Intrusion and Vulnerability Detection in Software-Defined Networks (SDNs) using Gradient Boosting Algorithms

Ndabuye S. Gideon
BSc. Software Engineering, The University of Dodoma, Department of Computer Science and Engineering

ABSTRACT

Gradient boosting models have proven efficient in a wide range of machine learning task in the recent years. Data scientists and ML engineers has been motivated by their gradient boosting models success. Most commonly used of these are LGBM (Light Gradient Boosting Machine), and Catboost.

In this study, these techniques are applied in classifying network traffic flows so as to detect intrusion and vulnerability in SDNs (Software Defined Networks). First, baseline models are built for establishing a benchmark using the three gradient boosting algorithms, LightGBM, HistGradientBoosting and Catboost. These with their default parameters.

Baseline models established a benchmark of above 90% accuracy. Then CatBoostClassifier was tuned and improved due to the support of GPU acceleration, it was then trained using a stratified split using stratified k-fold from sklearn library, where it showed very good performance. It showed an accuracy of 99.8922% on the test data (unseen data).

I. INTRODUCTION

In recent years, traditional networks have changed toward Software Defined Networks (SDNs), where a centralized controller manages and monitors all network traffic. Securing the controller is crucial to maintaining a secure network, and this single point of failure represents a potential vulnerability of SDN. This led network administrators to focus on this issue. Machine learning and AI has proven powerful in solving most of our data problems in almost every domain. SDNs intrusion and vulnerability cannot be an exception.

Gradient boosting models are a class of powerful machine learning algorithms used for classification tasks. They create a strong predictive model by combining the predictions of multiple simpler models, correcting errors iteratively. These models are adept at capturing intricate relationships in data, providing accurate results even for complex patterns. They offer insights into feature importance, handle imbalanced data well, and effectively prevent overfitting. Their ability to model non-linear decision boundaries makes them ideal for various classification challenges, offering a robust and accurate solution.

This paper presents a machine learning approach to classify traffic flows of an SDN which will serve as a vulnerability detection model.

II. RELATED WORK

Several research studies have addressed the crucial issue of intrusion detection and vulnerability assessment within Software Defined Networks (SDNs) using machine learning techniques. This section provides an overview of key contributions in this field.

[1] Zhang et al. conducted an extensive review of intrusion detection systems tailored specifically for SDNs. Their work comprehensively outlines existing approaches and underscores the pivotal role of Intrusion Detection Systems (IDS) in enhancing the security of SDN environments. [2] Al-Hasnawi and Al-Jaroodi conducted a comprehensive survey of machine learning techniques applied to intrusion detection systems within SDNs. Their study presents a comprehensive overview of state-of-the-art methods and their effectiveness in identifying network threats. [3] Hossain et al. explored the integration of deep learning techniques for intrusion detection in SDNs. Their research highlights the potential of deep neural networks in strengthening the security of SDN environments. [4] Xie et al. introduced an innovative intrusion detection framework based on the LightGBM algorithm. Their work demonstrates the efficiency of gradient boosting models in detecting anomalies in SDNs.

III. METHOD

Data Description

The data was provided by ULAK Comm. It combines a small portion of real users obtained from their SD-WAN environment with certain known datasets. The dataset encompasses four primary types of samples: *Normal flow data*, *DDoS flow data*, *Malware flow data*, and *web-based flow data*

The data was provided in train and test with shapes (1783356, 79) and (512077, 79) respectively. Each of which had 15 classes in the target column ‘Label’.

The labels were, *DoS Hulk, BENIGN, DDoS, PortScan, DoS GoldenEye, FTP-Patator, DoS slowloris, DoS Slowhttptest, SSH-Patator, Web Attack – XSS, Web Attack - Brute Force, Web Attack – Sql, Injection, Bot, Infiltration, Heartbleed.*

The figures below show their relative proportion in both the train and test sets.

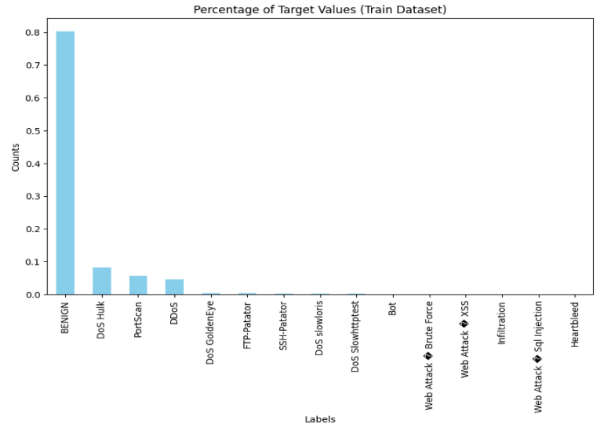

Figure 1: Train data percentage of target values showing imbalanced nature of the dataset

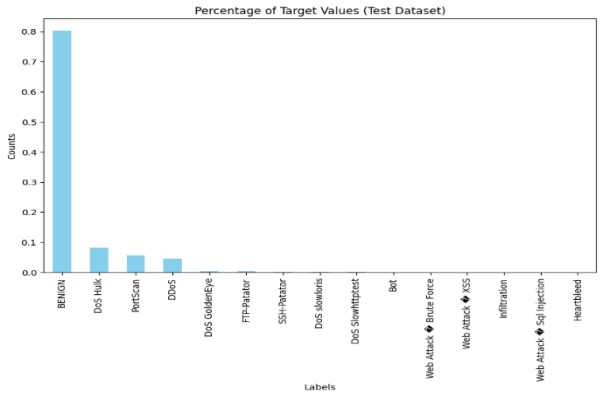

Figure 2: Test data percentage of target values showing imbalanced nature of the dataset

Although the target values are imbalanced, we see equal distribution of classes between the train and test datasets which is vital for building and evaluating a machine learning model.

Baseline Models

For establishing a performance benchmark, I utilized three well-established machine learning models, LightGBM (LGBM), HistGradientBoosting, and CatBoost. These models were chosen for their proven track records in classification tasks and their ability to provide insights into the fundamental predictive capabilities of established algorithms.

LGBM is known for its memory-efficient tree construction and native support for categorical features. HistGradientBoosting from scikit-learn is particularly well-suited for large datasets with a mix of numerical and categorical features in classification tasks. It uses histogram-based techniques for faster training, making it efficient for high-dimensional and noisy datasets. The model strikes a balance between accuracy and training speed, which makes it a strong choice for classification problems where efficiency is important. CatBoost specializes in effective categorical feature handling, showcasing its strength in optimizing splits while capturing intricate categorical relationships.

Data Preprocessing

The provided dataset was pre-cleaned, with only a single column having missing values, of which were filled with the value -99999 to signify their deviation. This was done to match the already-in-the-set negative values.

On exploratory data analysis (EDA), some columns were found to have only a single value, therefore having no use for the learning algorithms. These were 8 columns and they were removed prior to model training.

Finally, all columns with non-binary values were normalized using the StandardScaler.

Training Procedure

All the train data (which consisted of 1,783,356 rows) was used in building the model, and the test data (with 512,077 rows) was used to evaluate it.

Due to the dimensionality and size of the data, CatboostClassifier model was trained on T4 GPU on Google Colab. Thereby taking only 1 min for a single iteration.

The rest of the baseline models were trained on CPU since they have no support for GPU acceleration. And it took LightGBM more than 14 min and HistGB over 9 min for a single iteration.

Since it's a classification problem with imbalanced classes, stratified K-fold technique from sklearn was employed during training. CatBoost stands out for efficiently handling categorical features through techniques like ordered boosting and categorical embeddings. Its architecture leverages gradient boosting principles, integrating regularization, adaptive learning rates, and GPU acceleration for enhanced performance. By choosing CatBoost, a lot was benefited from its robust architecture and the use of GPU.

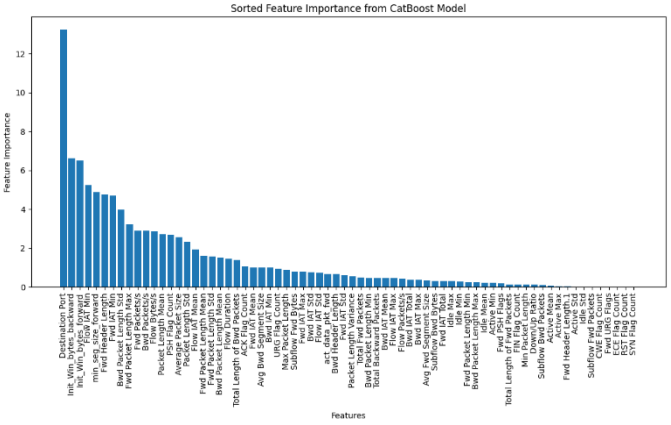

Figure 3: Feature importance from trained Catboost model

Evaluation

All models were trained on the 'train' dataset and evaluated on the 'test' dataset. Therefore, the given test set was not seen by the models during training. This was done to ensure that the evaluation is realistic and valid

After the models were trained, predictions were made on the test set (with 512,077 data points) and a confusion matrix was generated from the predictions to visualize the models' performances on unseen data.

Models were trained with an evaluation function as 'accuracy'.

The below table shows their relative performances in the three models with their default hyperparameters.

Model	Accuracy (%)	Precision (w/m)	Recall(w/m)
LGBM	90.69%	0.92/.28	.91/.24
HistGBoosting	99.21%	1.0/.67	.99/.72
CatBoost	99.70%	1.0/.93	1.0/.82

Figure 4: Baseline performances for three models with default hyperparameters (with w-weighted, m-macro averages)

Model Tuning

Due to the support of GPU acceleration, CatBoostClassifier was taken for hyperparameter tuning and further development.
CatBoostClassifier with GPU acceleration takes around 1 minute for an iteration. Therefore, suitable for further improvement.
CatBoostClassifier hyperparameters were found using Grid Search. The model was trained on 1500 iterations at 0.1 learning rate on 5 stratified splits technique to preserve classes in every fold, thus ensuring model robustness.

IV. Results and Discussion

CatBoostClassifier showed a very good performance after stratified k-splits and parameter tuning. With an accuracy of **99.8966%**
Below is a classification report generated after making predictions on the unseen test dataset. We can see precision, recall and f1-score of every label from the dataset.
Precision measures the accuracy of positive predictions made by a model. It's the ratio of true positive predictions to all positive predictions. Recall assesses the model's ability to identify all positive instances. It's the ratio of true positive predictions to all actual positives. F1-Score combines precision and recall into a single metric. It's the harmonic mean of precision and recall, striking a balance between them.
Some labels with a few numbers of samples (support) relative to the data size have shown slightly smaller recall and precision values.

Class	Precision	Recall	F1-Score	Support
BENIGN	1.0	1.0	1.0	411203
Bot	0.87	0.81	0.84	355
DDoS	1.0	1.0	1.0	23160
DoS GoldenEye	1.0	1.0	1.0	1861
DoS Hulk	1.0	1.0	1.0	41801
DoS Slowhttptest	0.99	0.99	0.99	994
DoS slowloris	0.99	1.0	1.0	1048
FTP-Patator	1.0	1.0	1.0	1436
Heartbleed	1.0	1.0	1.0	2
Infiltration	1.0	0.33	0.5	6
PortScan	0.99	1.0	1.0	28751
SSH-Patator	1.0	1.0	1.0	1067
Web Attack Brute Force	0.74	0.86	0.8	272
Web Attack Sql Injection	0.67	0.5	0.57	4
Web Attack XSS	0.53	0.3	0.38	117

A confusion matrix shown below was also generated from the predictions made by the model on the test set.

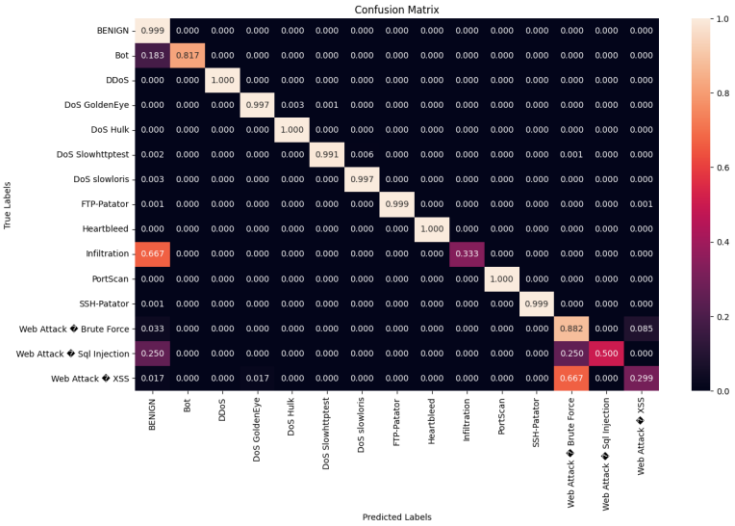

Figure 5: Confusion matrix of the predictions on the test set

V. Conclusion

In summary, this study demonstrates the effectiveness of gradient boosting models, particularly CatBoost, in classifying network traffic flows for intrusion and vulnerability detection in Software Defined Networks (SDNs).
The baseline models, including LightGBM, HistGradientBoosting, and CatBoost, achieved an accuracy benchmark exceeding 90%. CatBoost, with its GPU acceleration and further tuning, outperformed the others.
Through data preprocessing and stratified k-fold training, CatBoostClassifier achieved an impressive accuracy of **99.8966%** on unseen test data. The classification report showcased its robust performance across various traffic types.
However, this work acknowledges limitations due to limited data for some classes. Gathering more data for underrepresented classes could further enhance the model's efficiency.

VI. References

[1] J. Zhang, Y. Bi, and J. Li, "Intrusion Detection System for Software Defined Networking: A Comprehensive Review," in 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2018, pp. 1-6.
[2] A. Al-Hasnawi and J. Al-Jaroodi, "A Survey of Machine Learning Techniques for Intrusion Detection Systems in Software Defined Networking," in 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 4695-4702.
[3] M. S. Hossain, S. M. R. Hasan, and A. Alamri, "An Intrusion Detection System for Software-Defined Networks Using Deep Learning," in 2018 IEEE International Conference on Smart Cloud (SmartCloud), 2018, pp. 214-221.
[4] M. Xie, X. Huang, and J. Hong, "A Novel Intrusion Detection Framework for Software Defined Networks Based on LightGBM Algorithm," in 2020 IEEE Conference on Computer Communications (INFOCOM), 2020, pp. 2222-2229.