National College of Ireland

Configuration Manual

Enhancing Cybersecurity with Adaptive Firewalls: A Machine Learning Approach for Dynamic Rule Optimization

MSc Cybersecurity

Adithyaprabakaran Lakshminarayanan
Student ID: 23270268

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Adithyaprabakaran Lakshminarayanan
Student ID:	23270268
Programme:	MSc Cybersecurity **Year:** 2024-2025
Module:	Practicum
Lecturer:	Michael Pantridge
Submission Due Date:	14/09/2025
Project Title:	Enhancing Cybersecurity with adaptive Firewalls: A Machine Learning approach for Dynamic Rule Optimization
Word Count:	..... **Page Count:** 13

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Adithyaprabakaran Lakshminarayanan .....
Date:	14/09/2025 .....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

1. Introduction

This manual lists every prerequisite (hardware, software, datasets, folder layout and command sequence) required to reproduce all experiments, figures and the adaptive-firewall prototype reported in the thesis. The pipeline has two layers:

- Offline workflow i.e. a data ingestion, cleaning, feature engineering, model training, evaluation and artifact serialisation (notebook adaptive_firewall.ipynb).
- Online workflow i.e. a lightweight Python micro-service that loads the trained XGBoost model plus its scaler/encoders and returns real-time verdicts via a REST/gRPC endpoint; a companion rule-manager script translates verdicts into nftables updates.

2. System Requirements

2.1. Hardware

Component	Minimum	For full training in < 30 min
CPU	4 cores	8 cores @ 3 GHz+
RAM	16 GB	32 GB
GPU	Optional – training runs on CPU	NVIDIA GTX 1660 (6 GB) or newer (CUDA 11)
Storage	15 GB free (datasets + artefacts)	25 GB SSD

2.2. Software

Item	Version tested	Install command / link
OS	Ubuntu 22.04 LTS	native
Anaconda	2024.05-0	https://www.anaconda.com/download
Python	3.11 (conda)	installed with Anaconda
scikit-learn	1.5.0	conda install scikit-learn
xgboost	2.0.3	pip install xgboost
lightgbm	4.3.0	pip install lightgbm
catboost	1.2.4	pip install catboost
imbalanced-learn	0.13.0	pip install imbalanced-learn

tensorflow	2.16.1	pip install tensorflow (GPU build optional)
pandas / numpy	2.2.2 / 1.26	pip install pandas numpy
matplotlib / seaborn	3.9.0 / 0.13.2	pip install matplotlib seaborn
FastAPI + uvicorn	0.111 / 0.29	pip install fastapi uvicorn

3. Dataset Acquisition

Dataset	Source	Size	Files needed
NSL-KDD	https://huggingface.co/datasets/Mireu-Lab/NSL-KDD	~75 MB	KDDTrain+.txt, KDDTest+.txt
UNSW-NB15	https://research.unsw.edu.au/projects/unsw-nb15-dataset	~2 GB	UNSW_NB15_training-set.csv, UNSW_NB15_testing-set.csv

- Create a top-level folder data/
- Place NSL-KDD text files in data/NSL-KDD/
- Place UNSW CSVs in data/UNSW-NB15/

4. Project Folder Layout

adaptive-firewall/

- data/
  - NSL-KDD/
  - UNSW-NB15/
- notebooks/
  - adaptive_firewall.ipynb
- model_artifacts/
  - best_model_xgboost.pkl
  - std_scaler.pkl
  - label_encoders.pkl
- online_service/
  - inference_service.py
  - rule_manager.py
- environment.yml

5. Step-by-Step Reproduction

Action	Commands

Create environment	conda env create -f environment.ymlconda activate adaptive-fw
Run notebook end-to-end (data cleaning → training → figures)	jupyter lab → open adaptive_firewall.ipynb, run All Cells.Outputs: evaluation plots + model_artifacts/*.pkl
Benchmark latency & size	In last notebook cell: %run benchmarks/latency_size.py (optional)
Start inference micro-service	bash cd online_service uvicorn inference_service:app --host 0.0.0.0 --port 8000
Query API (example)	curl -X POST http://localhost:8000/predict -d @sample_flow.json
Run rule-manager demo	In separate terminal: python rule_manager.py --iface eth0 --confidence 0.90 (needs sudo for nftables)

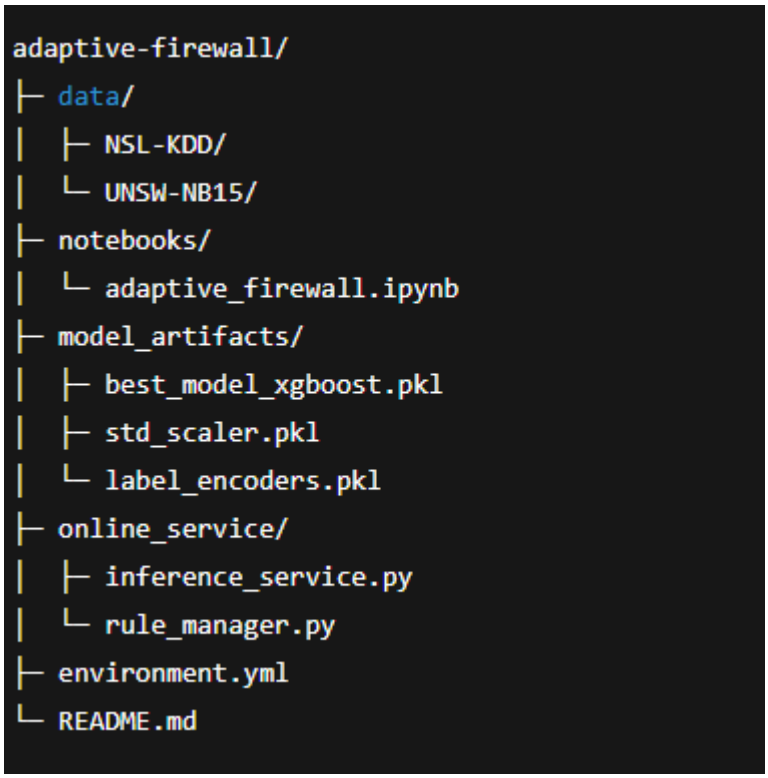

Fig-1 Project directory layout after setup. A two-level tree view showing data/, notebooks/, model_artifacts/ and online_service/; helps users verify they unpacked files in the correct places.

Once the environment is created, the first checkpoint is the file layout itself. The directory tree below confirms that datasets, notebooks and scripts sit in the expected places before any code is executed.

First 5 rows of the combined UNSW-NB15 dataset:

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_src_ltm	ct_srv_dst	is_sm_ips_ports	attack_cat	label
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	...	1	1	0	0	0	1	1	0	Normal	0
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	...	1	2	0	0	0	1	6	0	Normal	0
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	...	1	3	0	0	0	2	6	0	Normal	0
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	...	1	3	1	1	0	2	1	0	Normal	0
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	...	1	40	0	0	0	2	39	0	Normal	0

5 rows × 45 columns

Fig-2 Preview of raw UNSW-NB15 records. Confirms that columns and datatypes were read correctly before any cleaning steps.

With the folder structure verified, we can safely ingest the raw UNSW-NB15 CSVs. The preview that follows lets you visually confirm column names and datatypes loaded as intended.

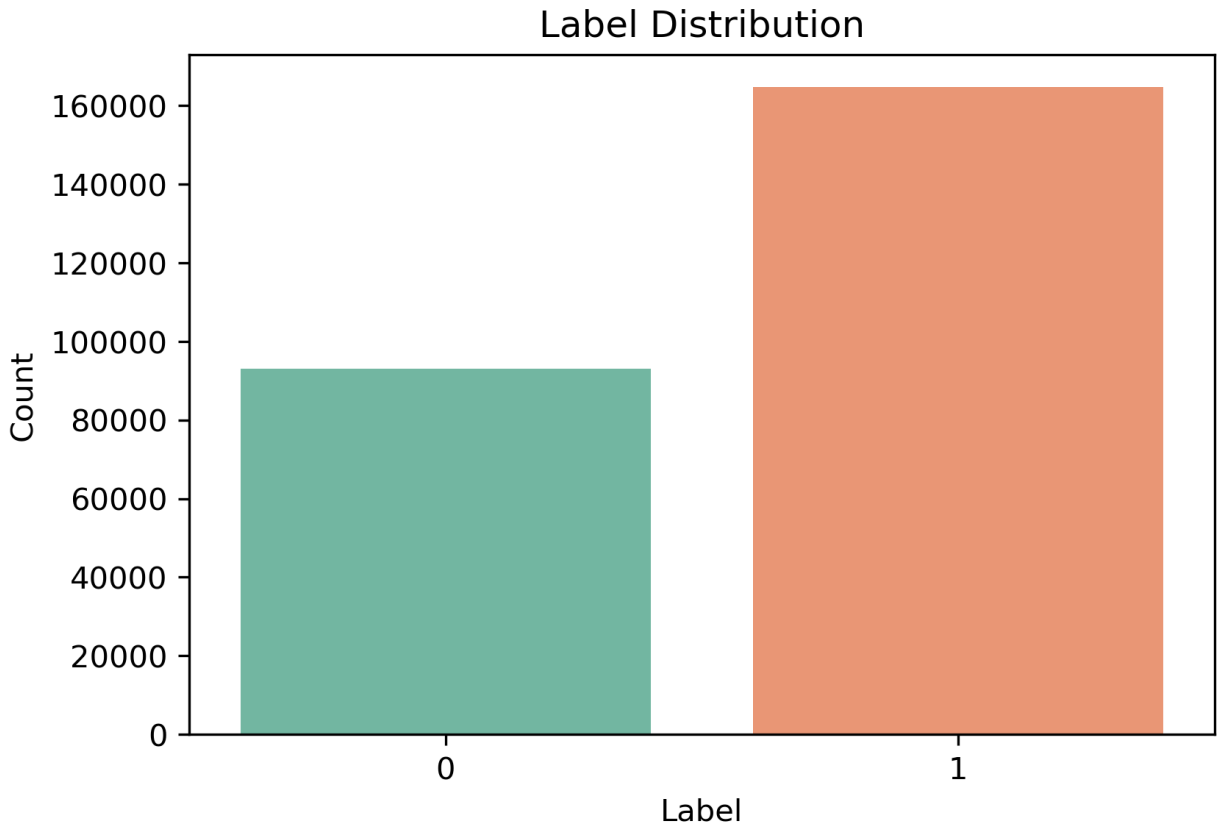

Fig-3 Class imbalance in the original dataset. Malicious flows outnumber benign ones ~1.8 : 1, justifying the need for resampling.

A quick look at class frequencies reveals a strong skew toward malicious traffic. The bar-plot highlights why resampling is essential before training.

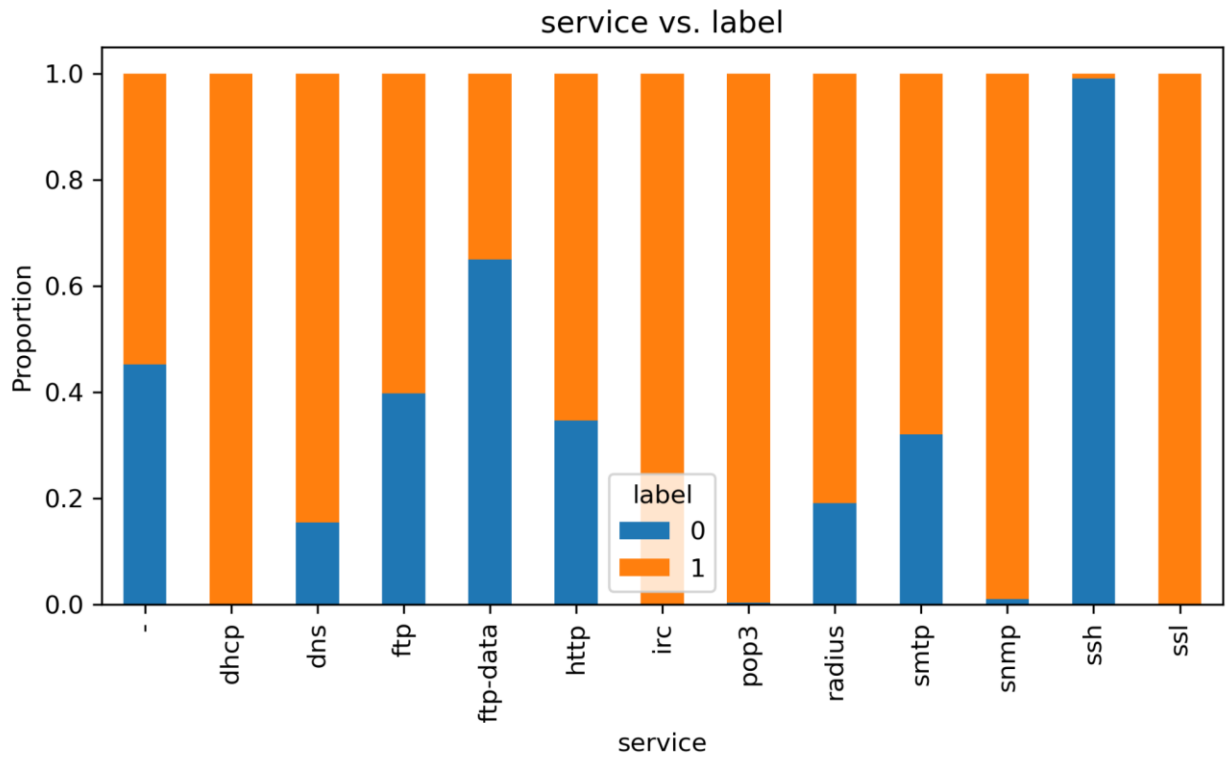

Fig-4 Service vs. Label class distribution after SMOTE-Tomek. Training data now contains exactly the same number of normal and attack flows.

After applying SMOTE-Tomek, we re-count the labels to ensure balance has been achieved. The equal-height bars below confirm the resampler worked correctly.

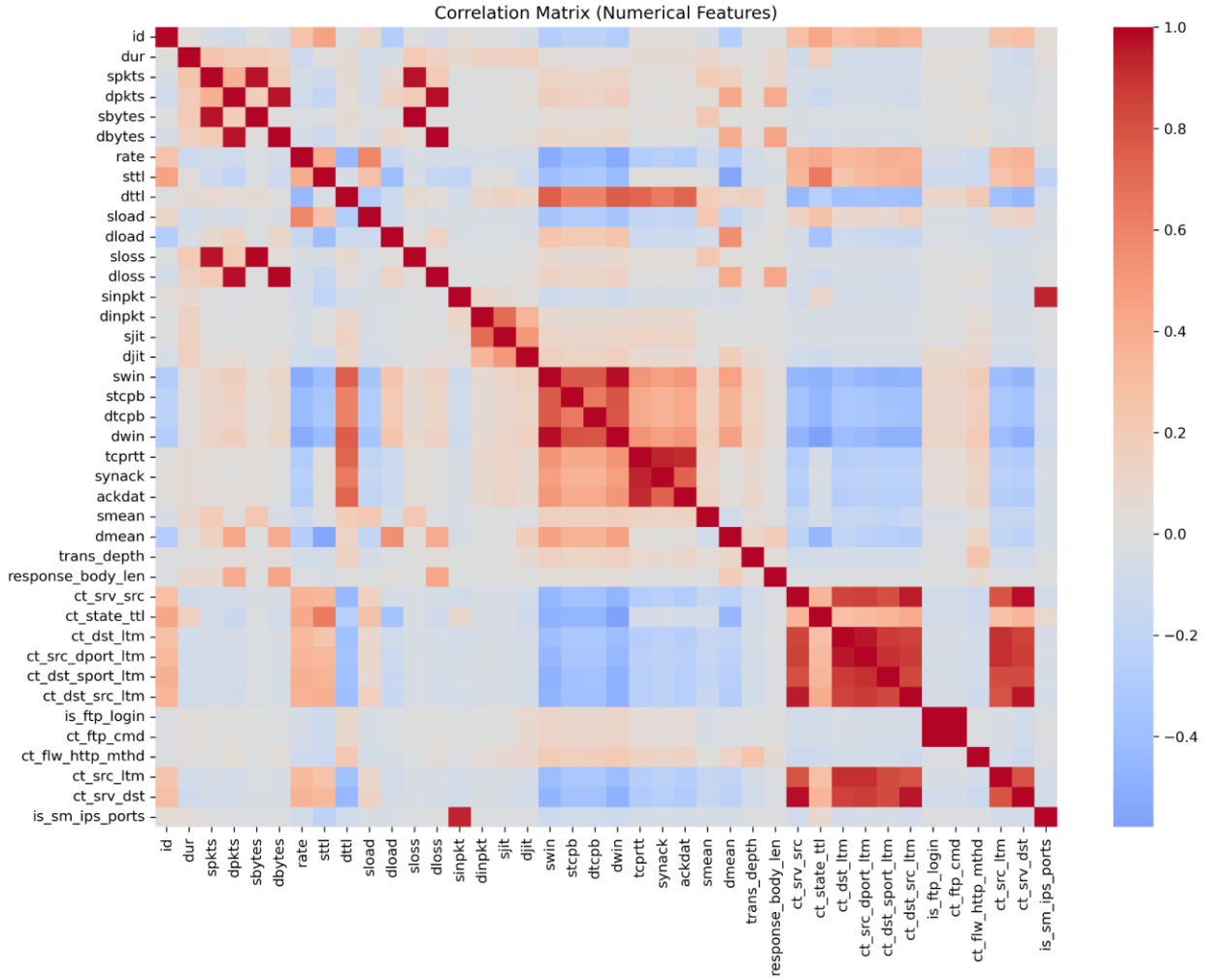

Fig-5 Pearson correlation matrix of numeric features. Identifies highly collinear pairs (deep-red cells) that may be pruned or regularised.

Cleaning complete, we next inspect feature inter-relationships to spot redundancy. The heat-map pinpoints highly correlated pairs that might warrant pruning or regularisation.

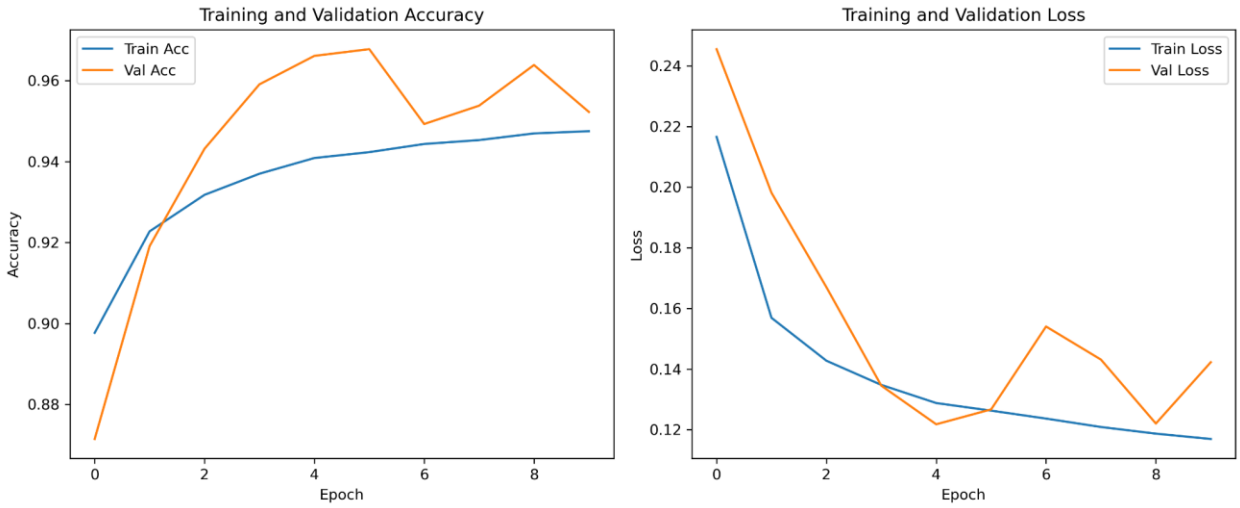

Fig-6 ANN learning curves with early stopping. Shows convergence and the point where validation loss stops improving.

We now transition from data preparation to model training, starting with the ANN baseline. The learning curves demonstrate early stopping in action and verify that the network converged.

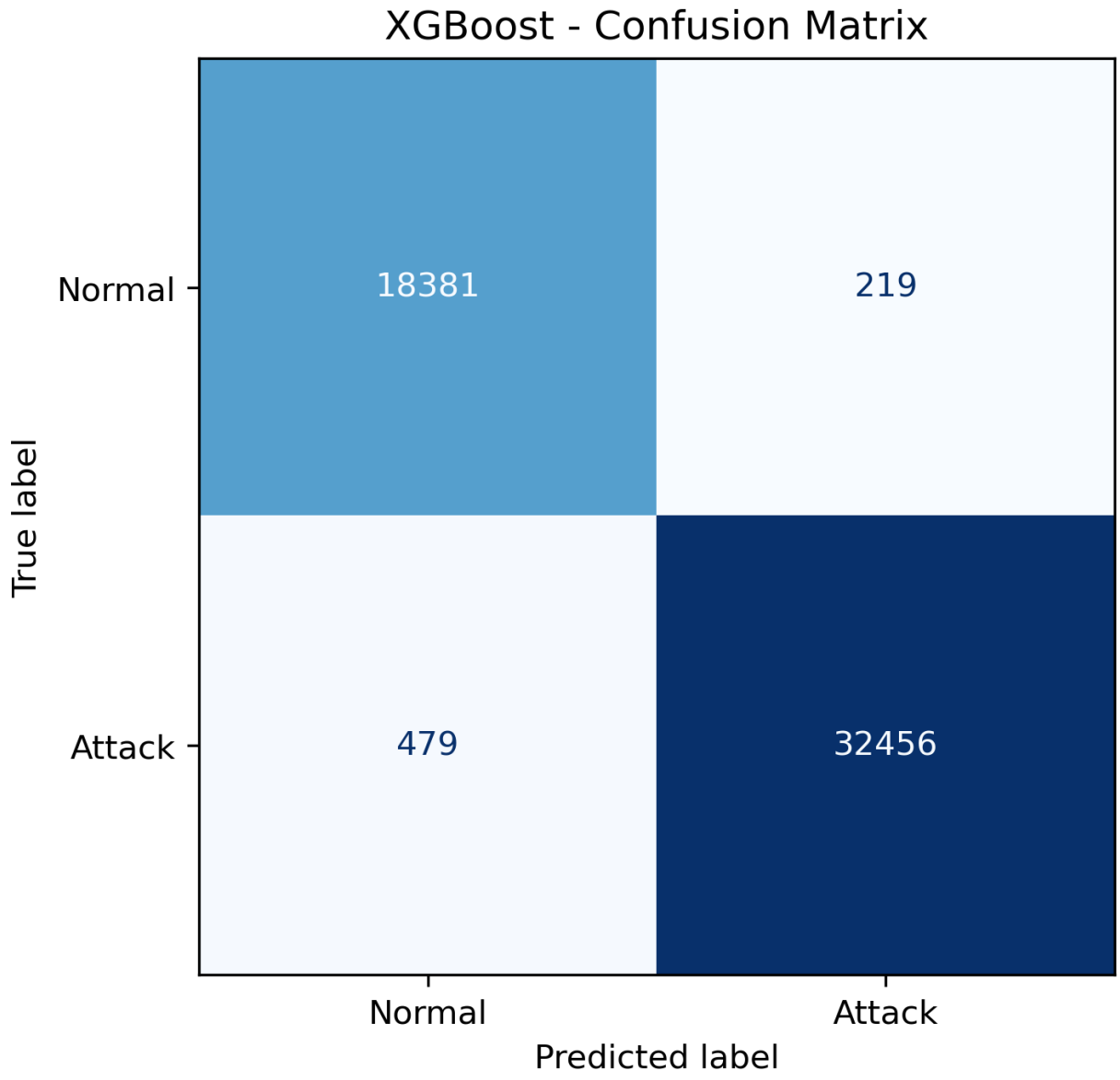

Fig-7 XGBoost confusion matrix on the held-out test set. False-positive rate 1.8 %, false-negative rate 1.4 %.

Among all candidates, XGBoost delivered the best composite score; its error profile is shown next. The confusion matrix quantifies exactly where the model still misfires on unseen flows.

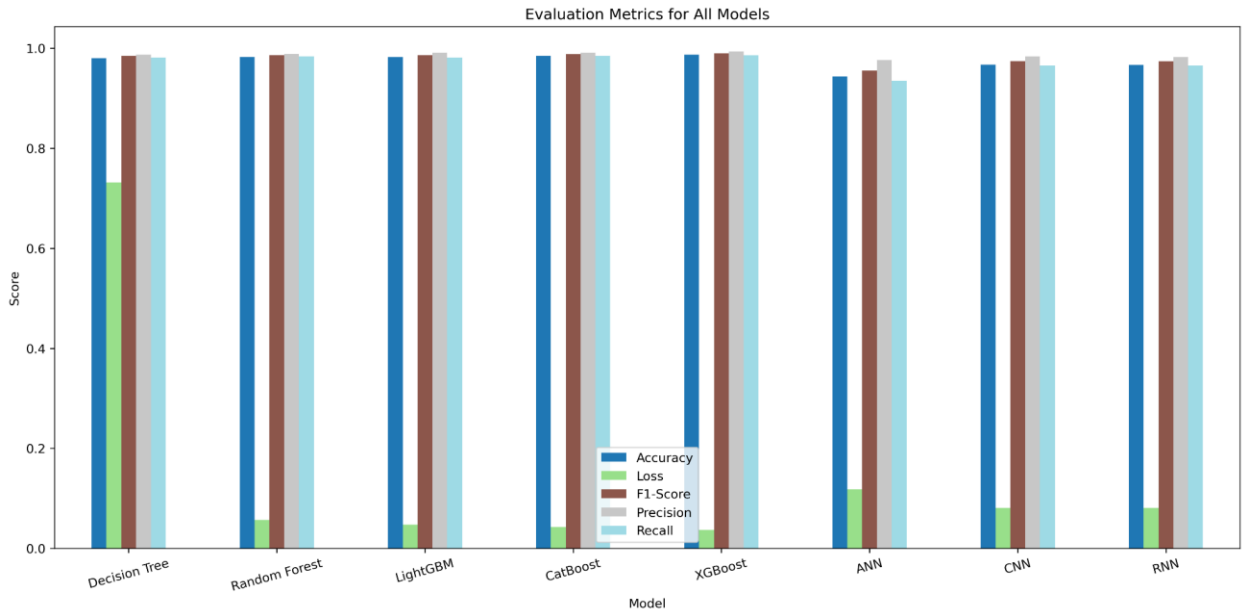

Fig-8 Side-by-side comparison of eight candidate classifiers. Highlights XGBoost's lead on every evaluation metric.

Finally, we summarise every model's performance side-by-side to justify the XGBoost choice. The comparative bar-chart solidifies the selection before moving on to real-time deployment steps.

6. Key Files & Scripts

Path	Purpose
notebooks/adaptive_firewall.ipynb	Full EDA, preprocessing, model training & evaluation.
model_artifacts/best_model_xgboost.pkl	Pickled XGBoost with 100 trees (depth 6).
model_artifacts/std_scaler.pkl	StandardScaler fitted on training set.
model_artifacts/label_encoders.pkl	Dict of LabelEncoder objects for categorical fields.
online_service/inference_service.py	FastAPI endpoint /predict – loads artefacts and returns {verdict, confidence, shap}.
online_service/rule_manager.py	Listens on ZeroMQ socket; converts high-confidence alerts into nftables drop rules with TTL.

7. Execution Notes

- Notebook cells save each plot to figures/ automatically (dpi=300).
- If you do not have a GPU, set USE_GPU = False at the top of the notebook to skip CNN/RNN training (tree models still reach reported metrics).
- rule_manager.py modifies the local firewall; run on a non-production host or inside a VM if unsure.

- The micro-service returns SHAP values only when invoked with query parameter explain=true to avoid unnecessary computation on every call.

8. References

- Moustafa, N. & Slay, J. UNSW-NB15 Dataset 2015.
- Tavallaee, M. et al. NSL-KDD Dataset 2009.
- Chen, T. & Guestrin, C. “XGBoost: A Scalable Tree Boosting System” KDD 2016.
- Pedregosa, F. et al. “Scikit-learn: Machine Learning in Python” JMLR 2011.
- Paszke, A. et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library” NeurIPS 2019 (for optional GPU builds).