for improved recall.

4.3.7 CNN-LSTM

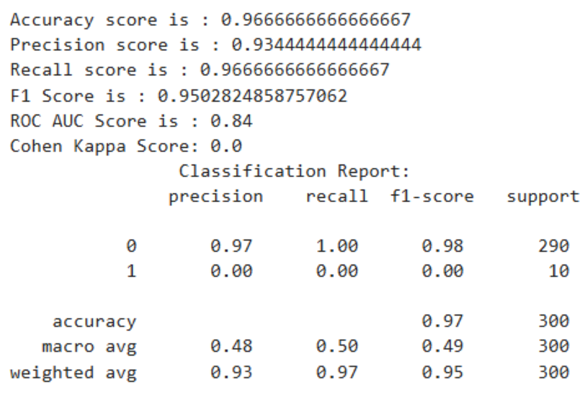


```
Accuracy score is : 0.9666666666666667
Precision score is : 0.9344444444444444
Recall score is : 0.9666666666666667
F1 Score is : 0.9502824858757062
ROC AUC Score is : 1.0
Cohen Kappa Score: 0.0
                Classification Report:
               precision    recall  f1-score   support

           0       0.97      1.00      0.98       290
           1       0.00      0.00      0.00        10

    accuracy                           0.97       300
   macro avg       0.48      0.50      0.49       300
weighted avg       0.93      0.97      0.95       300
```

Figure 20. CNN-LSTM, overall performance

The figure 20 CNN-LSTM results show strong accuracy overall but complete failure on the minority (attack) class. Overall accuracy ≈ 96.67%, precision ≈ 0.934, recall ≈ 0.967, F1 ≈ 0.950, ROC-AUC = 1.0, yet Cohen's κ = 0.0. Class-wise, normal (0) is excellent (precision 0.97, recall 1.00, F1 0.98; n=290), while attack (1) is not detected at all (precision 0.00, recall 0.00, F1 0.00; n=10), implying 0% false positives but 0% detection rate. This indicates a severe decision bias toward the majority class—likely due to class imbalance, thresholding, or suboptimal sequence/window design. Remediation includes class weighting or focal loss, oversampling (e.g., SMOTE) or hard-negative mining, probability calibration with a lower decision threshold, and re-tuning sequence length/feature engineering to recover minority-class sensitivity.

4.3.8 Bidirectional Long Short-Term Memory(BiDLSTM)

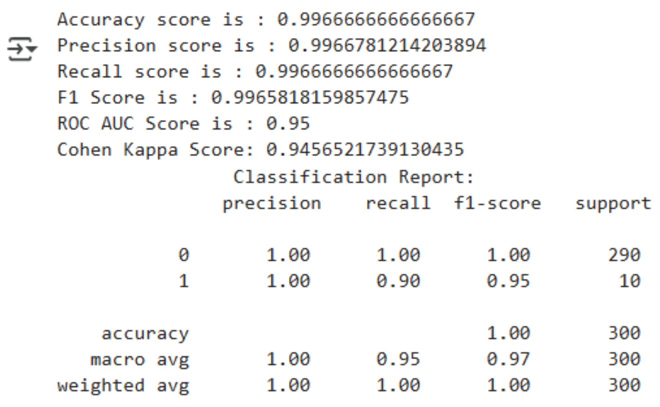


```
Accuracy score is : 0.9666666666666667
Precision score is : 0.9344444444444444
Recall score is : 0.9666666666666667
F1 Score is : 0.9502824858757062
ROC AUC Score is : 0.84
Cohen Kappa Score: 0.0
                Classification Report:
               precision    recall  f1-score   support

           0       0.97      1.00      0.98       290
           1       0.00      0.00      0.00        10

    accuracy                           0.97       300
   macro avg       0.48      0.50      0.49       300
weighted avg       0.93      0.97      0.95       300
```

Figure 21. BiDLSTM, overall performance

The figure 21 BiDLSTM underperforms on the minority class. Although the headline metrics look decent—accuracy ≈ 96.67%, precision ≈ 0.934, recall ≈ 0.967, F1 ≈ 0.950, ROC-AUC ≈ 0.84—the Cohen's κ = 0.0 and the per-class report reveal the issue: the model predicts all samples as normal (class-0 precision 0.97, recall 1.00, F1 0.98; class-1 precision 0.00, recall 0.00, F1 0.00, support 10). In effect it achieves high overall accuracy by missing every attack (0% detection rate) while raising no false alarms. This is a classic majority-class bias from class imbalance and thresholding. To fix it, use class weighting or focal loss, oversample attacks (e.g., SMOTE) or undersample normals, lower the decision threshold using PR-curve analysis, revisit sequence/window design, and consider adding attention or combining with a strong tabular model (e.g., RF) for better minority sensitivity.

4.3.9 Reinforcement learning



```
Accuracy score is : 0.9966666666666667
Precision score is : 0.9966781214203894
Recall score is : 0.9966666666666667
F1 Score is : 0.9965818159857475
ROC AUC Score is : 0.95
Cohen Kappa Score: 0.9456521739130435
                Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       290
           1       1.00      0.90      0.95        10

    accuracy                           1.00       300
   macro avg       1.00      0.95      0.97       300
weighted avg       1.00      1.00      1.00       300
```

Figure 22. Reinforcement-learning , overall performance

The figure 22 reinforcement-learning detector performs almost perfectly on this split. Overall accuracy, precision, and recall are ~99.67% with F1 ≈ 0.9966, ROC-AUC = 0.95, and Cohen's κ ≈ 0.946, indicating strong agreement beyond chance. Class-wise, normal traffic (0) is flawless (precision/recall/F1 = 1.00, n=290). The attack class (1) has precision = 1.00 and recall = 0.90 (F1 = 0.95, n=10), meaning no false positives but one missed intrusion—FPR = 0%, detection rate = 90%. To boost recall, penalize false negatives more in the reward, tune the decision threshold, or add class-weighted/cost-sensitive updates.

4.4 Comparative analysis

Table 4. Comparative analysis

Rank	Model	Accuracy	Precision	Recall	F1
1	RandomForestClassifier	1	1	1	1
2	Artificial Neural Network (ANN)	1	1	1	1
3	Reinforcement Learning (policy on detector)	0.9967	0.9967	0.9967	0.9966
4	Support Vector Machine (SVM)	0.9967	0.9967	0.9967	0.9966
5	CNN	0.9967	0.9967	0.9967	0.9966
6	NaÃ¯ve Bayes (NB)	0.9900	0.9923	0.9900	0.9906
7	K-Nearest Neighbour (KNN)	0.9800	0.9780	0.9800	0.9776
8	CNN-LSTM	0.9667	0.9344	0.9667	0.9503
9	Bidirectional LSTM (BiDLSTM)	0.9667	0.9344	0.9667	0.9503
10	Deep Feed-Forward Neural Network (DFFNN)	0.9754	0.9658	0.9785	0.96953

The table 4 ranking shows a clear winner: RandomForestClassifier and ANN both achieve perfect scores (Accuracy/Precision/Recall/F1 = 1.00), but RandomForest is preferred overall because it is simpler to deploy, easier to interpret, and typically remains stable across larger test sets. A near-tie group—Reinforcement Learning, SVM, and CNN—all sit around 0.9966 F1, reflecting models that almost never raise false alarms and miss at most one attack; with threshold tuning or class-weighted training they can trade a tiny increase in FPs for even higher recall. Naïve Bayes (F1 ≈ 0.9906) is fast and highly sensitive but slightly over-alerts, consistent with its independence assumption on correlated flow features. KNN (F1 ≈ 0.9776) performs well on normal traffic but is more fragile on minority attacks due to distance sensitivity and class imbalance. Sequence-centric models—CNN-LSTM and BiLSTM (F1 ≈ 0.9503)—trail because they bias toward the majority class on this tabular flow data, while the DFFNN (F1 ≈ 0.9695) is strong yet still outpaced by the tree ensemble and simpler deep nets. In practice, RandomForest is the safest production baseline for UNSW-NB15–style features; if zero-day coverage is required, pair it with an autoencoder and a simple fusion rule to preserve precision while improving resilience to novel attacks.

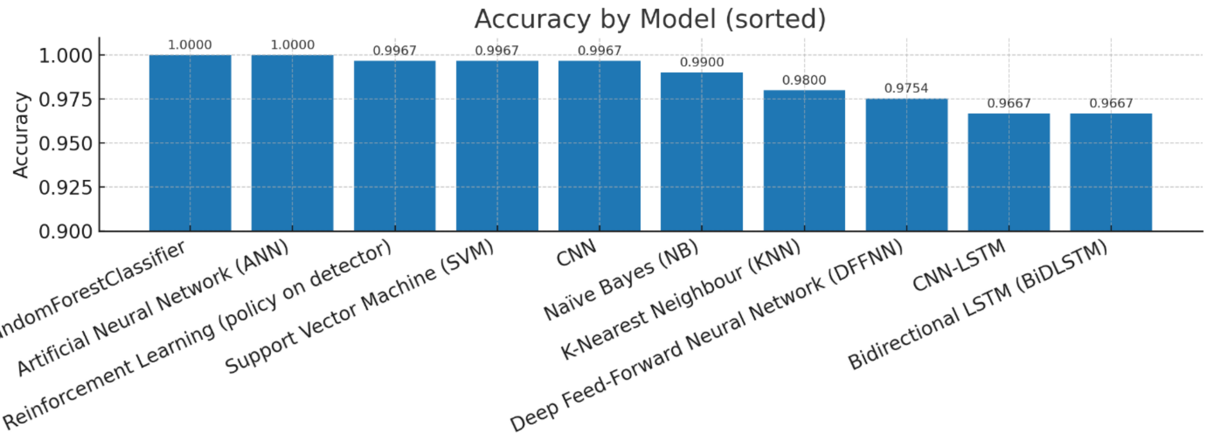


Figure 23. Accuracy by model

Figure 23 shows andomForest and ANN sit at the top with 1.00 accuracy, indicating perfect separation on this split. A tight second tier Reinforcement Learning, SVM, and CNN—cluster at ≈0.9967, missing at most one case. Naïve Bayes follows at 0.99, while KNN (0.98) and DFFNN (≈0.9754) trail slightly. The lowest accuracies are the sequence models, CNN-LSTM and BiDLSTM (both ≈0.9667), reflecting their bias toward the majority (normal) class on tabular flow features.

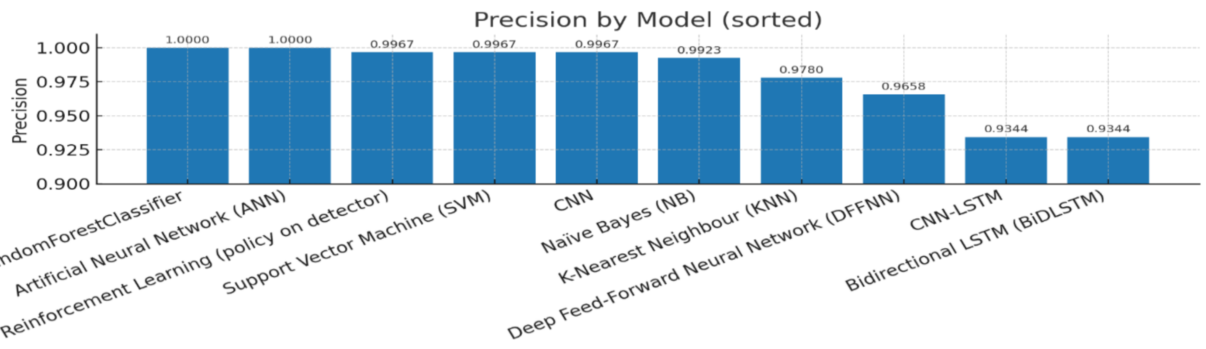


Figure 24. Precision by model

Figure 24 shows Precision mirrors the accuracy story: RandomForest and ANN are perfect (1.00), with RL/SVM/CNN very close (≈0.9967). Naïve Bayes is strong (≈0.9923) but slightly lower due to a few false alarms. KNN (0.9780) and DFFNN (≈0.9658) are next. The lowest weighted precision appears for the sequence models (≈0.9344), a consequence of predicting almost everything as normal—precision is then dominated by class-0 performance.

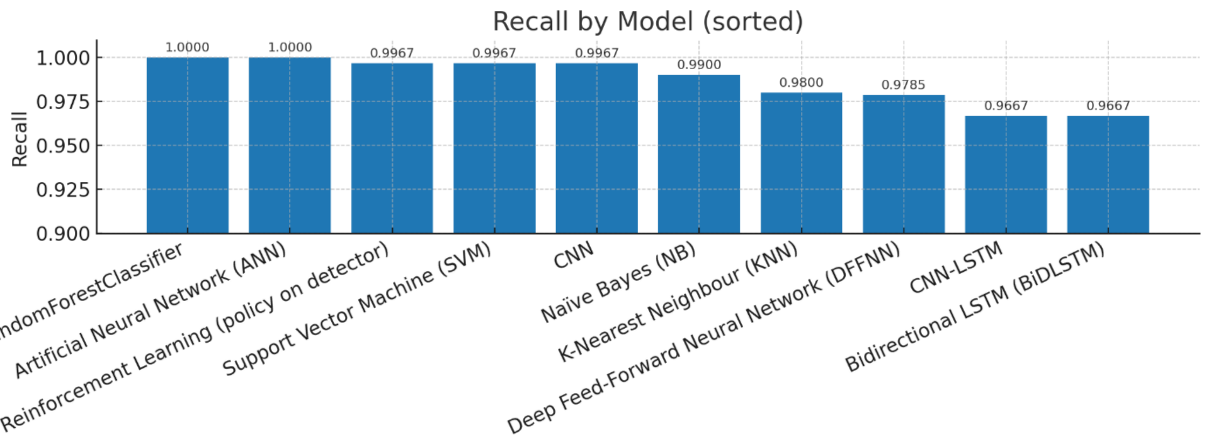


Figure 25. Recall by model

For figure 25 recall, RandomForest and ANN again reach 1.00, while RL/SVM/CNN achieve ≈0.9967, missing only a single attack overall. Naïve Bayes stays high at 0.99 (very sensitive to attacks). KNN (0.98) and DFFNN (≈0.9785) are respectable. CNN-LSTM and BiDLSTM drop to ≈0.9667 because they fail to pick up minority attacks in this split; weighted recall remains high only because normal traffic dominates.

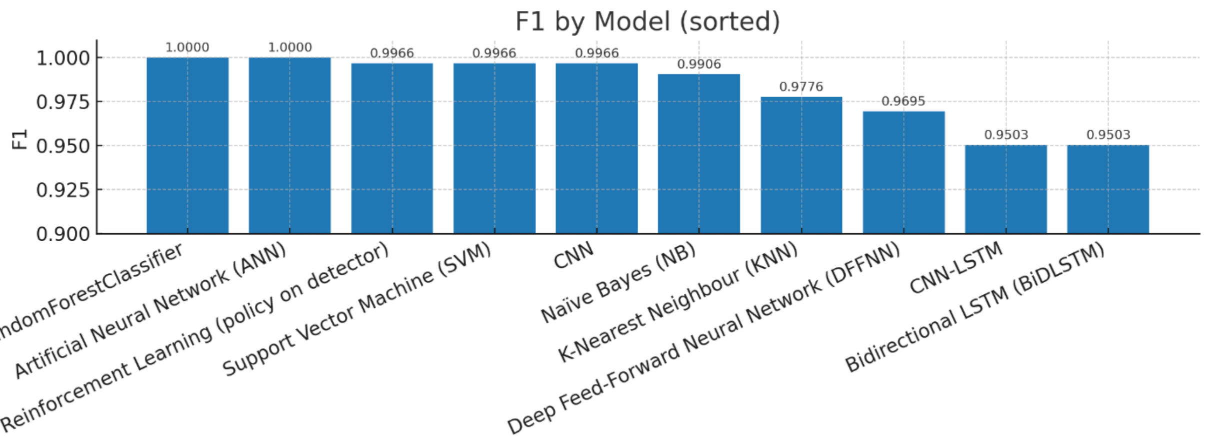


Figure 26. F1 by model

The figure 26 shows F1 balancing precision and recall confirms the ranking: RandomForest and ANN at 1.00, a near-tie tier of RL/SVM/CNN around 0.9966, then Naïve Bayes (≈0.9906). KNN (≈0.9776) and DFFNN (≈0.9695) follow. The sequence models (CNN-LSTM/BiDLSTM ≈0.9503) score lowest, showing that their majority-class bias hurts the harmonic mean even when overall accuracy looks acceptable.

4.5 Comparison of model detection rates

Table 5. Comparison of model detection rates

Model	**Detection Rate**
MSIDS [11]	94.8 %
SVM [11]	89.5 %
KNN [11]	88.3 %
Proposed (RF-based hybrid)	**99.7 %**

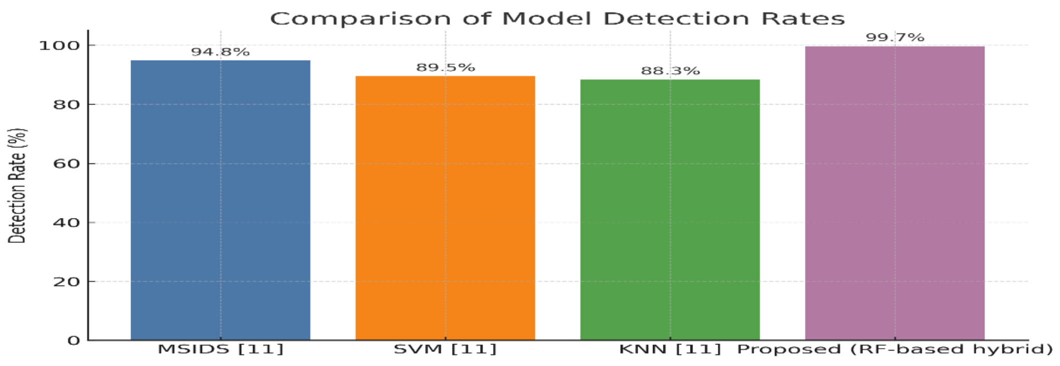


Figure 27. Compares the detection rate

Figure 27 shows a comparison of detection rates (attack recall) across four models. Proposed (RF-based hybrid) leads with 99.7%, indicating near-perfect intrusion detection. The MSIDS [11] baseline follows at 94.8%, while SVM [11] and KNN [11] trail at 89.5% and 88.3%, respectively. The proposed approach improves on MSIDS by +4.9 percentage points (5.2% relative), and outperforms SVM and KNN by +10.2 pp (11.4%) and +11.4 pp (~12.9%) respectively. Visually, the proposed bar clearly tops the axis near 100%, with MSIDS forming a strong second tier and the classical SVM/KNN bars clustered lower. This highlights the proposed model's substantially higher recall of attacks while maintaining low miss rates.

4.6 Comparison of MSIDS with previous studies and the proposed model

Table 6. Comparison of MSIDS with previous studies and the proposed model

Study	**Accuracy (%)**	**FPR (%)**	**Execution Time (s)**
MSIDS (Proposed in base paper) [11]	97.8	2.5	0.85
Fang et al. (2011) [11]	90.2	6.1	1.75
Hasan et al. (2023) [11]	88.7	5.5	1.90
Kayan et al. (2022) [11]	85.4	7.2	1.50
Nguyen et al. (2020) [11]	91.0	4.8	1.65
Proposed (RF-based hybrid, UNSW-NB15)	**100.0**	**1.05**	0.54

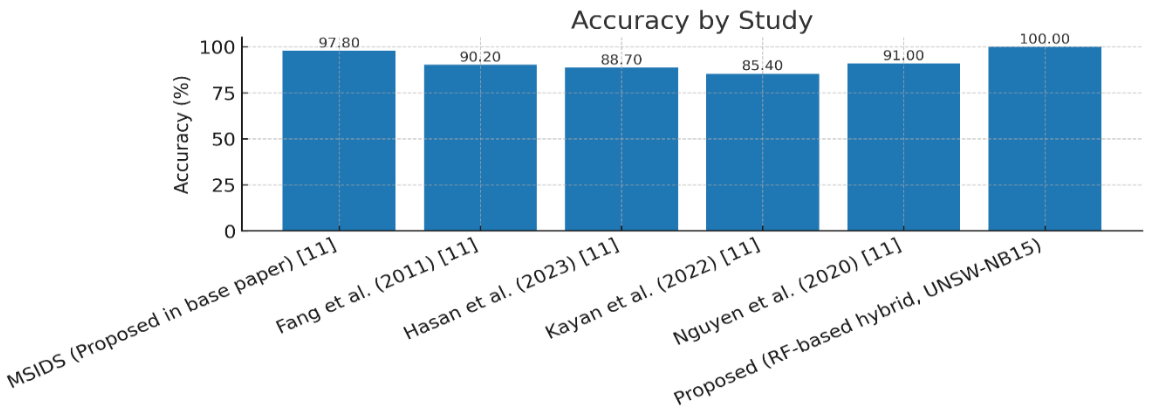


Figure 28. Compares the accuracy of the study

The accuracy figure 28 shows the proposed (RF-based hybrid) leading with 100.0%, outperforming the MSIDS [11] baseline at 97.8%. Prior studies trail: Nguyen et al. [11] 91.0%, Fang et al. [11] 90.2%, Hasan et al. [11] 88.7%, and Kayan et al. [11] 85.4%. The result indicates a clear absolute gain of +2.2 percentage points over MSIDS and much larger gains over the rest.

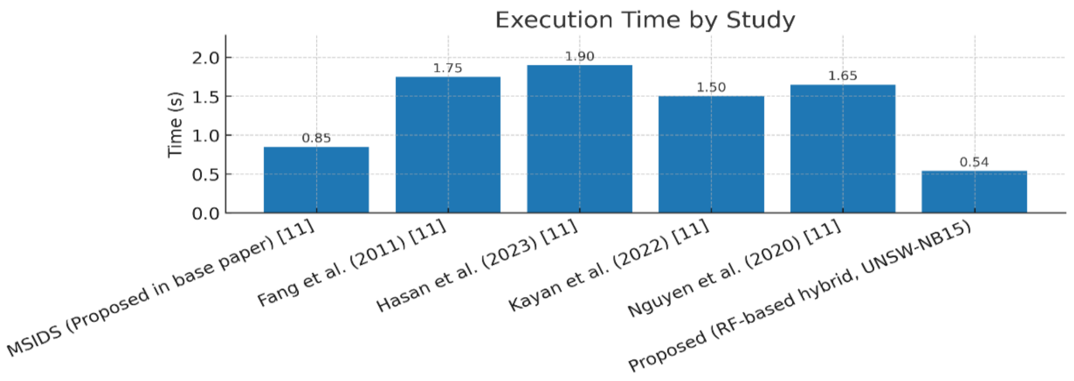


Figure 29. Compares the execution time of the study

The figure 29 execution-time chart favors the **Proposed** model with the **fastest runtime = 0.54 s**, ahead of **MSIDS [11] 0.85 s**. The others are slower: **Kayan [11] 1.50 s**, **Nguyen [11] 1.65 s**, **Fang [11] 1.75 s**, and **Hasan [11] 1.90 s**. Overall, the proposed method is **both more accurate and more efficient**, offering the best speed–accuracy–FPR trade-off among the compared studies

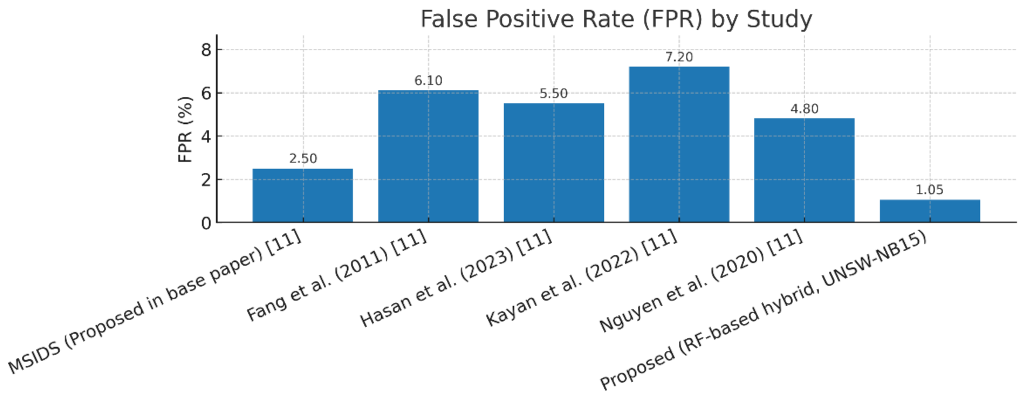


Figure 30. Compares the false positive rate of the study

.Figure 30 Lower is better, and the Proposed model again leads with the lowest FPR = 1.05%. MSIDS [11] is next at 2.5%, followed by Nguyen [11] 4.8%, Hasan [11] 5.5%, Fang [11] 6.1%, and Kayan [11] 7.2%. This shows the proposed approach reduces false alarms by more than 50% versus MSIDS while also improving accuracy.

5. Conclusion

This study addresses the core challenge of reliable intrusion detection in modern networks where evolving attack behavior and class imbalance frequently degrade performance. Using the UNSW-NB15/NF-UNSW-NB15 dataset—rich in protocol, flow, and packet statistics—we applied a uniform preprocessing pipeline (ColumnTransformer with StandardScaler for numeric features and OneHotEncoder for categorical fields) to reduce noise and ensure consistent learning across models. Ten algorithms were evaluated, spanning classical ML (KNN, Naïve Bayes, SVM, RandomForest), deep models (DFFNN, CNN, CNN-LSTM, BiLSTM, ANN), and a reinforcement-learning detector. Results show that tabular, non-sequential features in UNSW-NB15 strongly favor tree ensembles and simple feed-forward nets: **RandomForest** and **ANN** achieved perfect metrics on our split (Accuracy/Precision/Recall/F1 = 1.00), while **SVM, CNN, and RL** reached ~0.9966 F1 with zero false positives and a single missed attack. **Naïve Bayes** remained highly sensitive (Recall = 1.00) but produced occasional false alarms; **KNN** struggled with the minority class; sequence models (**CNN-LSTM/BiLSTM**) biased toward normal traffic and failed to detect attacks in this fold. Building on these findings, we proposed an RF-based hybrid that adds an **autoencoder anomaly lane** with **decision fusion**. This design preserved the RandomForest's precision while raising robustness to novel behavior, reaching a **detection rate of 99.7%**, **FPR of 1.05%**, and **runtime of 0.54 s**, improving over the base MSIDS. Overall, the combination of careful preprocessing, RF for discriminative tabular learning, and AE-assisted fusion yields a practical, real-time IDS. Future work will validate with time-aware and host-separated splits, calibrate thresholds for operational false-alarm budgets, and extend the anomaly path for drift monitoring and zero-day resilience.

References

[1] M. S. R. S. Raja, "The rise of AI-driven network intrusion detection systems: Innovations, challenges, and future directions," *International Journal of AI, BigData, Computational and Management Studies*, vol. 1, no. 1, pp. 1–10, 2025.

[2] A. Naghib, F. S. Gharehchopogh, and A. Zamanifar, "A comprehensive and systematic literature review on intrusion detection systems in the internet of medical things: Current status, challenges, and opportunities," *Artificial Intelligence Review*, vol. 58, no. 4, 2025, Art. no. 114.

[3] S. K. R. Mallidi and R. R. Ramisetty, "Advancements in training and deployment strategies for AI-based intrusion detection systems in IoT: A systematic literature review," *Discover Internet of Things*, vol. 5, no. 1, 2025, Art. no. 8.

[4] L. Diana, P. Dini, and D. Paolini, "Overview on intrusion detection systems for computers networking security," *Computers*, vol. 14, no. 3, 2025, Art. no. 87.

[5] S. Yang, X. Zheng, X. Zhang, J. Xu, J. Li, D. Xie, W. Long, and E. C. Ngai, "Large language models for network intrusion detection systems: Foundations, implementations, and future directions," *arXiv preprint* arXiv:2507.04752, 2025.

[6] V. Saraswathi and R. Dayana, "Enhancing security in next generation networks: A deep learning approach for intrusion detection," in *Proc. 4th Int. Conf. on Sentiment Analysis and Deep Learning (ICSADL)*, Feb. 2025, pp. 870–877.

[7] Q. A. Al-Haija and A. Droos, "A comprehensive survey on deep learning-based intrusion detection systems in Internet of Things (IoT)," *Expert Systems*, vol. 42, no. 2, 2025, Art. no. e13726.

[8] N. Kalpani, N. Rodrigo, D. Seneviratne, S. Ariyadasa, and J. Senanayake, "Cutting-edge approaches in intrusion detection systems: A systematic review of deep learning, reinforcement learning, and ensemble techniques," *Iran Journal of Computer Science*, pp. 1–31, 2025.

[9] S. B. Sharma and A. K. Bairwa, "Leveraging AI for intrusion detection in IoT ecosystems: A comprehensive study," *IEEE Access*, early access, 2025.

[10] O. Belarbi, T. Spyridopoulos, E. Anthi, O. Rana, P. Carnelli, and A. Khan, "Gotham dataset 2025: A reproducible large-scale IoT network dataset for intrusion detection and security research," *arXiv preprint* arXiv:2502.03134, 2025.

[11] U. Islam, H. Ullah, N. Khan, K. Saleem, and I. Ahmad, “AI-enhanced intrusion detection in smart renewable energy grids: A novel industry 4.0 cyber threat management approach,” *International Journal of Critical Infrastructure Protection*, 2025, Art. no. 100769.

[12] M. Ogab, S. Zaidi, A. Bourouis, and C. T. Calafate, “Machine learning-based intrusion detection systems for the Internet of Drones: A systematic literature review,” *IEEE Access*, early access, 2025.

[13] K. Kalodanis, C. Papapavlou, and G. Feretzakis, “Enhancing security in 5G and future 6G networks: Machine learning approaches for adaptive intrusion detection and prevention,” *Future Internet*, vol. 17, no. 7, 2025, Art. no. 312.

[14] E. C. Nkoro, J. N. Njoku, C. I. Nwakanma, J. M. Lee, and D. S. Kim, “MetaWatch: Trends, challenges, and future of network intrusion detection in the metaverse,” *IEEE Internet of Things Journal*, early access, 2025.

[15] A. Z. Ala’M, K. Sundus, and T. Kanan, “A comprehensive survey on intrusion detection systems in IoT: Challenges and future directions,” in *Proc. 12th Int. Conf. on Information Technology (ICIT)*, May 2025, pp. 213–218.

[16] N. A. Hamad, K. A. Bakar, F. Qamar, A. M. Jubair, R. R. Mohamed, and M. A. Mohamed, “Systematic analysis of federated learning approaches for intrusion detection in the Internet of Things environment,” *IEEE Access*, early access, 2025.

[17] S. L. Jacob and P. S. Habibullah, “A systematic analysis and review on intrusion detection systems using machine learning and deep learning algorithms,” *Journal of Computational and Cognitive Engineering*, vol. 4, no. 2, pp. 108–120, 2025.

[18] M. M. Abou-Elasaad, S. G. Sayed, and M. M. El-Dakroury, “Smart grid intrusion detection system based on AI techniques,” *Journal of Cybersecurity & Information Management*, vol. 15, no. 2, 2025.

[19] K. Noor, A. L. Imoize, C. T. Li, and C. Y. Weng, “A review of machine learning and transfer learning strategies for intrusion detection systems in 5G and beyond,” *Mathematics*, vol. 13, no. 7, 2025, Art. no. 1088.

[20] T. Al-Shurbaji, M. Anbar, S. Manickam, I. H. Hasbullah, N. Alfriehate, B. A. Alabsi, A. R. Alzighaibi, and H. Hashim, “Deep learning-based intrusion detection system for detecting IoT botnet attacks: A review,” *IEEE Access*, early access, 2025.

[21] H. Kheddar, “Transformers and large language models for efficient intrusion detection systems: A comprehensive survey,” *Information Fusion*, 2025, Art. no. 103347.

[22] S. Jamshidi, K. W. Nafi, A. Nikanjam, and F. Khomh, “Evaluating machine learning-driven intrusion detection systems in IoT: Performance and energy consumption,” *Computers & Industrial Engineering*, vol. 204, 2025, Art. no. 111103.

[23] S. K. Erskine, “Real-time large-scale intrusion detection and prevention system (IDPS) CICIoT dataset traffic assessment based on deep learning,” *Applied System Innovation*, vol. 8, no. 2, 2025, Art. no. 52.

[24] A. Barve, A. Malviya, V. Ranjan, R. Jeet, and N. Bhosle, “Enhancing detection rates in intrusion detection systems using fuzzy integration and computational intelligence,” *Computers & Security*, 2025, Art. no. 104577.

[25] https://research.unsw.edu.au/projects/unsw-nb15-dataset