University of Massachusetts Dartmouth

Department of Computer and Information Science

Title : Comparative Analysis of Machine Learning and Deep Learning Models for Network Intrusion Detection

A Project in

Machine Learning

by

Robin Singh

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

August 2024

We approve the thesis of Author

Date of Signature

______________________________ ______________

{Full name of Professor}
{Academic Title}, Department of Choose an item.
Thesis Advisor

______________________________ ______________

{Full name of Professor}
{Academic Title}, Department of Choose an item.
Thesis Committee

______________________________ ______________

{Full name of Professor}
{Academic Title}, Department of Choose an item.
Thesis Committee

______________________________ ______________

{Full name of Graduate Program Director}
Graduate Program Director, Choose an item.

______________________________ ______________

{Full name of Professor} Choose an item.
Chairperson, Department of Choose an item.

______________________________ ______________

{Full name of Dean}
Dean, Choose an item.

______________________________ ______________

Tesfay Meressi
Associate Provost for Graduate Studies

Abstract

Title : Comparative Analysis of Machine Learning and Deep Learning Models for Network Intrusion Detection
by Robin Singh

Network intrusion detection systems (NIDS) are essential for safeguarding networks against the constantly changing cyber threat environment. The goal of this project is to compare the effectiveness of different machine learning and deep learning models in identifying network intrusions on two benchmark cybersecurity datasets, CICIDS 2017 and UNSW-NB15. Seven supervised learning models were put into practice and assessed: Logistic Regression, Support Vector Machines, Decision Trees, Random Forests, Naive Bayes, Convolutional Neural Networks, and Long Short-Term Memory networks. The project is centered on multi-class classification for categorizing network traffic into distinct attack classes instead of just differentiating between normal and malicious. Findings indicate that Decision Trees and Random Forests consistently performed well, whereas other models faced challenges when dealing with minority attack classes. Sampling methods were used to enhance the sensitivity of the minority class, even though this led to minor reductions in overall metrics. The results highlight the need for customized methodologies to accurately identify rare yet crucial forms of attacks in imbalanced data sets. This comparative analysis provides insights to guide future enhancements in NIDS model development and training strategies.

Acknowledgments

I would like to express my deepest appreciation to **Dr. Gokhan Kul**, whose invaluable guidance and support have been instrumental in the completion of this project. His expertise and insights have greatly enriched my work, and his encouragement has been a driving force in my academic journey.

I would also like to extend my heartfelt gratitude to my family and friends. Their unwavering belief in my abilities and their constant encouragement have been a source of strength and motivation. Their support, both tangible and intangible, has been a key factor in my success.

Thank you all for your invaluable contribution to this project.

Table of Contents

Chapter 1: Introduction 1

Chapter 2: Dataset and Preprocessing 2

2.1 CICIDS 2017 2

2.2 UNSW NB-15 3

2.3 Dataset Imbalance 5

2.3.1 Sampling Techniques for Balance 5

2.3.2 Our Approach 5

2.4 Dataset Preprocessing 7

Chapter 3: Model Development, Training 9

3.1 Model Development 9

3.1.1 Implemented Models 9

3.2 Training 14

3.2.1 Logistic Regression 14

3.2.2 Support Vector Machine 14

3.2.3 Decision Tree 15

3.2.4 Random Forest 15

3.2.5 Naive Bayes 16

3.2.6 Convolutional Neural Network (CNN) 17

3.2.7 Long Short Term Memory (LSTM) 19

Chapter 4: Evaluation 21

4.1 Evaluated Metrics 21

4.2 Overview of Model Performance (CICIDS Dataset) 23

4.2.1 Challenges with Minority Classes 23

4.2.2 Strategy Adjustments 23

4.2.3 Model Evaluation and Insights 24

4.3 Overview of Model Performance (UNSW Dataset) 32

4.3.1 Challenges with Minority Classes 32

4.3.2 Strategy Adjustments 33

4.3.3 Model Evaluation and Insights 33

Chapter 5: Results 41

Chapter 6: Conclusion and Future Scope 62

Chapter 1: Introduction

This Project aims to do a comparative study about two major datasets for cybersecurity, such as the CICIDS 2017 and UNSW dataset using machine learning algorithms and deep learning algorithms.

The ability to accurately detect unauthorized activities in network security is an essential aspect of strong cybersecurity, given the ever-changing and unpredictable nature of cyber threats. Researchers are focusing on improving Network Intrusion Detection Systems (NIDS) in response to the increasing number of advanced cyberattacks. The objective of this Project is to compare two significant datasets for cybersecurity, by employing major machine learning algorithms and deep learning algorithms.

The primary objective of this project is to not only create a high performing (NIDS) but also to analyze the effectiveness and performance of machine learning models when trained and tested on various datasets. The CICIDS 2017 and UNSW-NB15 datasets are chosen because they are important and contain various data, which includes various types of malicious attacks and normal attacks. By carefully examining the performance of numerous machine learning and deep learning models on these datasets, the study attempts to disclose insights into the strengths and boundaries built-in into each dataset.

A notable trait of this project is that it focuses on multi-class classification, which involves sorting network data into multiple specific classes rather than just distinguishing between benign and malicious attack . The main focus of this contrastive analysis is to examine various machine learning and deep learning models, each possessing distinct abilities and capacities to detect network intrusions. Our project evaluates the effectiveness of traditional algorithms including Logistic Regression, Decision Trees, Random Forest, Naive Bayes Classifier, and Support Vector Machines, as well as cutting-edge neural networks like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) such as LSTMs. This evaluation is based on various metrics such as accuracy, F1 score, recall and precision

Chapter 2: Dataset and Preprocessing

We have utilized two benchmark datasets that are highly regarded for their large size and applicability to real-world network environments. These datasets are the CIC-IDS2017 [1] and the UNSW-NB15[2]. Each dataset has been carefully selected to offer a practical simulation of network traffic, both benign and malicious, and incorporates a range of various attack types that are crucial for the development of efficient IDS solutions.

Researchers may choose to analyze network traffic at either the flow-level or packet-level. Flow-level analysis involves summarizing network communications by collecting metadata about data streams between sources and destinations using specific protocols. The concise representation of traffic patterns and behaviors in this study is advantageous, as it enables quick processing and examination. Packet-level data offers detailed understanding by recording individual packets that pass over network tiers, including their payloads, providing a wealth of information for in-depth study.

In this project, our focus was on analyzing flow-level data. This method was chosen for its efficient handling of a large volume of network traffic data. It achieves a balance between providing sufficient detail and being manageable, thereby ensuring precise classification of network attacks.

2.1 CICIDS 2017

CICIDS 2017 is a publicly accessible dataset published in [1]. It is a publicly available collection of data. It contains over 3 million labeled data points and includes 80 flow-based characteristics. Additionally, it encompasses 14 different classes of attack types. The labeling is comprehensive, as it includes timestamps, source and destination IPs, source and destination ports, protocols, and the type of attack for each data-point. This level of accuracy ensures that researchers and security experts can effectively identify and assess suspicious activities within the dataset. The CIC-IDS[1] dataset incorporates a diversified range of attack types, such as SSH brute force, heartbleed, botnet, DoS, DDoS, web attacks, infiltration, and other various attacks, as can be seen in Figure 1.

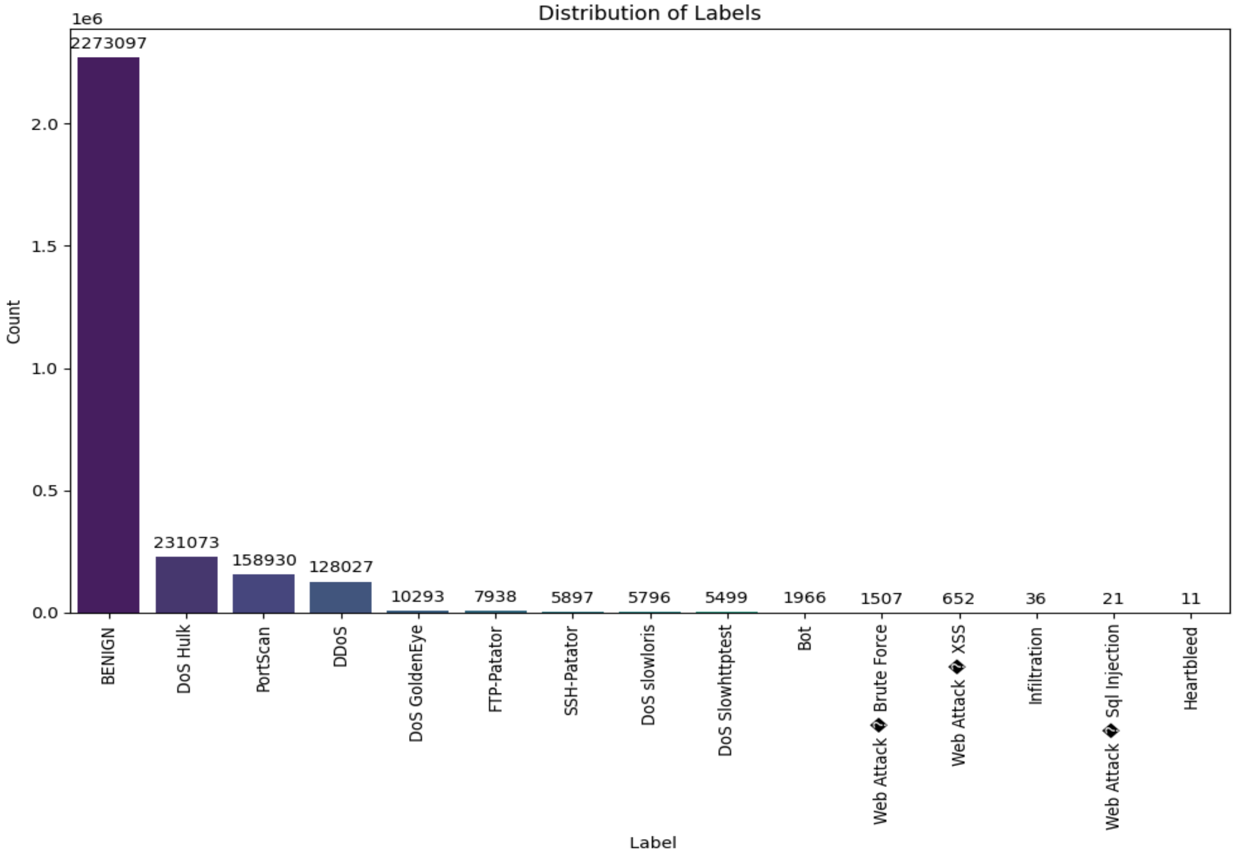


Figure 1 : Distribution of classes in CIC-IDS2017

2.2 UNSW NB-15

The UNSW-NB15 dataset is a publicly accessible collection of network traffic data available to the public, commonly utilized in cybersecurity research to develop and evaluate network intrusion detection systems. The dataset, which was published in a research article [2], has comprehensive traffic-related data. The Australian Centre for Cybersecurity's Cyber Range Lab utilized the IXIA Perfect Storm program to create the UNSW-NB15 dataset. This dataset comprises approximately 2.5 million packets, encompassing both authentic normal and attack traffic scenarios. This dataset can be accessed in different formats, including BroIDS, CSV, PCAP, and Argus. The CSV files only represent about 10% of the entire dataset [3]. The dataset includes 49 characteristics for each entry, providing a comprehensive set of data properties for analysis. The labeling is comprehensive, distinguishing each data point based on date, source and destination IPs, source and destination ports, protocols, and the kind of attack, guaranteeing accurate recognition and evaluation of suspicious activity. This dataset includes a total of 2,540,044 records. The data is divided into four CSV files: UNSW-NB15_1.csv, UNSW-NB15_2.csv, UNSW-NB15_3.csv, and UNSW-NB15_4.csv. Furthermore, the dataset has been separated into two particular partitions: a training set and a testing set. The training set is accessible as UNSW_NB15_training-set.csv, while the testing set is available as UNSW_NB15_testing-set.csv. The training set includes 175,341 records, and the testing set includes 82,332 records. Both sets contain a mix of malicious and normal network traffic, providing a comprehensive environment for developing and assessing ML models. We are using this training set and testing set to train our models.

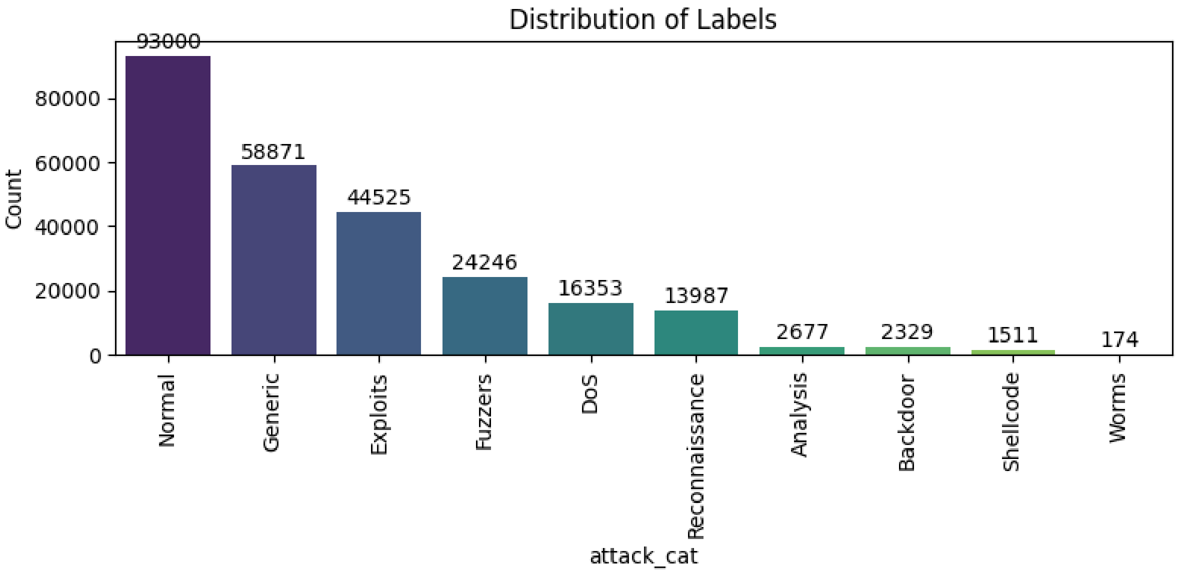


Figure 2 : Distribution of classes in UNSW-NB15 (Train and Test set)

The UNSW-NB15 dataset includes nine distinct categories of attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shell code, and Worms as can be seen in

Figure 2. The dataset contains a variety of contemporary attack patterns, making it a valuable asset for training and evaluating intrusion detection techniques.

2.3 Dataset Imbalance

The presence of notable class imbalances in both the CIC-IDS-2017 and UNSW-NB15 datasets is evident in Figures 1 and 2. This existing imbalance raises a notable issue, as it could lead to machine learning models favoring the majority class, potentially neglecting the classification of malicious minority classes. Maintaining a balanced dataset for training that accurately represents all classes is crucial for achieving optimal performance.

2.3.1 Sampling Techniques for Balance

To create a balanced dataset, two main sampling approaches are employed: oversampling and undersampling. Oversampling involves duplicating data points from the minority class, while undersampling involves removing data points from the majority class. While both procedures are simple, they come with their respective drawbacks. Such as oversampling may lead to overfitting, while undersampling can result in the loss of important information.

2.3.2 Our Approach

We have explored two different methods to address the imbalance in the CICIDS2017 & UNSW NB-15 datasets. These approaches aim to equalize the proportions of malicious classes all together and benign class in the CICIDS2017 dataset. The strategies considered are the almost balanced strategy and the minority favored strategy. We utilized a single strategy for the UNSW dataset, as it produced satisfactory results without the need for sampling. To enhance the accuracy of the UNSW dataset, a similar minority favored approach was put into place. The main objective of these strategies is to improve the effectiveness of model training and performance, which is particularly crucial in the cybersecurity domain. Accurately identifying various attack types is of the utmost significance in this field.

The first approach used in CICIDS2017 seeks to achieve a high level of equality across the classes, with a malicious-to-normal (minority to majority) class ratio of around 0.994:1. Initial tests on the CICIDS dataset without employing sampling resulted in significantly poor performance in accurately detecting minority classes. The goal of this almost balanced strategy is to reduce any potential favoritism towards a specific class, which was seen when the CICIDS dataset was trained without sampling, ensuring that the model is equally skilled at recognizing both minority and majority classes. The second approach slightly overrepresents the minority classes, with a ratio of approximately 1.12:1. This method proves beneficial when the main goal is to improve the detection of minority attack types that could potentially pose a greater risk. By giving priority to the minority classes, models trained with this strategy may show heightened sensitivity towards the classes linked to minority attacks.

The approach used for balancing the UNSW-NB15 dataset differs from the two methods implemented for the CICIDS-2017 dataset. For the UNSW-NB15 dataset, we have adopted a single strategy to address the imbalance. This approach involves employing a technique that favors the minority classes. It includes oversampling the minority classes significantly to ensure they are adequately represented in the dataset. A cumulative total of 154,500 samples have been distributed for the minority classes, resulting in a malicious-to-normal (minority-to-majority) class sample ratio of roughly 2.21:1. This strategy is to enhance the ability of machine learning models to detect minority classes by giving them a numerical advantage over the majority class.

A mix of random undersampling and the synthetic minority oversampling technique (SMOTE) was used to apply these strategies to training sets after the test/train split. To reduce the size of the dominant class, random undersampling was used, whereas SMOTE was utilized to enhance the representation of the minority classes. The use of this dual methodology enables the manipulation of the dataset in order to get the intended class ratios, hence augmenting the training process.

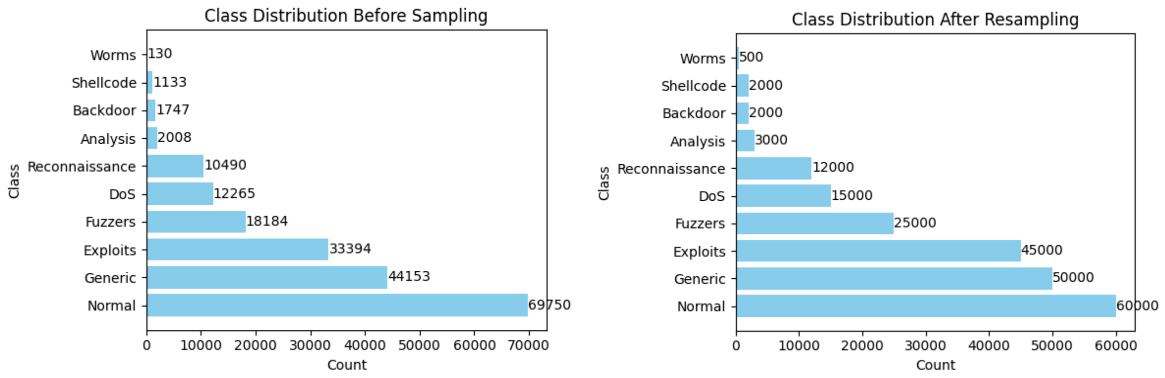


Figure 3 : Training Set Class Distribution Before and After Resampling (UNSW NB15)

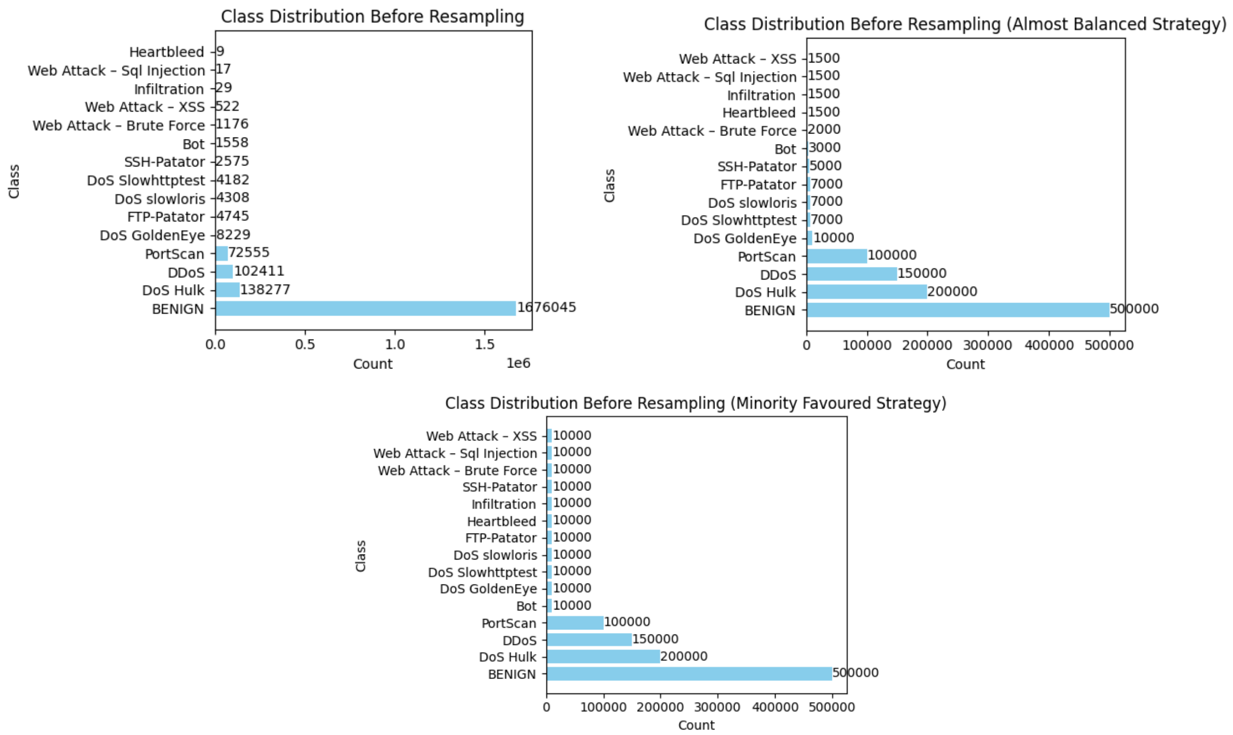


Figure 4 : Training Set Class Distribution Before and After Resampling (CICIDS2017)

2.4 Dataset Preprocessing

The CICIDS-2017 dataset was available into daily captures and extracted into separate data frames. These individual data frames were then merged together to create a unified and comprehensive dataset. The combined dataset was checked for missing values and infinities, which were replaced with NaNs. Subsequently, any rows containing NaNs were removed to ensure data integrity. In order to avoid duplication and potential overfitting of the model, any duplicate entries were identified and eliminated. Additionally, columns with zero variance were excluded from the data frame.

The variable 'Label' was encrypted numerically to clarify the understanding of category labels in the model. Two subsets were created from the dataset: the training set and the test set, with the test set consisting of 20% of the data. We utilized stratification to keep the class distribution unchanged. In order to handle the uneven distribution of classes, we utilized resampling techniques on the training set. The MinMaxScaler was utilized to normalize the features, ensuring that all machine learning models had uniform input dimensions. Ensuring proper reshaping of the data was crucial to meeting the specific input shape criteria of certain deep learning models, such as Conv1D. Additionally, we utilized one-hot encoding to represent the target classes in the required format for classification.

The preprocessing method applied to the UNSW dataset was quite similar to that used for the CICIDS-2017 dataset, with some significant adjustments made to accommodate its unique characteristics. One key change was the implementation of One-Hot Encoding for categorical features like 'proto', 'service', and 'state', which translated them into numerical representations suitable for model consumption. Furthermore, several sampling techniques were employed to enhance the quality of the UNSW dataset.

Chapter 3: Model Development, Training

3.1 Model Development

In this chapter, we explore the practical application of multiple supervised machine learning techniques to determine their effectiveness in multi-label classification scenarios. By making use of the preprocessed datasets, we investigated the complexities of intrusion detection, employing multiple algorithms, each possessing distinct advantages and methodologies for identifying patterns within the data.

3.1.1 Implemented Models

For classification purposes, we have employed conventional machine learning algorithms such as logistic regression, support vector machines (SVM), decision trees (DT), random forests (RF), and naive bayes. In addition to these typical models, we also included deep learning techniques, specifically convolutional neural networks (CNNs) and long short-term memory networks (LSTMs).

Logistic Regression

Logistic regression is an essential statistical approach employed specifically for binary classification tasks [4]. Even though it is used for binary classification, it has the capability to incorporate more classes through techniques such as one-vs-rest (OvR) or multinomial logistic regression. This is especially helpful for analyzing the detailed and complex data in the CICIDS 2017 and UNSW-NB15 datasets. It can serve as a benchmark for evaluating complex models. This model's usefulness lies in its capability and its ability to interpret probabilities, which is crucial for understanding the probability of various malicious attacks.

Support Vector Machines (SVM)

Support Vector Machines (SVM) stand out as a reliable collection of supervised learning techniques commonly applied in classification and regression tasks. SVM's primary concept involves determining the hyperplane that best separates different classes by maximizing the margin between them. Within SVM, the margin is described as the gap between the hyperplane (decision boundary) and the closest data points from each class, termed support vectors. The primary goal of SVM in maximizing the margin is to maintain a significant separation between the classifier and the nearest training data points of any class, leading to better generalization on unseen data. [5].

While support vector machines (SVM) are primarily designed for binary classification tasks, they may be expanded to tackle multiclass classification issues via the use of techniques like one-vs-rest (OvR) or one-vs-one (OvO).

SVM's capacity to identify complex decision boundaries may be very beneficial in the CICIDS2017 and UNSW-NB15 datasets, which consist of advanced and subtle attack categories. By using Support Vector Machines (SVM) with a suitable kernel function and a multiclass approach, it becomes feasible to develop a robust classifier capable of effectively detecting various types of network intrusions.

Decision Trees

Decision trees are non-parametric methods used for supervised learning in both classification and regression applications [6]. They are highly preferred due to their simplicity in understanding and visualization, since they provide a hierarchical representation of choices. The algorithm divides the dataset into branches, producing a decision outcome that is representable and understandable as a series of "if-then-else" rules.

Decision Trees are essentially capable of performing multiclass classification. In a multiclass setting, the tree structure will consist of several leaves, with each leaf representing a distinct class. The approach operates by using a sequence of binary splits to partition the data into subsets that exhibit increasing homogeneity in relation to the target variable as the tree expands.

Decision trees have been utilized in several intrusion detection systems (IDS) within the framework of IDS, such as [7]. Decision trees are very valuable in network intrusion detection as they can effectively find the most predictive elements for various forms of network attacks. The ability of the model to effectively process various data types and capture non-linear associations makes it very suitable for the complex and heterogeneous data often seen in network traffic.

Random Forest

Random Forest utilizes ensemble learning, wherein multiple decision trees are constructed simultaneously as part of the training phase. Following this, these decision trees are used to determine the class that most accurately represents the mode of the classes (in the case of classification) or the mean prediction produced by the individual trees (in the case of regression). This approach proves highly effective for both classification and regression purposes, as it mitigates overfitting and enhances prediction accuracy by adding together the results of each tree.

Random Forest demonstrates its capability in properly handling the multiclass composition of our datasets. Within the datasets, there exist various attack types, each identifiable by unique signature patterns. By employing its ensemble setup, Random Forest has the capability to differentiate between these patterns, as individual trees may focus on detecting particular kinds of attacks. The collaborative decision-making process adopted by Random Forest leads to reliable classification.

Naive Bayes

Naive Bayes classifiers are a set of straightforward probabilistic classifiers that depend on the application of Bayes' theorem, assuming strong (naive) independence between the features. They are renowned for their exceptional efficiency and capability in classification tasks, particularly when dealing with fewer data in contrast to more complex methods.

Types of NB classifiers

- Gaussian Naive Bayes: This approach is most appropriate for datasets that have continuous or real-valued characteristics and are expected to follow a normal distribution.

- Multinomial Naive Bayes algorithm: These feature vectors are compatible for capturing the frequencies at which certain events have occurred, such as word counts in the context of text categorization.
- Bernoulli Naive Bayes algorithm: This system is particularly designed to handle binary/boolean features, whereby the features are self-reliant boolean variables that explain the inputs. This approach is applicable to multivariate Bernoulli data, specifically where the features consist only of binary digits (0s and 1s).
- Categorical Naive Bayes : well-suited for categorization tasks with distinct characteristics that follow a categorical distribution.

While selecting a Naive Bayes method for the CICIDS and UNSW datasets, it was really important to take into account the distribution of features. These datasets have a wide range of characteristics. Even though each variation has its perks, the Gaussian Naive Bayes algorithm came out on top, proving that the continuous features in the datasets were key for classification and that the normal distribution effectively captured their importance. The decision to use Gaussian Naive Bayes was made due to the circumstances mentioned above.

Convolutional Neural Network (CNN)

Convolutional neural networks (CNNs) are a distinct kind of neural network designed specifically for the purpose of processing data with a grid-like structure, such as images. Convolutional Neural Networks (CNNs) have been shown to be effective in a wide range of applications, such as image and video recognition, natural language processing, and time series prediction.

Although CNNs are well known for their efficiency in computer vision tasks, they can also be adjusted to handle other forms of data, such as sequential data. The aforementioned adaptability enables them to be well-suited for the analysis of time-series network traffic, which is common in datasets like CICIDS and UNSW. In order to boost the productivity of Convolutional Neural Networks (CNNs) on our datasets, it is essential to transform the data into a format that preserves spatial correlations. Convolutional Neural Networks (CNNs) utilize the ordering of packets or temporal characteristics as spatial dimensions, allowing the network to efficiently capture temporal patterns.

Long Short Term Memory (LSTM)

LSTMs represent a specific type of Recurrent Neural Network (RNN) capable of understanding long-term relationships within sequential input data. LSTMs are designed specifically to address the issue of the vanishing gradient problem, which presents obstacles for traditional RNNs in gradient-based learning methods and long-term dependency challenges. Long Short-Term Memory (LSTM) models have shown noteworthy efficiency in a various range of applications relating to sequential data, including but not limited to language modeling and translation, voice recognition, and anomaly detection in time series data. LSTMs are very suitable for multiclass classification problems that use sequential data, such as the CICIDS and UNSW datasets.

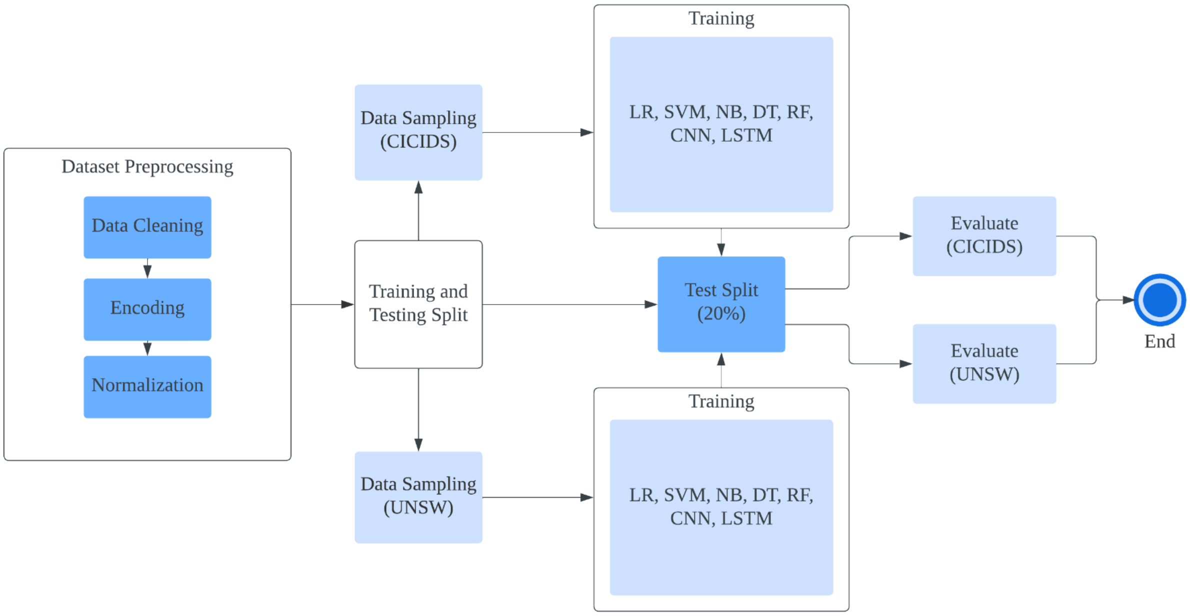


Figure 5 : Overview of Data Processing and ML Implementation

3.2 Training

3.2.1 Logistic Regression

A Logistic Regression model was set up with a high number of iterations to aid in achieving convergence. The 'saga' solver was selected due to its productiveness when working with comprehensive sets of data. A grid search was performed across various regularization strengths and penalties to ascertain the optimal hyperparameters. The grid search was carried out with StratifiedKFold cross-validation to maintain class proportions in each fold, hindering bias from class imbalance in the model's performance.

The training procedure for the UNSW dataset emulated that of the CICIDS. The model development approach remained consistent by using the same grid search strategy and cross-validation technique. Maintaining this consistency is essential to ensuring a fair evaluation of the model's performance on various datasets. After the grid search was finished, the optimal hyperparameters for the CICIDS dataset were found to be a penalty value of 'l2', a regularization strength 'C' of 1, and a maximum iteration limit of 10,000. The UNSW dataset revealed that the best hyperparameters included a penalty of 'l1' and a regularization strength 'C' value of 1.

Subsequent to discovering the optimal hyperparameters, the logistic regression models were trained on their individual datasets. The performance of the models was evaluated using a different test set that was not used in the training process. This assessment focused on testing the model's capacity to apply to fresh, unknown data.

3.2.2 Support Vector Machines

A random grid search with 3-fold cross-validation was utilized in the Linear SVM training approach for optimizing hyperparameters in both the CICIDS and UNSW datasets. The distribution of parameters was established to evaluate a variety of regularization strengths, loss functions, multi-class strategies, and iteration limits. More specifically, the value of the regularization parameter C was examined at 0.1,1 and 10, the loss function switched between 'hinge' and 'squared_hinge', the multi-class approach switched between 'ovr' (one-vs-rest) and 'crammer_singer', and the maximum number of iterations was either 1000 or 10000. The grid search with randomness effectively explored the hyperparameter space to discover the best configurations for the SVM model. This method is particularly beneficial for handling comprehensive datasets and a complex hyperparameter space, as it can explore a broad range of values without the need for a time-consuming exhaustive search.

For both datasets, the search determined that the optimal hyperparameters were a C value of 1, showing a typical trade-off between maximizing margin and minimizing classification error. The ‘squared_hinge’ loss function showed good results and was chosen because it is widely used in SVM classification tasks for its resilience. The optimal method for multi-class classification was determined to be the 'ovr' strategy, with 1000 iterations being enough for the model to converge.

3.2.3 Decision Tree

In order to find the optimal hyperparameters, again a grid search method was used for decision tree models on the CICIDS and UNSW datasets. The procedure included utilizing the DecisionTreeClassifier, along with a range of possible values for important hyperparameters: max_depth for regulating tree depth, min_samples_split and min_samples_leaf for determining node split conditions, and a criterion for choosing the metric to evaluate split quality. The grid search was set up to employ 5-fold cross-validation.

The thorough process of grid search resulted in finding the optimal hyperparameters for every dataset. The best combination for the CICIDS dataset was found to be a max_depth of 20, min_samples_split of 10, min_samples_leaf of 4, and utilizing the entropy criterion. However, the UNSW dataset leaned towards a slightly varied combination of hyperparameters. The optimal outcomes were obtained when using a max_depth of 20, min_samples_split of 2, min_samples_leaf of 1, and the ‘gini’ criterion.

3.2.4 Random Forest

The Random Forest Classifier was fine-tuned using GridSearchCV using a predefined set of hyperparameters for both the CICIDS and UNSW datasets. The parameters taken into account in the grid were the number of trees (n_estimators) with values ranging from 50, 100, and 200, the maximum depth of the trees (max_depth) with options of None, 10, 20, and 30, the minimum number of samples for splitting (min_samples_split) with values of 2, 5, and 10, the minimum number of samples at the leaf node (min_samples_leaf) with values of 1, 2, and 4, and the split quality measurement function (criterion) with values of 'gini' and 'entropy'.

Upon applying the grid search to the data, the optimal parameters for the CICIDS dataset were determined to be: 'gini' criterion, maximum depth of None (signifying that trees grow until all leaves are pure or contain fewer than the minimum samples for splitting), minimum samples per leaf of 1, minimum samples for splitting of 2, and 200 estimators. This indicates that having more mature trees with fewer constraints on leaf size and splitting behavior resulted in the highest accuracy for this data set.

The top parameters for the UNSW dataset were slightly varied, suggesting a customized fit for this specific data. The best parameters included the ‘gini’ criterion, maximum depth of 30, minimum samples per leaf of 1, minimum samples per split of 2, and 200 estimators. In this case, a particular maximum depth was established, indicating that controlling the depth at a certain level improved model performance by possibly balancing overfitting and enabling the model to grasp complex patterns.

3.2.5 Naive Bayes

Gaussian Naive Bayes was selected as the classifier for the CICIDS and UNSW datasets because of their specific features. A grid search was carried out to determine the optimal hyperparameter for this model, using var_smoothing as the parameter to adjust. var_smoothing helps prevent overfitting by smoothing the distribution's variance and considering additional samples situated far from the mean. The grid search evaluated var_smoothing with 100 values by using np.logspace(0, -9, num=100), which evenly distributes numbers on a logarithmic scale.

Following grid search with 5-fold cross-validation, the optimal parameter for the CICIDS dataset was identified as var_smoothing set to 1e-09. This suggests that only a tiny amount of smoothing was required, indicating that the model needed to closely match the data with minimal variance adjustment.

In the UNSW dataset, the Gaussian Naive Bayes model found the best var_smoothing parameter of around 0.43287612810830584 through grid search. The var_smoothing value indicates that there was a beneficial level of smoothing applied to the UNSW dataset.

3.2.6 Convolutional Neural Network (CNN)

Two different training approaches were used when applying convolutional neural networks (CNNs) to the CICIDS and UNSW datasets for intrusion detection: one with no fine-tuning and one with fine-tuning, each utilizing varied sampling methods to address class imbalance.

At first, for the CICIDS dataset, the CNN model was developed for multi-class classification without fine-tuning. The design of the model featured a Conv1D layer with 64 filters and a kernel size of 3, using ‘relu’ activation. This was followed by a MaxPooling1D layer, a flattening layer, a dense layer with 50 neurons (also employing 'relu' activation), and a dropoutt layer with a rate of 0.5 to avoid overfitting. The dense layer at the final output had a softmax activation function, with the number of neurons matching the number of classes in the dataset. The 'adam' optimizer and 'categorical_crossentropy' loss function were employed when compiling the model, which is ideal for multi-class classification. The model was subsequently trained on the normalized training data with one-hot encoded tags for 10 epochs, utilizing a batch size of 32, and validation was carried out on the normalized test data.

To fine tune the CNN model, an EarlyStopping callback function was used to stop training if the validation loss stopped improving, helping to avoid overfitting. Furthermore, a ReduceLROnPlateau callback was employed to decrease the learning rate once the validation loss stopped getting better, enabling finer enhancements in subsequent training cycles. The refined model architecture closely resembled the original one, with a Batch Normalization layer added after the Conv1D layer to normalize the activations. The Dropout rate was decreased to 0.2 to preserve additional neuron connections. Minority Favored Resampling was done to give a bit more representation to the minority classes, with a target ratio of around 1.12:1 in order to tackle class imbalance. The fine-tuned model was built with the 'adam' optimizer and 'categorical_crossentropy' loss function. The model was trained with the same normalized training data and one-hot encoded labels, now including class weights and using callbacks for early stopping and reducing the learning rate.

For the UNSW dataset, two distinct training strategies were used for the Convolutional Neural Network (CNN) model, mirroring those in the CICIDS dataset. One method did not involve sampling, while the other method used a sampling approach to enhance the classification of minority classes.

The CNN model consisted of two convolutional layers, each with 32 filters, and two more layers with 64 filters, using 'relu' activation without fine-tuning. MaxPooling and a small dropout rate of 0.002 were implemented following the convolutional layers, in addition to batch normalization, to enhance learning stability and optimize gradient flow in the network. The structure concluded with a flattening layer, followed by a dense layer containing 32 neurons and a softmax output layer. The model was built using the 'adam' optimizer and 'categorical_crossentropy' loss function, trained for 10 epochs with a batch size of 32, without any particular sampling strategy, with the goal of reaching a baseline accuracy for the dataset.

The CNN model was fine-tuned and had the same structure as the non-fine-tuned model, however, it had small modifications. Batch normalization was used right after the initial Conv1D layer for early normalization, enhancing learning stability, gradient flow, and regularization to boost fine-tuning efficiency. This is especially useful during the fine-tuning stage, when the learning rate is modified and the model is enhanced to be more responsive to the minority classes through a sampling approach. Methods such as EarlyStopping and ReduceLROnPlateau were used to avoid overfitting and improve convergence during the fine-tuning process. The model was built using the 'adam' optimizer and 'categorical_crossentropy' loss function, and was trained with a sampling technique to improve the classification of minority classes.

3.2.7 Long Short Term Memory (LSTM)

The initial LSTM model for the CICIDS dataset, includes a sequential structure beginning with a 64-unit LSTM layer that utilizes ‘tanh’ activation and is configured to output sequences. This is then followed by a 0.2 dropout in order to reduce overfitting. There is an additional LSTM layer with 32 units that handles the output from the initial layer, followed by another dropout layer with the same rate. After that, the model adds a dense layer with 50 neurons and uses ‘relu’ activation, followed by a softmax output layer that matches the number of classes. The Adam optimizer is employed in the model with a learning rate of 1e-4. It undergoes training using callbacks for EarlyStopping and ReduceLROnPlateau to avoid overfitting and fine-tune the learning rate for improved training results. No sampled data was used to train this model as a baseline model.

The LSTM model that has been fine-tuned maintains the same structure as the baseline version, but it is trained using a strategy that gives preference to minority groups in order to improve the classification of classes that are not well represented. This method aids the model in prioritizing the minority classes while undergoing training. The model's learning process is fine-tuned using the same callbacks for EarlyStopping and ReduceLROnPlateau to prevent overfitting and ensure good generalization to new data. The optimizer and loss function are the same as in the initial model.

For the UNSW dataset, the initial LSTM model begins with an LSTM layer containing 100 units, uses ‘tanh’ activation, which aims to capture temporal relationships, and is set to return_sequences=True to send sequential data to the subsequent layer. A 0.2 dropout is included to avoid overfitting. The next LSTM layer contains 50 units, uses ‘tanh’ activation, and does not produce sequences, showing a consolidation of time-based data into a lone vector. Before the final dense layer, there is an additional dropout of 0.2 used. This layer utilizes a softmax activation function to categorize the data into the various attack classes found in the dataset. The model is built using the 'adam' optimizer and 'categorical_crossentropy' loss function, incorporating callbacks like EarlyStopping and ReduceLROnPlateau to improve model performance and avoid overfitting while training.

The layout of the fine-tuned LSTM model for the UNSW dataset remains consistent, however, the training data undergoes preprocessing with a sampling technique to tackle class imbalance. This approach guarantees that the model is presented with a balanced dataset, improving its capability to classify minority classes. The model is trained using identical optimizer, loss function, and callbacks as the baseline model. The main goal of fine-tuning is to optimize the model's performance, especially in accurately classifying the minority classes in the dataset.

Chapter 4: Evaluation

4.1 Evaluation Metrics

Evaluation metrics play a crucial role in analyzing the effectiveness of algorithms in machine learning (ML) and deep learning (DL) models. The selection of metrics is dependent upon the specific characteristics of the task at hand, such as classification, regression, clustering, and so on. This section will mostly focus on the fundamental metrics used in the evaluation of classification models.

True Positive (TP) : When the model accurately detects an attack. A large quantity of TPs shows that the model is good at identifying real threats.

True Negative (TN): When the model accurately identifies non-malicious traffic as safe. TNs are essential in making sure that false alarms do not disrupt regular network operations.

False Positive (FP) : Incorrectly classifying legitimate traffic as an attack by the model results in a False Positive. False positives can be expensive because they can result in unnecessary inquiries and may interrupt valid operations.

False Negative (FN): A situation where the model mistakenly identifies real attacks as normal traffic. FNs pose a significant threat since they signify missed detections, which enable malicious activities to remain undetected.

Accuracy (AC)

Accuracy offers a quick look at the general correctness of the model by taking into account both true positives and true negatives in comparison to all predictions. Nevertheless, for datasets like CICIDS and UNSW with imbalanced classes, accuracy may not provide an accurate reflection of the model's performance due to TNs having a disproportionate impact.

$$AC = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

This metric is essential in cases where a false positive can have serious impacts. It is the proportion of accurately predicted positive instances out of all predicted positive instances. High precision means that when the model predicts an attack, it is likely to be correct.

$$Precision = \frac{TP}{TP + FP}$$

Recall (Sensitivity)

Recall is particularly important in assessing our model because failing to detect an actual attack can be very costly. It is the ratio of correctly anticipated positive observations to all actual positives. A high recall suggests that the model is capable of noticing the majority of actual attacks.

$$Recall = \frac{TP}{TP + FN}$$

F1 Score

The F1 score is a balance between accuracy and recall, which is important when you need to take both false positives and false negatives into account. F1 Score is especially beneficial for the CICIDS and UNSW datasets due to its ability to offer a more fair assessment by taking into account both accuracy and recall.

$$F1\,Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Confusion Matrix

This table does not represent a metric, but rather displays the classification model's performance visually. It displays the matrix format of true positives, false positives, true negatives, and false negatives. This helps in determining how well the model performs with different classes and is crucial to assess the model's ability to differentiate between various types of traffic, including different attack vectors.

4.2 Overview of Model Performance (CICIDS Dataset)

The wide range of attack types in the CICIDS dataset poses a significant challenge for network intrusion detection systems. Our study evaluated the effectiveness of various supervised machine learning and deep learning models in different attack categories. Models such as Random Forest (RF) and Decision Tree (DT) showed strong performance overall by effectively identifying both majority and minority classes.

4.2.1 Challenges with Minority Classes

Nevertheless, specific models like Logistic Regression (LR), Support Vector Machines (SVM), and Naive Bayes (NB) faced challenges when it came to effectively identifying minority classes such as the three web attack classes (SQL Injection, XSS and Brute Force), infiltration, and heart bleed. Equivalent difficulties were noted in deep learning models such as Long Short-Term Memory networks (LSTMs) and Convolutional Neural Networks (CNNs), as a nearly equal approach produced similar outcomes to LR, SVM, and NB, highlighting a difficulty in capturing the subtle patterns of minority classes.

4.2.2 Strategy Adjustments

To address these challenges, two key strategies were employed: adjustments in sampling strategies and the introduction of balanced weights during model training.

Adjustments were made to the sampling strategy to improve the models' sensitivity to minority classes, with different strategies explored for both versions without fine-tuning and with fine-tuning. This method was intended to offer a more equitable portrayal of classes in model training, consequently enhancing the classification of minority classes of attacks.

Balanced Weights: The class_weight='balanced' argument was added during training to assist models like LR, SVM, and CNNs that struggle with minority classes. This modification directed the models to focus more on minority classes to balance their lack of representation in the dataset and reduce the bias towards the majority classes. DT, RF, NB, and LSTMs showed impressive performance even without requiring balanced weights, demonstrating their natural ability to handle class imbalance with robustness or flexibility.

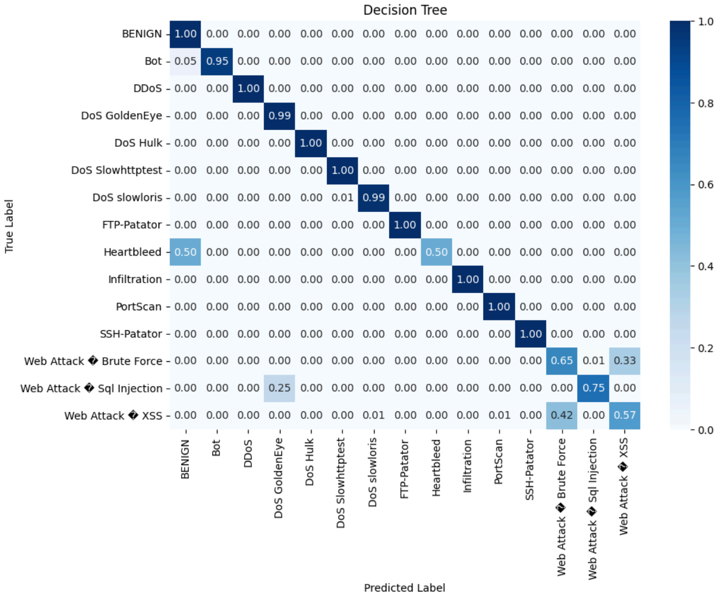
On top of that, additional training techniques were used to improve the performance of deep learning models such as convolutional neural networks (CNNs) and long short-term memory networks (LSTMs). Early stopping was used to avoid overfitting and ensure models stop training when their performance on a validation set begins to decline. This method watches a particular performance measurement and stops the training if there is no significant improvement in the measurement for a set number of epochs. Apart from this, ReduceLROnPlateau approach was employed. This method adjusts the learning rate based on changes in model performance, enabling more precise tuning during the learning process to enhance the accuracy and efficiency of the model.

4.2.3 Model Evaluation and Insights

To thoroughly assess how the strategies have affected the model's performance, we provide a confusion matrix for each model. This comparison study enables us to directly witness the alterations in model performance.

Decision Tree and Random Forest

The decision tree trained using an almost balanced strategy provided the best results and demonstrated a strong capability of accurately recognizing most attack classes. Nonetheless, there are inaccuracies in identifying minority classes, especially in the 'Heartbleed' category, with only a 50% accuracy rate, as well as significant mix-ups between the 'Web Attack – Brute Force' and 'Web Attack – XSS' classes. This suggests that although the model excels in detecting common attack types, it faces challenges in differentiating between comparable minority attack patterns.



	BENIGN	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS slowloris	FTP-Patator	Heartbleed	Infiltration	PortScan	SSH-Patator	Web Attack � Brute Force	Web Attack � Sql Injection	Web Attack � XSS
BENIGN	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Bot	0.05	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DDoS	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DoS GoldenEye	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DoS Hulk	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DoS Slowhttptest	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DoS slowloris	0.00	0.00	0.00	0.00	0.00	0.01	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
FTP-Patator	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Heartbleed	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00
Infiltration	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
PortScan	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
SSH-Patator	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
Web Attack � Brute Force	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.01	0.33
Web Attack � Sql Injection	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.75	0.00
Web Attack � XSS	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.42	0.00	0.57

Figure 6 : Decision Tree CM (Almost Balanced Strategy)

The Random Forest model, with its almost balanced strategy, provided the best results and showed a small enhancement compared to the Decision Tree in distinguishing between 'Web Attack – Brute Force' and 'Web Attack – XSS' categories, yet it continues to misclassify 50% of 'Heartbleed' attacks. The higher true positive rate in detecting 'Web Attack - Brute Force' is significant compared to the Decision Tree model.

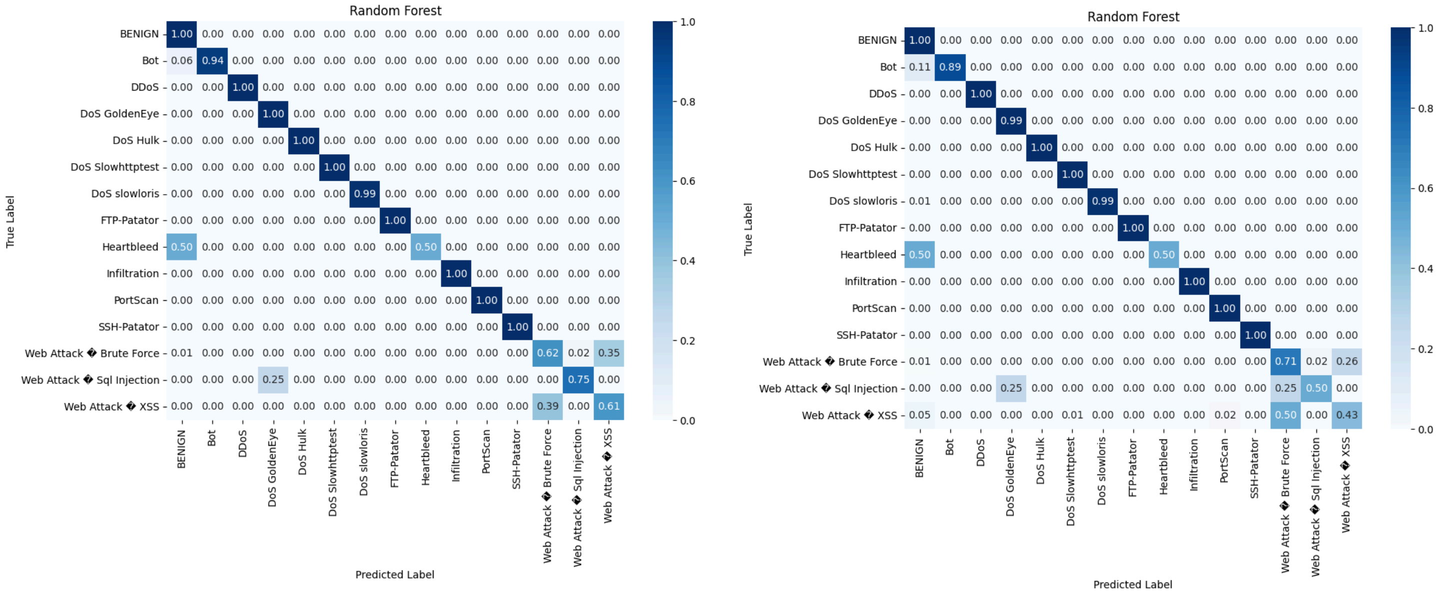


Figure 7 : Random Forest CM (Almost Balanced Strategy, Minority Favored Strategy)

By implementing a strategy that gives preference to minority classes, by increasing the number of data points in those classes, we saw similar results in the Random Forest model's predictions. The model still continues to misclassify 50% of 'Heartbleed' attacks. The detection of ‘Web Attack - SQL Injection’ has decreased from 75% to 50%. The detection of 'Web Attack – Brute Force' has increased, the detection of 'Web Attack – XSS' showed a small decrease, suggesting a compromise when prioritizing certain minority classes.

Logistic Regression

In an almost balanced strategy, the Confusion Matrix for LR displays a high rate of true positive for the majority classes. Nevertheless, the model faces challenges when dealing with certain minority classes such as 'DoS Slowloris', 'Heartbleed', ‘Infiltration’ and the subclasses of 'Web Attack' ('Brute Force', 'SQL Injection', 'XSS'), either detecting inaccurately or having a high misclassification rate. The 'Infiltration' and 'Web Attack' subclasses were completely overlooked, resulting in a false true positive rate of 1.00, showing that all instances were incorrectly grouped into other classes. This clearly shows the model's difficulty in dealing with these minority classes, a common occurrence in datasets with imbalances.

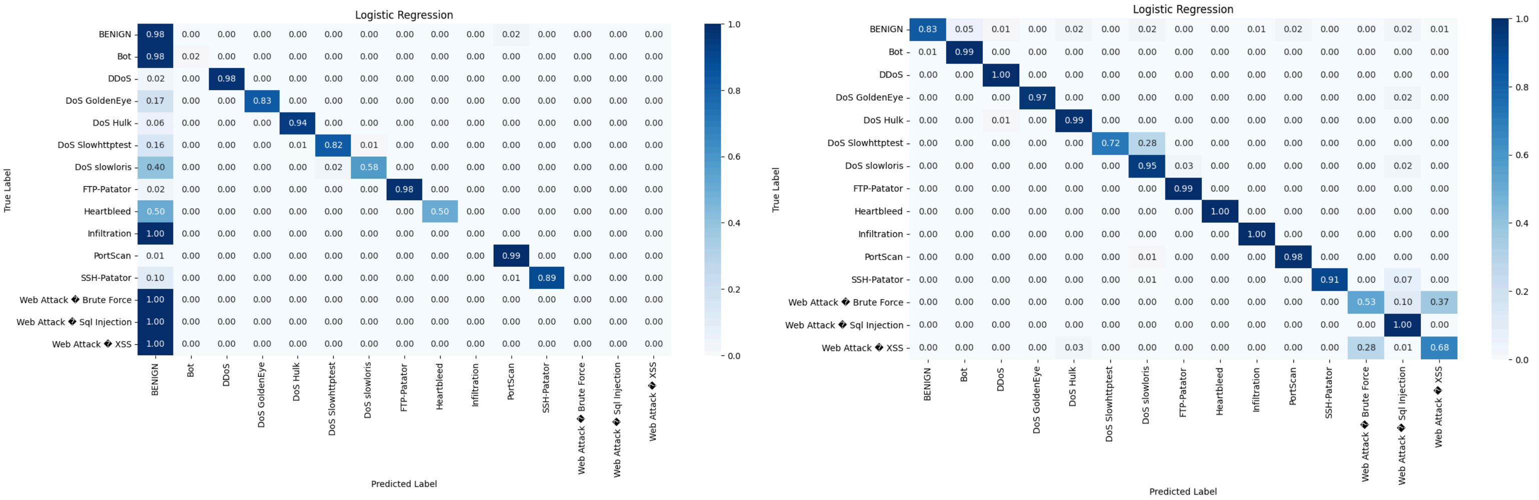


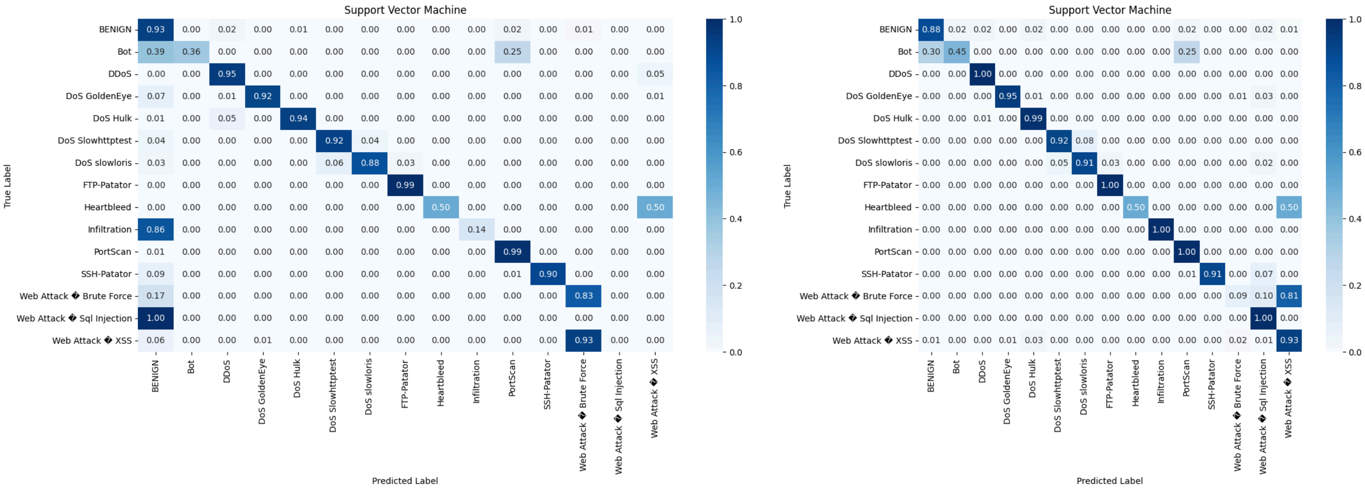
Figure 8 : Logistic Regression CM (Almost Balanced Strategy, Minority Favored Strategy)

After implementing a minority-favored strategy and adjusting class weights to be balanced, we observe a significant improvement in the model's ability to detect the minority classes. The true positive rates for 'DoS Slowloris' and 'Heartbleed' saw improvements, with 'Heartbleed' going from a 0.50 to a 1.00 detection rate and infiltration going from 0.00 to 1.00, indicating a perfect identification post-adjustment.

The 'Web Attack' subclasses show remarkable improvements as well. For 'Web Attack – Brute Force' and 'Web Attack – SQL Injection', the model now perfectly identifies these attacks. These improvements suggest that the minority-favored strategy and the balanced class weights allowed the LR model to learn more about the patterns and characteristics unique to these classes, thus improving its predictive accuracy.

Support Vector Machine

The SVM model implemented on the CICIDS dataset, which followed an almost balanced strategy, achieved notable accuracy in distinguishing benign traffic from the majority of attacks. However, it struggled notably with categorizing 'Bot' and 'Heartbleed' attacks, as well as web-based intrusions like 'SQL Injection' and 'XSS', resulting in numerous misclassifications. Additionally, the model had trouble identifying any cases of SQL Injection and XSS in web attacks. This shows that SVM is effective in detecting common attacks, but requires adjustments to better recognize minority classes.



True Label	BENIGN	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS slowloris	FTP-Patator	Heartbleed	Infiltration	PortScan	SSH-Patator	Web Attack � Brute Force	Web Attack � Sql Injection	Web Attack � XSS
BENIGN	0.93	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.00
Bot	0.39	0.36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00
DDoS	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
DoS GoldenEye	0.07	0.00	0.01	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
DoS Hulk	0.01	0.00	0.05	0.00	0.94	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DoS Slowhttptest	0.04	0.00	0.00	0.00	0.00	0.92	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DoS slowloris	0.03	0.00	0.00	0.00	0.00	0.06	0.88	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
FTP-Patator	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Heartbleed	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.50
Infiltration	0.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.00	0.00	0.00	0.00	0.00
PortScan	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00
SSH-Patator	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.90	0.00	0.00	0.00
Web Attack � Brute Force	0.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.83	0.00	0.00
Web Attack � Sql Injection	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Web Attack � XSS	0.06	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.00	0.00

True Label	BENIGN	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS slowloris	FTP-Patator	Heartbleed	Infiltration	PortScan	SSH-Patator	Web Attack � Brute Force	Web Attack � Sql Injection	Web Attack � XSS
BENIGN	0.88	0.02	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.02	0.01
Bot	0.30	0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00
DDoS	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DoS GoldenEye	0.00	0.00	0.00	0.95	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.03	0.00
DoS Hulk	0.00	0.00	0.01	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DoS Slowhttptest	0.00	0.00	0.00	0.00	0.00	0.92	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DoS slowloris	0.00	0.00	0.00	0.00	0.00	0.05	0.91	0.03	0.00	0.00	0.00	0.00	0.00	0.02	0.00
FTP-Patator	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Heartbleed	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.50
Infiltration	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
PortScan	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
SSH-Patator	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.91	0.00	0.07	0.00
Web Attack � Brute Force	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.10	0.81
Web Attack � Sql Injection	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
Web Attack � XSS	0.01	0.00	0.00	0.01	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01	0.93

.

Figure 9 : Support Vector Machine CM (Almost Balanced Strategy, Minority Favored Strategy)

Following adjustments made to the Support Vector Machine (SVM) model and the implementation of a strategy that favors the minority class, changes in the results for 'Bot', and web attacks such as 'SQL Injection' and 'XSS' can be observed in the revised Confusion Matrix. Within the 'Bot' class, there has been an increase in detection, resulting in a true positive rate of 45%. Meanwhile, the accuracy of identifying true positives in the 'Heartbleed' category remains constant. The model excelled in detecting 'SQL Injection' with a true positive rate of 1.00 and displayed significant improvement in identifying 'XSS' attacks. Despite the overall improvement, there was a sharp decrease in accuracy for detecting 'Brute Force' attacks, dropping from 0.83 to

0.09. These findings indicate that although the strategy favored by the minority has strengthened the SVM's ability to detect particular attack types, specifically 'Web Attacks', it has also resulted in a compromise, with a noticeable reduction in the detection of 'Brute Force' attacks.

Naive Bayes

In the almost equal strategy, the Naive Bayes model shows moderate results in differentiating most attack types, excelling in detecting 'DDoS' and specific 'DoS' attacks such as 'DoS GoldenEye' and 'DoS Hulk'. However, it struggles to accurately identify 'Bot' attacks, with a significant portion mistakenly labeled as 'BENIGN'. Detection of the 'Heartbleed' attack continues to be a challenge, achieving only a 50% success rate. Additionally, it demonstrates challenges with web attacks, particularly 'Brute Force', where a large portion are mislabeled.

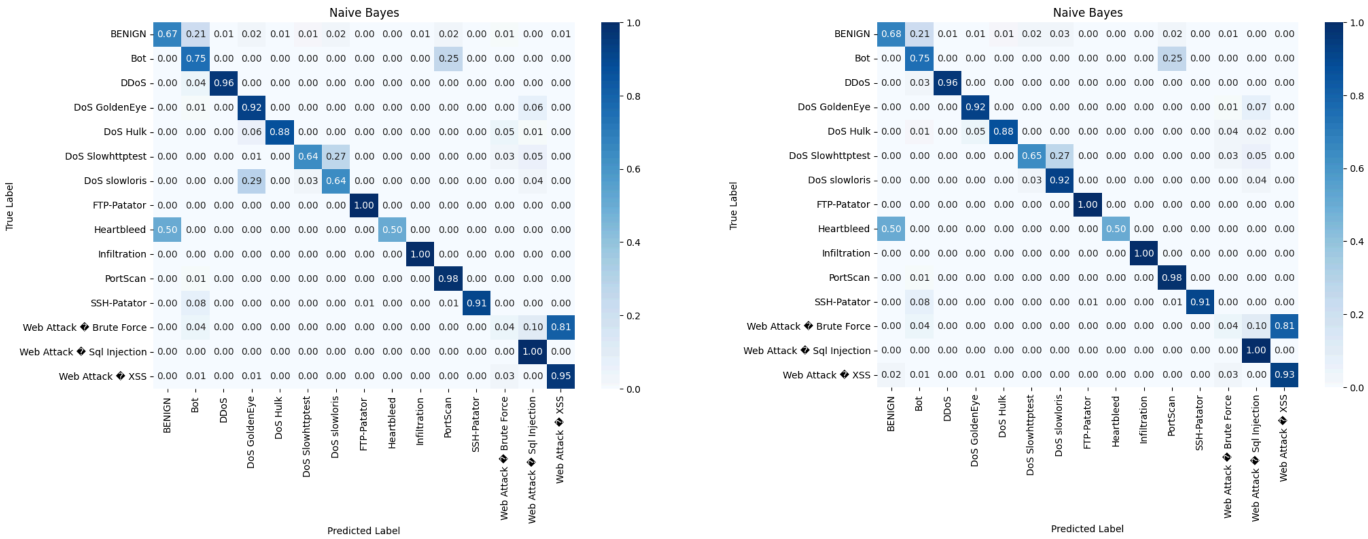


Figure 10 : Naive Bayes CM (Almost Balanced Strategy, Minority Favored Strategy)

The second Confusion Matrix indicates that transitioning to a minority-favored strategy doesn't lead to substantial improvements for the Naive Bayes model. Although there's been a slight improvement in classifying 'DoS Slowloris' attacks, the true positive rates for 'Bot' and

'Heartbleed' attacks have not changed. Additionally, the increase in detecting web attacks doesn't meet expectations, hinting that the model's fundamental probabilistic structure might not be as compatible with the resampling technique in comparison to more intricate models.

Convolutional Neural Network

In the first diagram for the Convolutional Neural Network (CNN) trained with an almost equal strategy, We see a clear contrast in performance. The CNN model exhibits a high level of accuracy when it comes to categorizing benign traffic, 'DDoS', 'DoS GoldenEye', 'DoS Hulk', 'FTP-Patator', 'SSH-Patator', and 'Portscan', with true positive rates between 0.9 and 1.0. The high scores show how well CNN extracts features for these attack types. Nonetheless, the model faces challenges when dealing with classes such as 'Heartbleed', 'Infiltration', and various web attack types like 'Brute Force', 'SQL Injection', and 'XSS', as well as 'Bot' attacks, giving them all a score of 0. Which shows that although the CNN is effective at identifying patterns that occur frequently, it struggles to identify attack classes that are less common or more subtle, possibly because they are not well represented in the training data as compared to other classes.

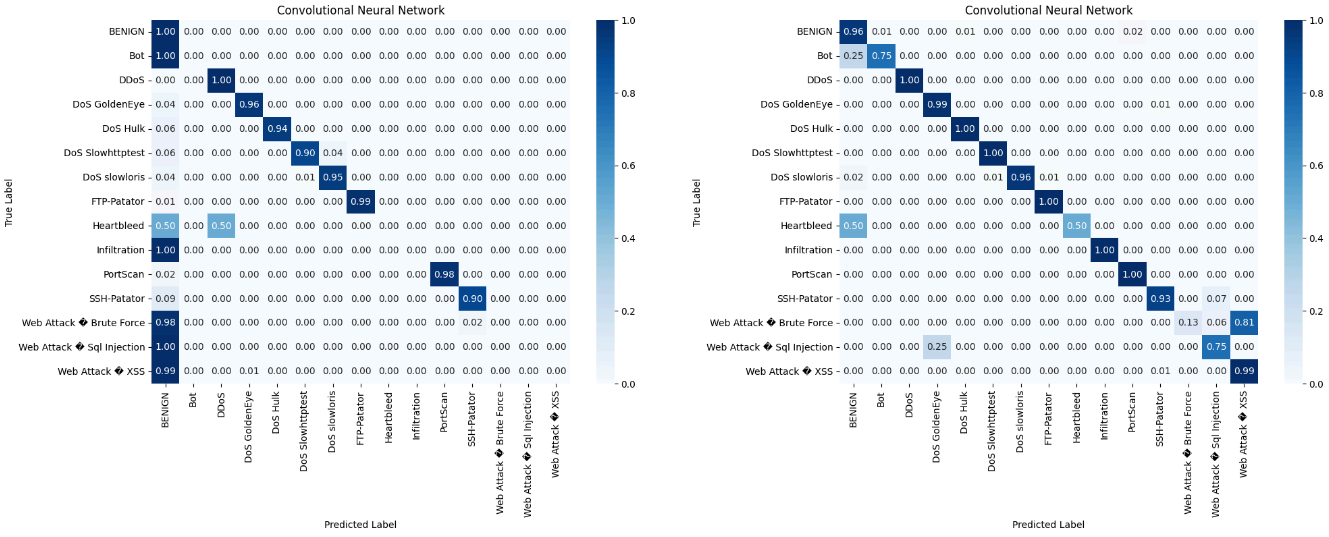
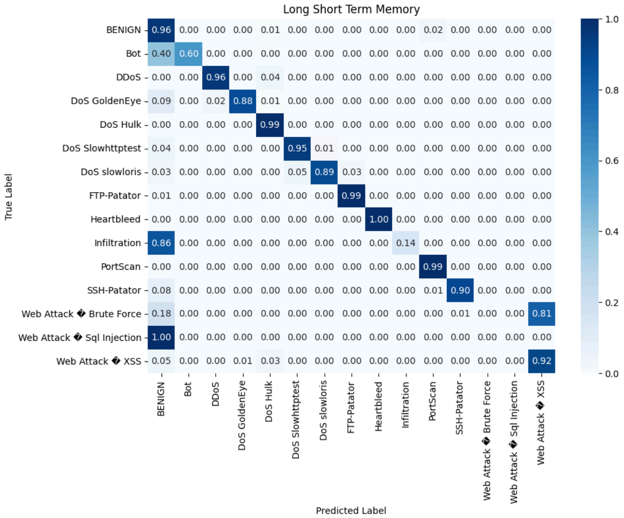


Figure 11 : Convolutional Neural Network CM (Almost Balanced Strategy, Minority Favored Strategy)

After transitioning to a strategy that favors the minority, the second Confusion Matrix shows significant improvement in all areas. 'Heartbleed', 'Infiltration', along with other types of web attacks are now demonstrating high true positive rates, indicating a significant improvement in the model's ability to detect them. An example is the detection rate of 'Web Attack - XSS' increasing from 0 to 0.99 and 'Web Attack – SQL Injection' rising from 0 to 0.75. An increase in detection of 'bot' attacks also rises, highlighting the effectiveness of the strategy.

Long Short Term Memory

At first, using an almost equal strategy, the LSTM model excels at detecting benign traffic and various types of DoS and DDoS attacks, achieving scores ranging from 0.88 to 1.00 for 'DoS GoldenEye', 'DoS Hulk', and 'DDoS'. The accurate detection rates for 'FTP-Patator' and 'SSH-Patator' are also notable, indicating the model's effectiveness in identifying these behaviors. Yet, the LSTM continues to face challenges in detecting 'Bot', 'Heartbleed', and various web attack types like 'Brute Force', 'SQL Injection', and 'XSS', resulting in significantly low or non-existent true positive rates.



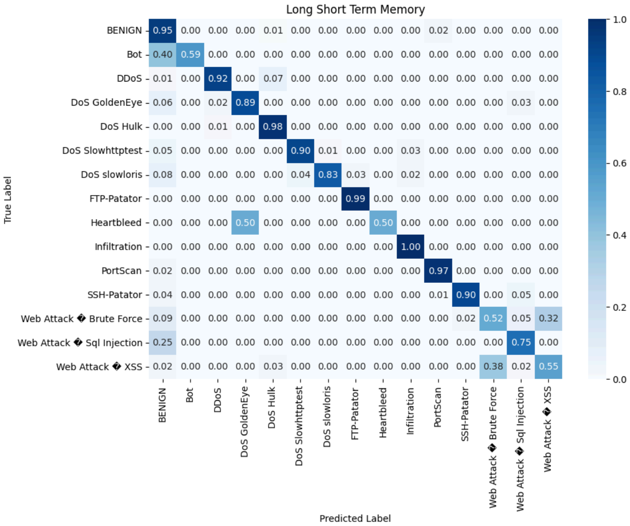


Figure 12 : Long Short Term Memory CM (Almost Balanced Strategy, Minority Favored Strategy)

Transitioning to a strategy that benefits minorities has led to significant improvements in various types of web attacks. For example, the detection of 'Web Attack: Brute Force' shows a marked enhancement, with the true positive rate rising from 0 to 0.52. Likewise, the detection of 'Web Attack - SQL Injection' rises from 0 to 0.75, while 'Web Attack - XSS' decreases from .92 to 0.55. The high true positive rates for other types of attacks indicate that the LSTM's focus on minority classes has improved its recognition of these less common attack signatures without compromising its performance on more prevalent types.

4.3 Overview of Model Performance (UNSW Dataset)

Following our examination of model performances on the CICIDS dataset, we are now focusing on the UNSW dataset, which presents a unique set of challenges for network intrusion detection systems. Similar machine learning and deep learning models were trained using the UNSW dataset.

Using the UNSW dataset, Decision Trees (DT) and Random Forests (RF) continued to show strong effectiveness in accurately classifying attack classes. Furthermore, positive results were also achieved with Support Vector Machines (SVM), which was not the case with CICIDS dataset. On the other hand, Logistic Regression (LR) and Naive Bayes (NB) struggled with certain attack categories such as 'Analysis', 'Backdoor', 'Shellcode', and 'Worms'. When compared, Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs) performed better than LR and NB, showing enhanced capability in distinguishing 'Shellcode' and 'Worms' attack classes.

.

4.3.1 Challenges with Minority Classes

Similar to the CICIDS dataset, certain models faced difficulties in identifying minority classes in the UNSW dataset. However, due to the UNSW dataset having a less significant imbalance, initial tests were carried out without sampling, enabling models to still perform well. Despite the slight imbalance, strategies were still modified to improve sensitivity towards minority classes when necessary.

4.3.2 Strategy Adjustments

The technique for handling class imbalance involved two steps: initially training models on the original, unsampled data, and then fine-tuning them with sampling techniques to account for any imbalances. In the case of SVM, balanced class weights were introduced during training to improve the model's ability to detect minority classes. This specific modification was also implemented in other models, However the models did not show considerable enhancements when class weights were applied.

In order to enhance the efficiency of CNNs and LSTMs, similar training techniques were implemented as those used for the CICIDS dataset. The use of Early Stopping and ReduceLROnPlateau methods were implemented. These methods improved the tuning process, resulting in increased model accuracy and effectiveness.

4.3.3 Model Evaluation and Insights

To thoroughly assess how the strategies have affected the model's performance, we provide Confusion Matrices for each model. This comparison study enables us to directly witness the alterations in model performance.

Decision Tree and Random Forest

The Decision Tree (DT) showed strong performance in accurately recognizing the majority of attack classes when trained on the UNSW dataset without using any sampling technique. In the Confusion Matrix provided, the Decision Tree (DT) showed high accuracy in detecting 'Generic', 'Normal', 'Reconnaissance', ‘Fuzzers’ and 'Exploits' attacks and minimal accuracy in detecting, other minority classes such as ‘Shellcode’ and ‘Worms’. Nevertheless, the DT encountered challenges in accurately categorizing minority classes like 'Analysis' and 'Backdoor', demonstrating limitations in its ability to distinguish between these less common attack types.

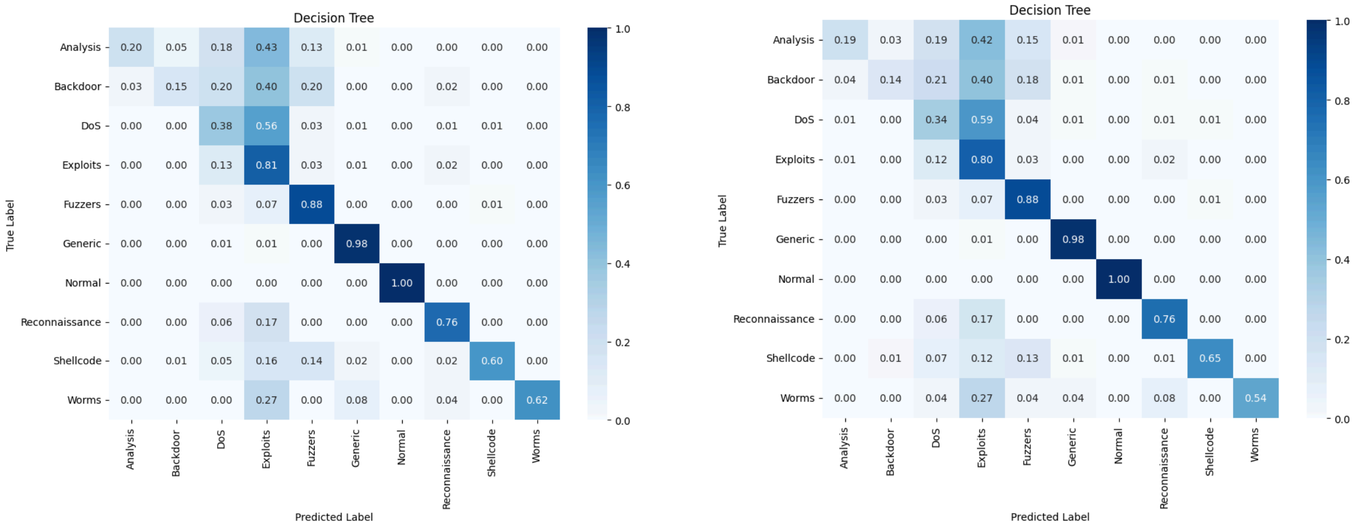


Figure 13 : Decision Tree CM for UNSW (Before Sampling, After Sampling)

Switching to a sampling technique to tackle class imbalance in the UNSW dataset did not result in a noticeable change in performance, as shown in the second confusion matrix. Moreover, the decision tree's true positive rates for 'Shellcode' and 'Worms' slightly decreased. The sampling strategy did not seem to have a significant impact on the decision tree's performance for the main classes as well. It continues to effectively identify majority class attacks but does not significantly enhance its ability to detect minority classes like 'Analysis', 'Backdoor', and other classes, which still have lower detection rates.

Contrarily, in the case of Random Forest without any sampling technique, the model displays similar performance to DT, especially for 'Exploits', 'Fuzzers', ‘Reconnaissance’ and 'Generic' types of attacks, achieving high true positive rates. Nevertheless, the model shows limitations in distinguishing minority classes such as 'Analysis', 'Backdoor', ‘DOS’, ‘Shellcode’ and ‘Worm’ attacks, with lower true positive rates for these types of attacks, indicating challenges in identifying these attack classes.

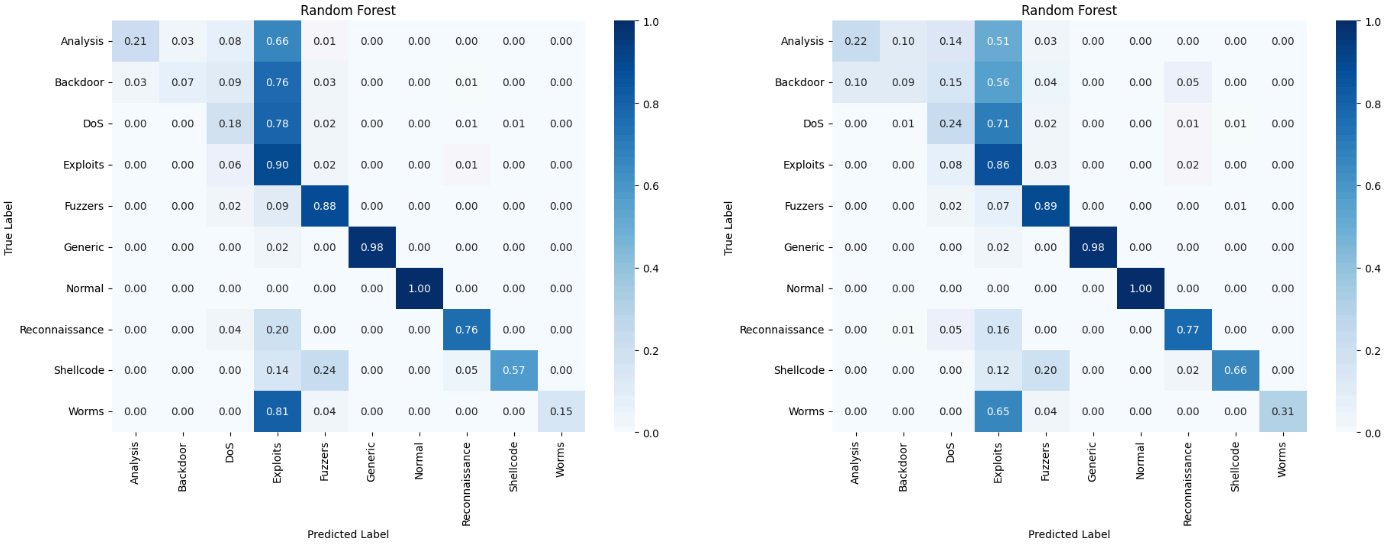


Figure 14 : Random Forest CM for UNSW (Before Sampling, After Sampling)

Following the implementation of the sampling strategy, the Random Forest model applied to the UNSW dataset shows subtle outcomes. The rate of correctly identifying 'Worm' attacks increased from 0.15 to 0.30 due to the sampling approach, indicating an enhancement in the model's ability to detect this type of attack. However, there was a drop in accuracy for minority attack categories like 'Analysis', 'DoS', 'Backdoor', and 'Shellcode' in contrast to the period before sampling. The accuracy for themajority ofy classes remained constant.

Logistic Regression

Logistic regression shows excellent accuracy for most classes, like 'Exploits', 'Fuzzers', 'Generic', and 'Normal', with significantly high true positives. Nonetheless, the model shows a poor performance in identifying minority classes such as 'Analysis', 'Backdoor', 'DoS', and 'Shellcode', with very low true positives and high false positives and negatives, suggesting a tendency to incorrectly classify these attacks as being absent.

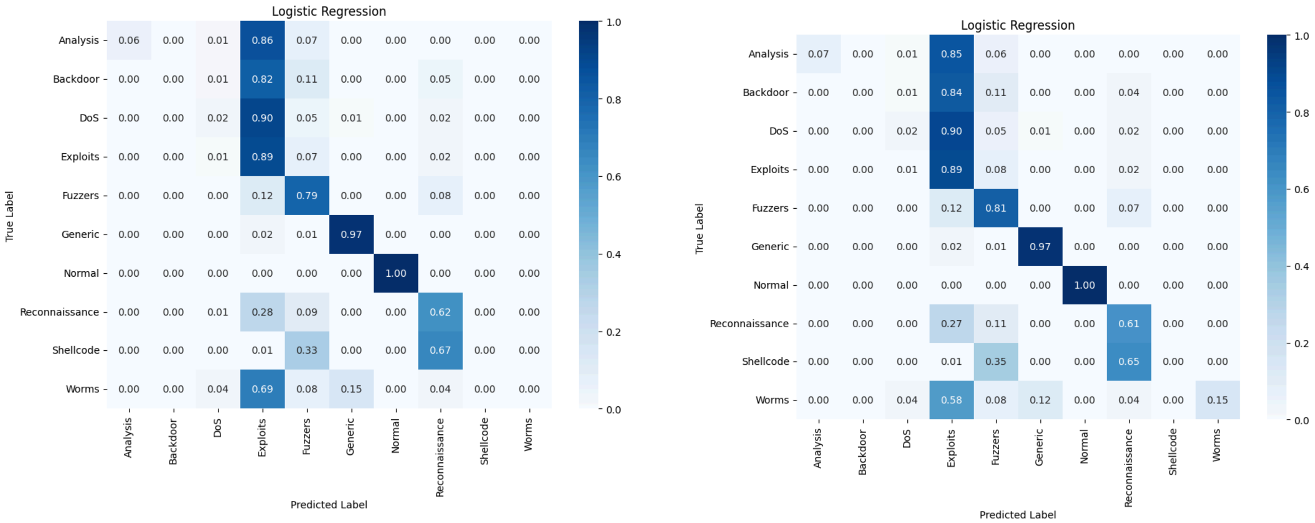


Figure 15 : Logistic Regression CM for UNSW (Before Sampling, After Sampling)

In terms of the sampling technique utilized, examination of the second Confusion Matrix demonstrates that the results remained the same for 'Analysis,' 'Backdoor,' 'DoS,' and 'Shellcode' , implying that the sampling method did not lead to improved classification of these categories by the model. A marked improvement is seen in the 'Worm' attack class, reflected by an increase in true positive rates. Moreover, despite the use of the sampling strategy, false positives and negatives for minority classes like 'Analysis', 'Backdoor', and 'DoS', ‘Shellcode’ and ‘Worms’ remain unchanged.

Support Vector Machine

The SVM showcases remarkable improvements in distinguishing 'Generic' and 'Normal' traffic, achieving nearly flawless true positive rates. The model consistently shows proficiency in identifying attacks within the 'DoS' and 'Worm' classes, achieving high detection rates. However, 'Reconnaissance' and 'Shellcode' classes, even though being smaller, showed good predictions. However, SVM faces challenges with low true positive rates in the 'Analysis' and 'Backdoor' attack classes, indicating potential for improvement. In general, SVM displayed remarkable performance, even without the need for sampling, reaching similar outcomes as DT.

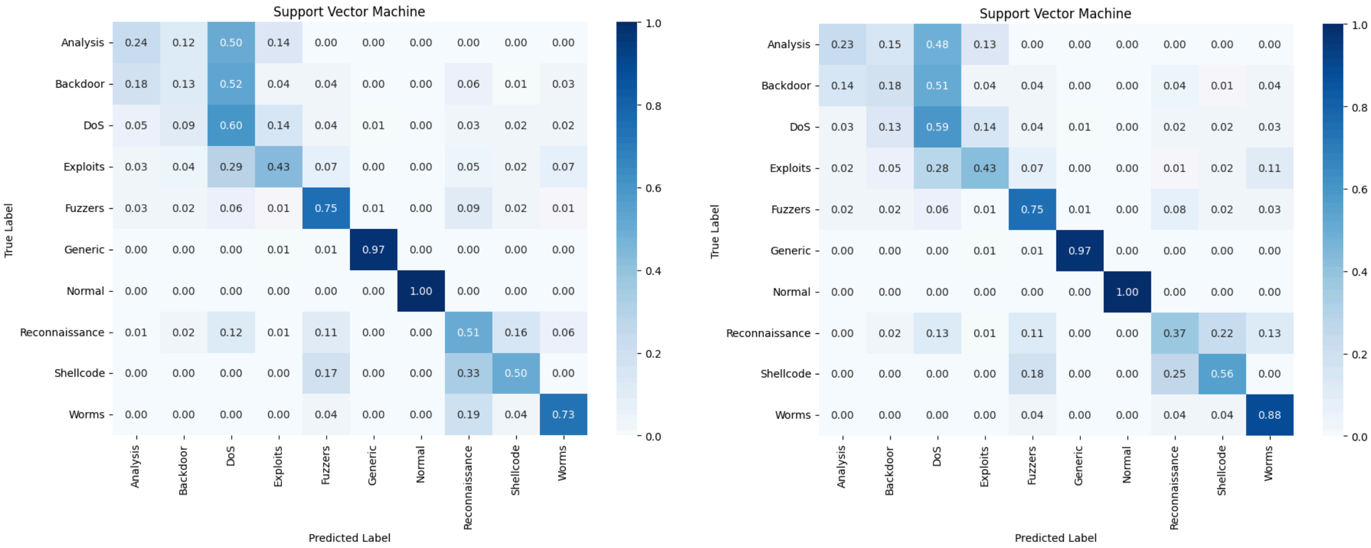
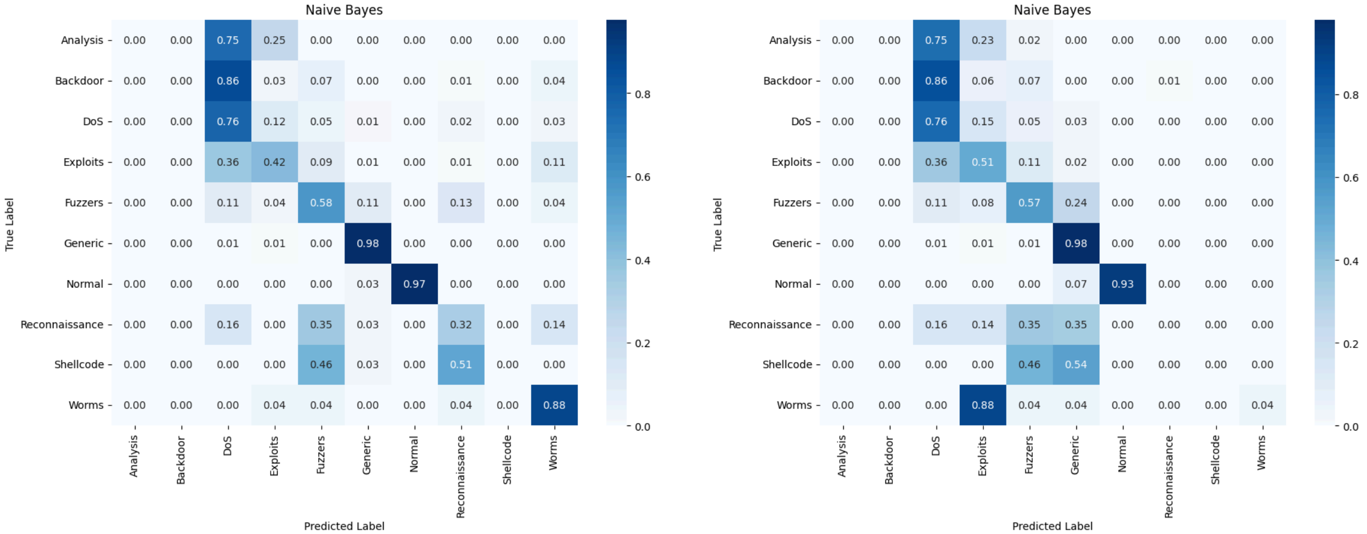


Figure 16 :Support Vector Machine CM for UNSW(Before Sampling, After Sampling)

When a sampling technique is used to handle class imbalance, the SVM model demonstrates significant enhancement in identifying 'Worms' attacks, with the true positive rate rising from 0.73 to 0.88. 'Shellcode' detection instances are also increasing, confirming the beneficial effect of sampling in balancing the performance of the model across different classes. Although there was a small decrease in the 'Analysis' and 'Backdoor' categories, the general trend after sampling indicates a more equal and enhanced model, especially for minority classes excluding Analysis, Backdoor, and Reconnaissance. Throughout both approaches, the SVM maintains its strong performance for 'Exploits', 'Fuzzers', 'Generic', and 'Normal' classes.

Naive Bayes

Naive Bayes showed strong classification capability for various attack types without using sampling. 'Generic' and 'Normal', as well as 'DOS' and 'Worm' classes, had very high true positive rates, almost reaching perfect scores. 'Worms' also achieved a high score of 0.88. Nevertheless, the model struggled with 'Analysis', 'Backdoor', and 'Shellcode' again, receiving 0



scores, which indicate these categories are harder to classify correctly. Aside from that, every other class received scores that were near .5.

Figure 16 : Naive Bayes CM for UNSW(Before Sampling, After Sampling)

After sampling, Naive Bayes performance declined overall, as indicated by the Confusion Matrix. The accuracy in predicting 'Worms' dropped significantly from 0.88 to 0.04 post-sampling, and similar declines were observed in other classes. True positive stayed consistent for 'Analysis', and 'Backdoor'. Furthermore, an noticeable rise in both false positives and false negatives can be observed in the top middle part and bottom middle part.

Convolutional Neural Networks

The CNN exhibited good performance, notably achieving perfect scores in detecting 'Generic' and 'Normal' attacks. Additionally, the 'Exploits', 'Fuzzers', and 'Reconnaissance' classes demonstrate high scores, suggesting successful understanding of these classes .However, minority categories such as 'Analysis', 'Backdoor', and 'DoS' demonstrate low true positive rates, underscoring the model's struggle to accurately detect these attacks. Moreover, the 'Worms' and

'Shellcode' categories exhibit moderate true positives, suggesting that the model is somewhat effective in detecting these types of attacks.

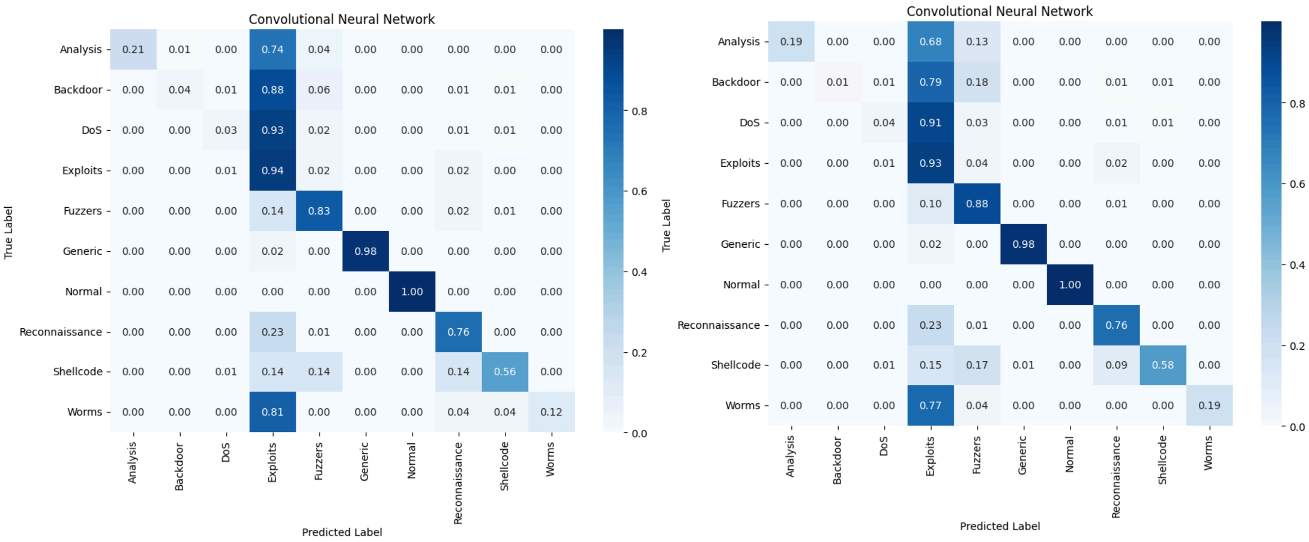


Figure 17 : Convolutional Neural Networks CM for UNSW(Before Sampling, After Sampling)

While post-sampling introduces a more balanced data representation, it does not uniformly improve model performance across all the classes and remains almost similar with minimal changes in the false positives, false negatives, and true positives.

Long Short Term Memory

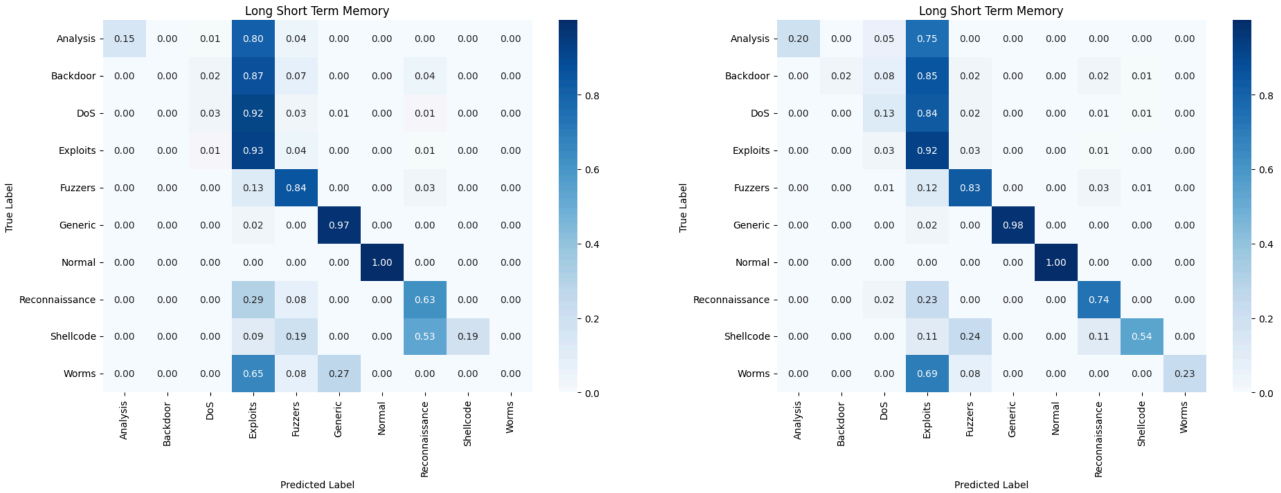


Figure 18 : Long Short Term Memory CM for UNSW(Before Sampling, After Sampling)

The model achieved a high level of accuracy in identifying most attack types, including 'Exploits', 'Fuzzers', 'Generic', 'Normal', and 'Reconnaissance', without employing any sampling strategy, with true positive rates close to 1.0. Detecting minority classes such as 'Analysis', 'Backdoor', 'DoS', and 'Shellcode' continues to pose challenges with low detection rates, highlighting the difficulty in differentiating these minority clases.

Using a sampling technique to target minority classes, the LSTM model showed enhanced performance for 'Worm' attacks, increasing from 0 to 0.23. Still, this method led to a rise in false positives and negatives, specifically in the 'Analysis', 'Backdoor', and 'DoS' groups, where the accuracy did not notably improve.

When looking at the LSTM and CNN performance on the UNSW dataset, it is clear that both models demonstrate a similar pattern of being effective at identifying the majority classes while struggling with the minority classes.

Chapter 5: Results

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	97.54	97.84	97.54	97.55
Support Vector Machine	93.55	96.18	93.55	94.53
Naive Bayes	71.30	96.82	71.30	81.02
Decision Tree	99.81	99.83	99.81	99.82
Random Forest	99.86	99.87	99.86	99.86
Convolutional Neural Network	99.04	98.88	99.04	98.95
Long Short Term Memory	96.37	97.27	96.37	96.66

Table 1 - Accuracy, Precision, Recall and F1 Score Achieved for CICIDS dataset with Almost Equal Strategy

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	85.84	95.98	85.84	89.91
Support Vector Machine	90.00	95.67	90.00	92.19
Naive Bayes	72.46	96.46	72.46	81.68
Decision Tree	99.84	99.85	99.84	99.84
Random Forest	99.86	99.87	99.86	99.86
Convolutional Neural Network	96.49	97.99	96.49	97.00
Long Short Term Memory	95.09	96.74	95.09	95.71

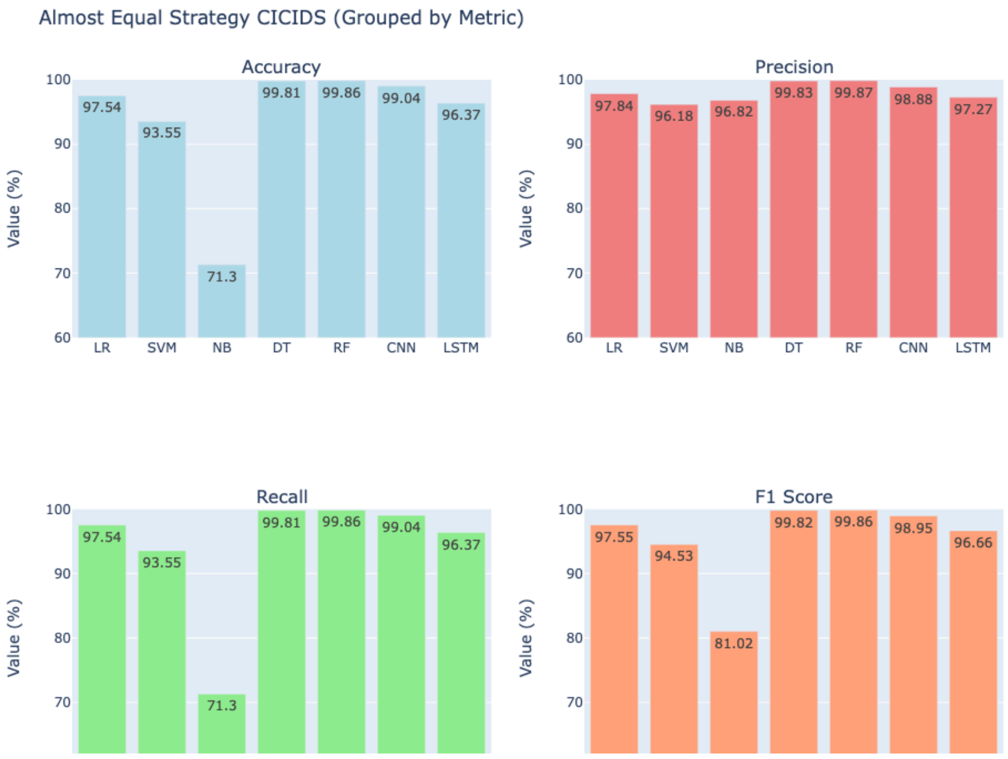
Table 2 - Accuracy, Precision, Recall and F1 Score Achieved for CICIDS dataset with Minority Favored Strategy

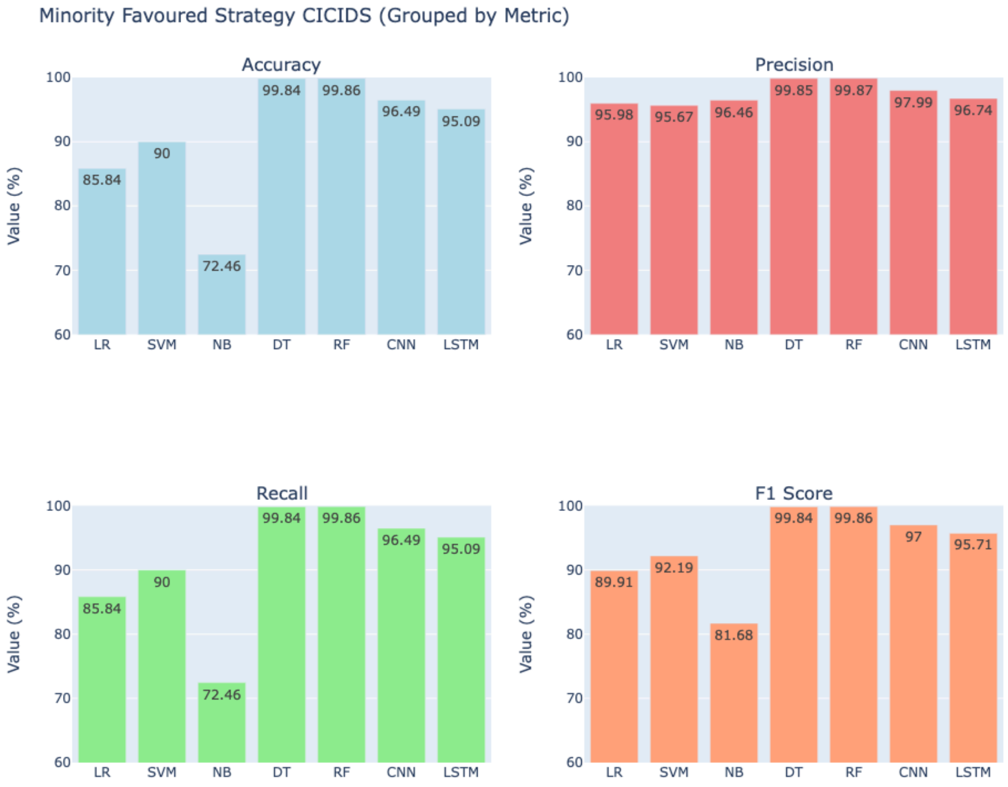
The performance metrics of the CICIDS dataset show a clear pattern in how different models perform with various sampling strategies in use. Decision Trees (DT) and Random Forest (RF) consistently show exceptional effectiveness, with nearly flawless accuracy, precision, recall, and F1 scores observed in both approaches. These findings highlight the strength of their ability to detect a wide range of attack types, even accurately classifying the more difficult minority classes.

The Almost Equal Strategy posed major difficulties in identifying minority classes for Logistic Regression (LR), Support Vector Machine (SVM), and Naive Bayes (NB) models, particularly in subcategories of web attacks like brute force, SQL injection, XSS, and other attacks such as Bot, Heartbleed, and Infiltration. However, the implementation of the Minority Favored Strategy, which increased the number of these minority classes, resulted in little reduction in model accuracy.

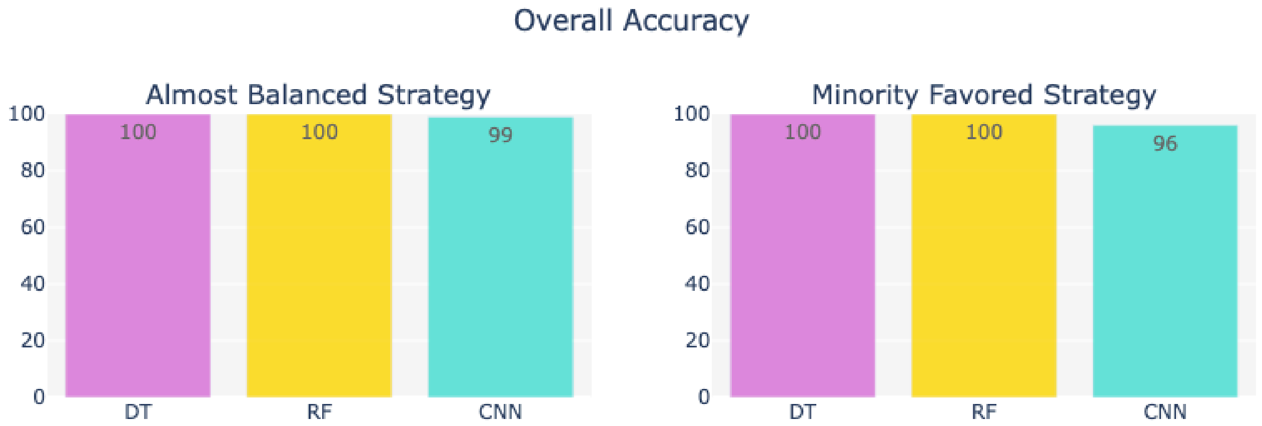
Initially, deep learning models like CNN and LSTM encountered similar challenges as LR, SVM, and NB did using the Almost Equal Strategy. The advanced models did not accurately predict the same minority classes, leading to increased metrics in Table 1. Nevertheless, there is a noticeable decline in the overall performance metrics in Table 2 following the implementation of the Minority Favored Strategy. This drop is due to the models starting to accurately recognize certain instances of minority classes, indicating a more fair and accurate evaluation of performance.

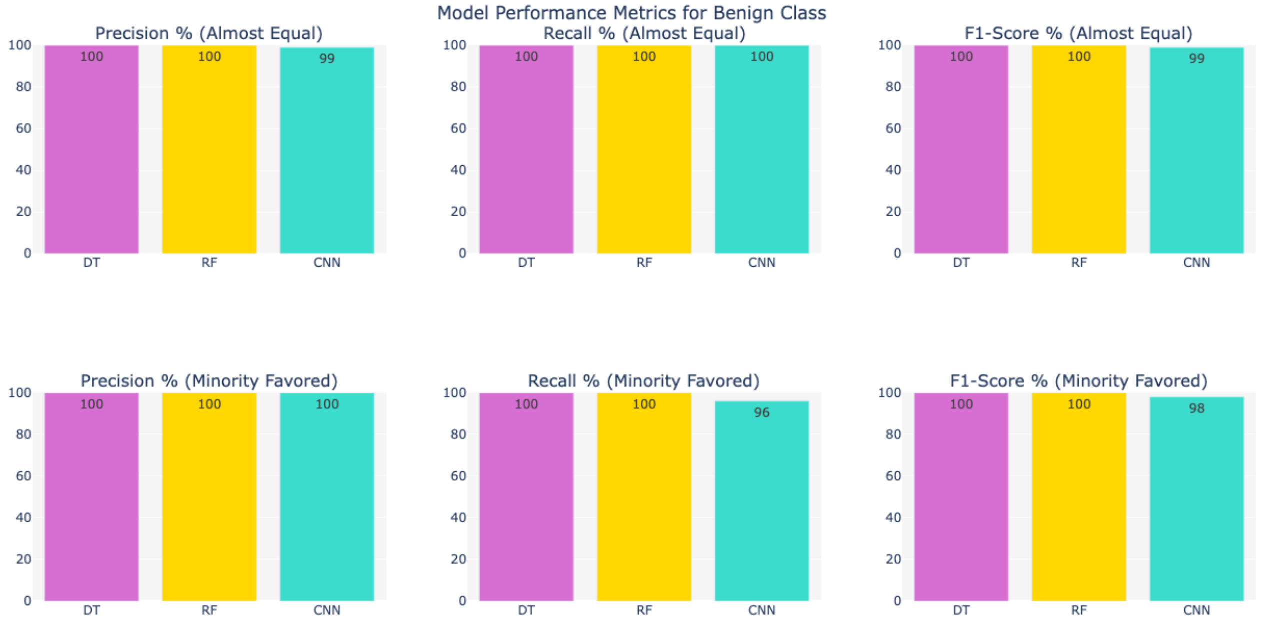
It was interesting that the difficulty in classifying minority classes mainly involved accurately predicting true positives. The first approach resulted in many incorrect predictions for classes like Web Attacks, Bot, Infiltration, and Heartbleed, a situation that the Minority Preferred Strategy has partly improved. This approach, which concentrated on increasing the number of samples from minority classes, enabled the models to distinguish these groups more effectively, albeit with a minor decrease in the overall performance measurements. The compromise shows a greater focus on minority class sensitivity, with a slight decrease in overall scores.

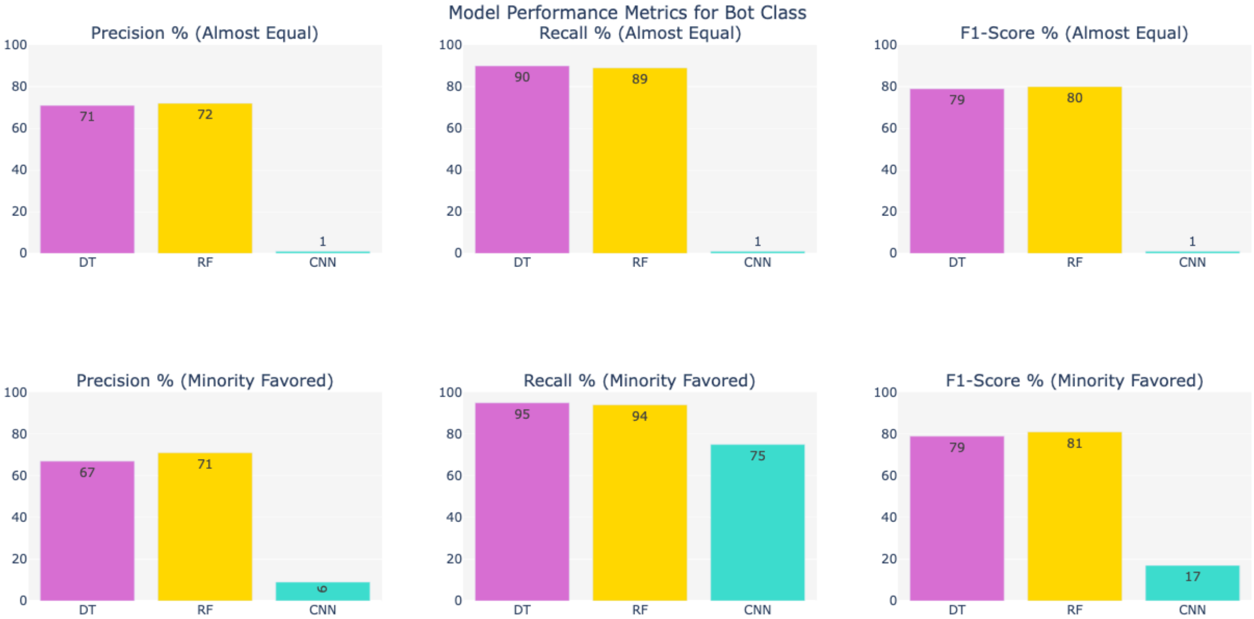

Almost Equal Strategy CICIDS (Grouped by Metric)
Accuracy
Value (%)
100
90
80
70
60
97.54
93.55
71.3
99.81
99.86
99.04
96.37
LR
SVM
NB
DT
RF
CNN
LSTM
Precision
Value (%)
97.84
96.18
96.82
99.83
99.87
98.88
97.27
LR
SVM
NB
DT
RF
CNN
LSTM
Recall
Value (%)
97.54
93.55
71.3
99.81
99.86
99.04
96.37
F1 Score
Value (%)
97.55
94.53
81.02
99.82
99.86
98.95
96.66

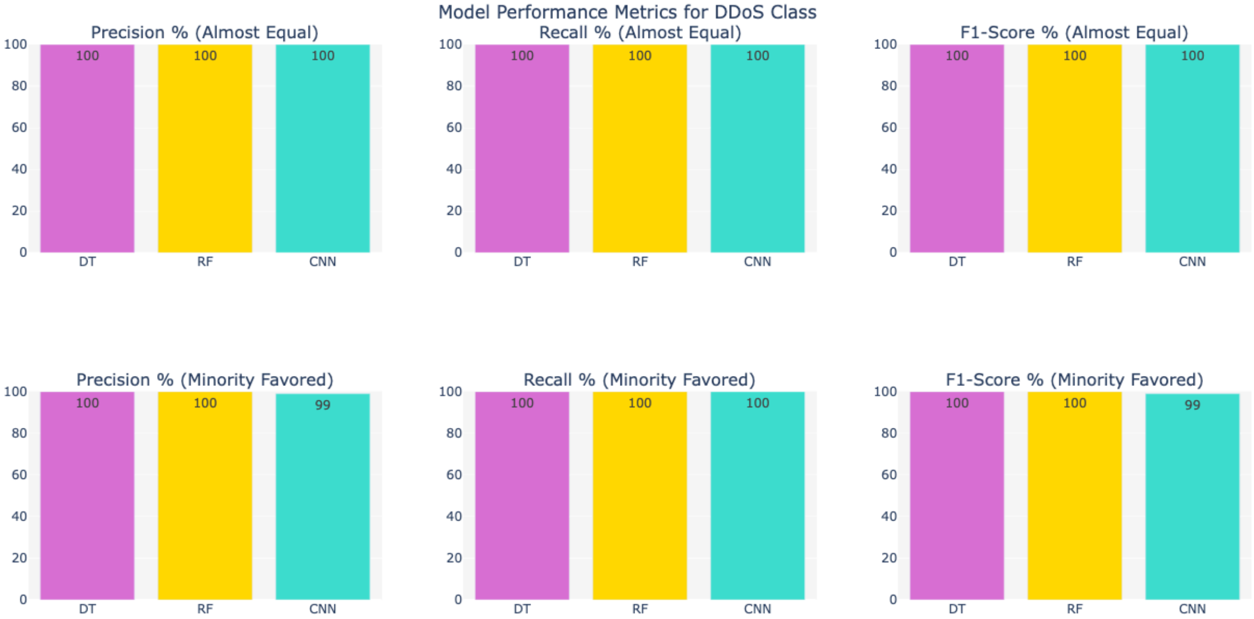

Minority Favoured Strategy CICIDS (Grouped by Metric)
Accuracy
Value (%)
100
90
80
70
60
85.84
90
72.46
99.84
99.86
96.49
95.09
LR
SVM
NB
DT
RF
CNN
LSTM
Precision
Value (%)
95.98
95.67
96.46
99.85
99.87
97.99
96.74
LR
SVM
NB
DT
RF
CNN
LSTM
Recall
Value (%)
85.84
90
72.46
99.84
99.86
96.49
95.09
LR
SVM
NB
DT
RF
CNN
LSTM
F1 Score
Value (%)
89.91
92.19
81.68
99.84
99.86
97
95.71
LR
SVM
NB
DT
RF
CNN
LSTM

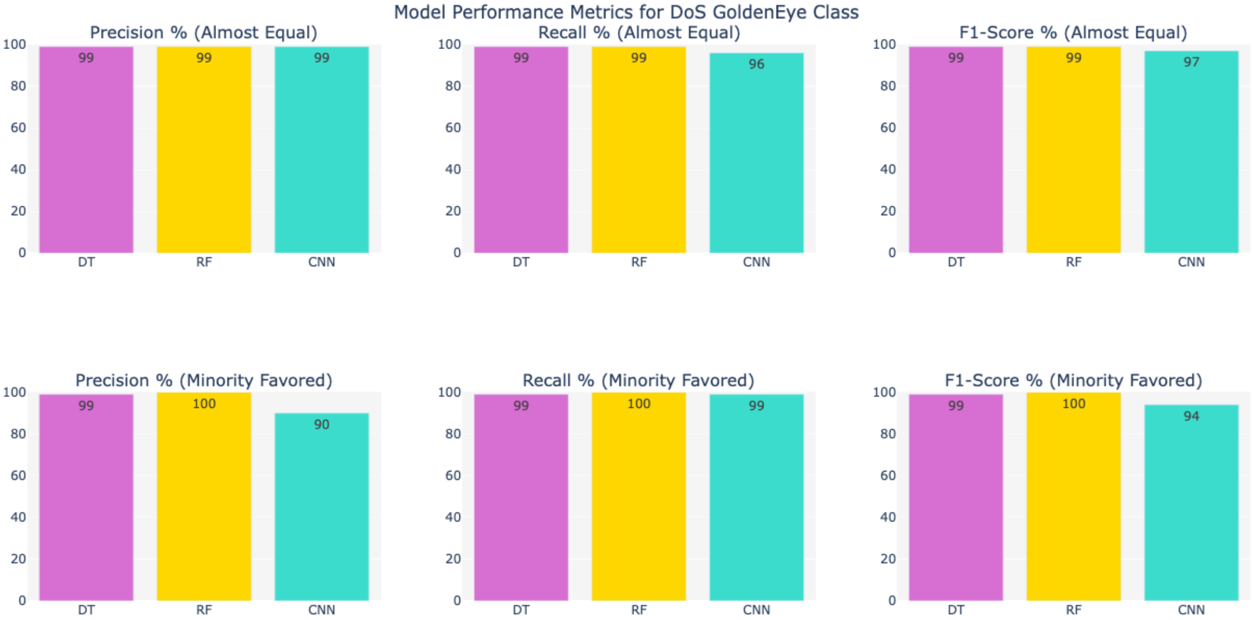
Class by Class Comparison for CICIDS based on top 3 models

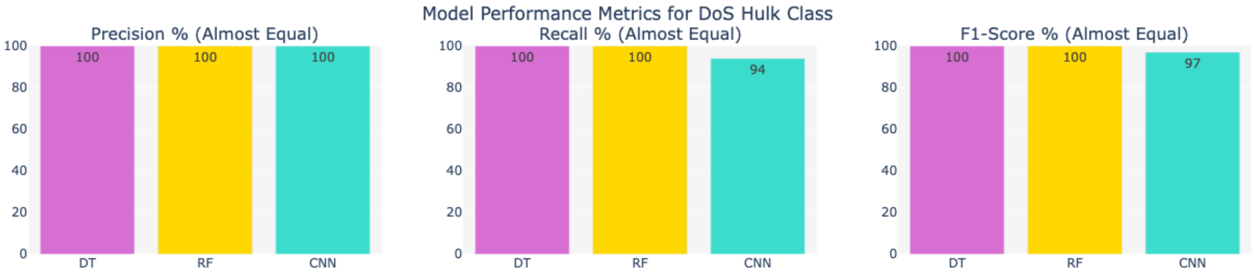





Model Performance Metrics for Bot Class
Precision % (Almost Equal)
Recall % (Almost Equal)
F1-Score % (Almost Equal)
Precision % (Minority Favored)
Recall % (Minority Favored)
F1-Score % (Minority Favored)
0
20
40
60
80
100
DT
RF
CNN
71
72
1
90
89
1
79
80
1
67
71
9
95
94
75
79
81
17


Model Performance Metrics for DDoS Class
Precision % (Almost Equal)
100
100
100
Recall % (Almost Equal)
100
100
100
F1-Score % (Almost Equal)
100
100
100
Precision % (Minority Favored)
100
100
99
Recall % (Minority Favored)
100
100
100
F1-Score % (Minority Favored)
100
100
99
DT
RF
CNN
0
20
40
60
80
100


Model Performance Metrics for DoS GoldenEye Class
Precision % (Almost Equal)
99
99
99
Recall % (Almost Equal)
99
99
96
F1-Score % (Almost Equal)
99
99
97
Precision % (Minority Favored)
99
100
90
Recall % (Minority Favored)
99
100
99
F1-Score % (Minority Favored)
99
100
94
DT
RF
CNN
0
20
40
60
80
100



Model Performance Metrics for DoS Hulk Class

Precision % (Almost Equal)

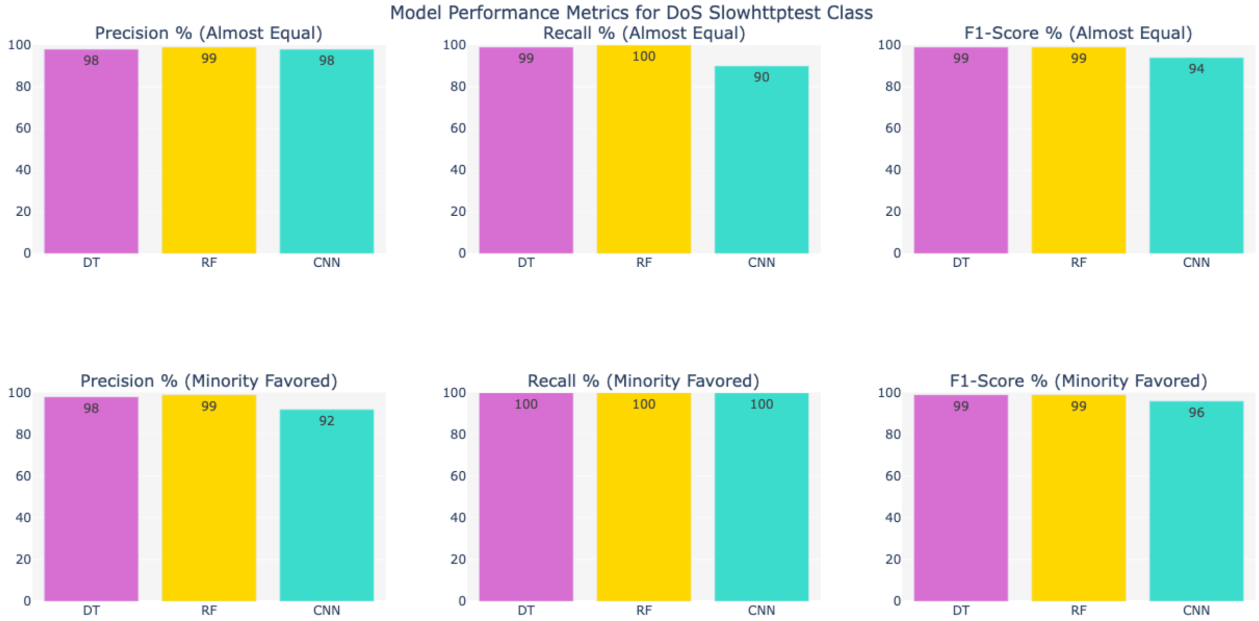
Recall % (Almost Equal)

F1-Score % (Almost Equal)

Precision % (Minority Favored)

Recall % (Minority Favored)

F1-Score % (Minority Favored)



Model Performance Metrics for DoS Slowhttptest Class

Precision % (Almost Equal)

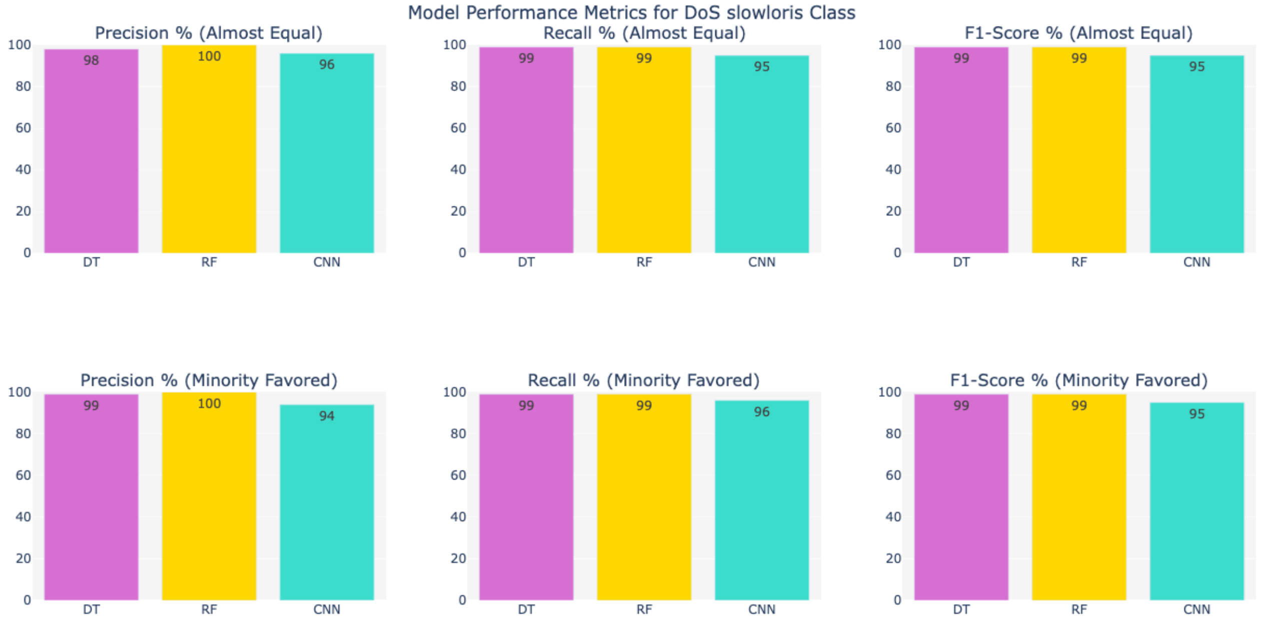
Recall % (Almost Equal)

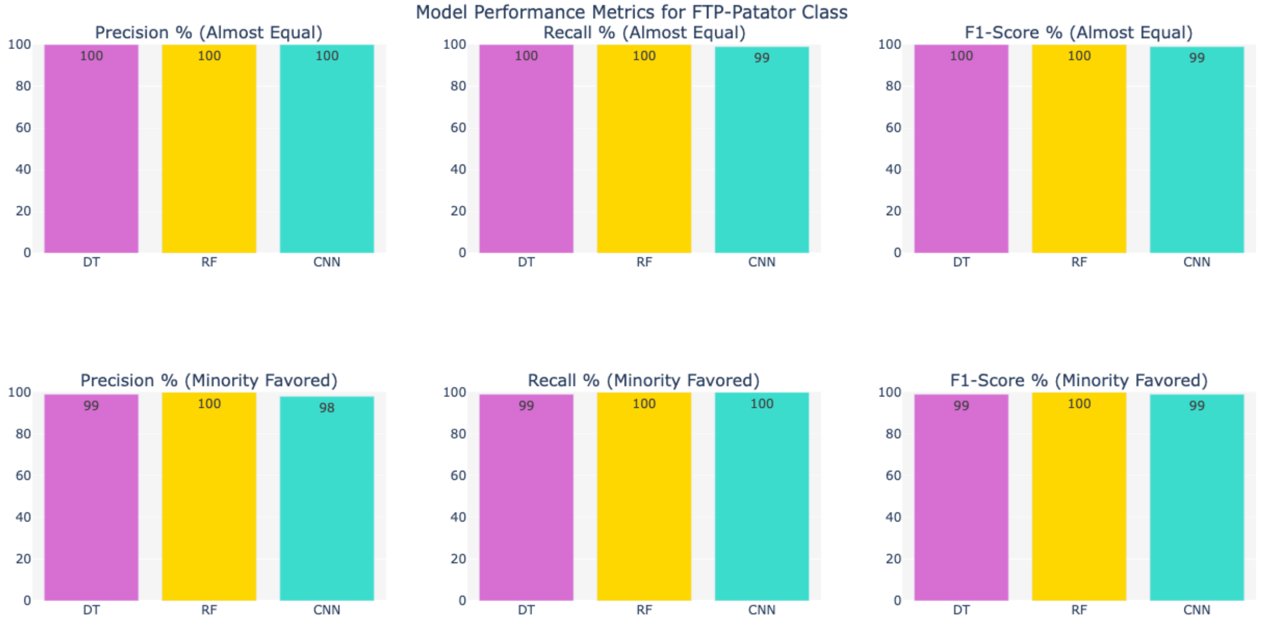
F1-Score % (Almost Equal)

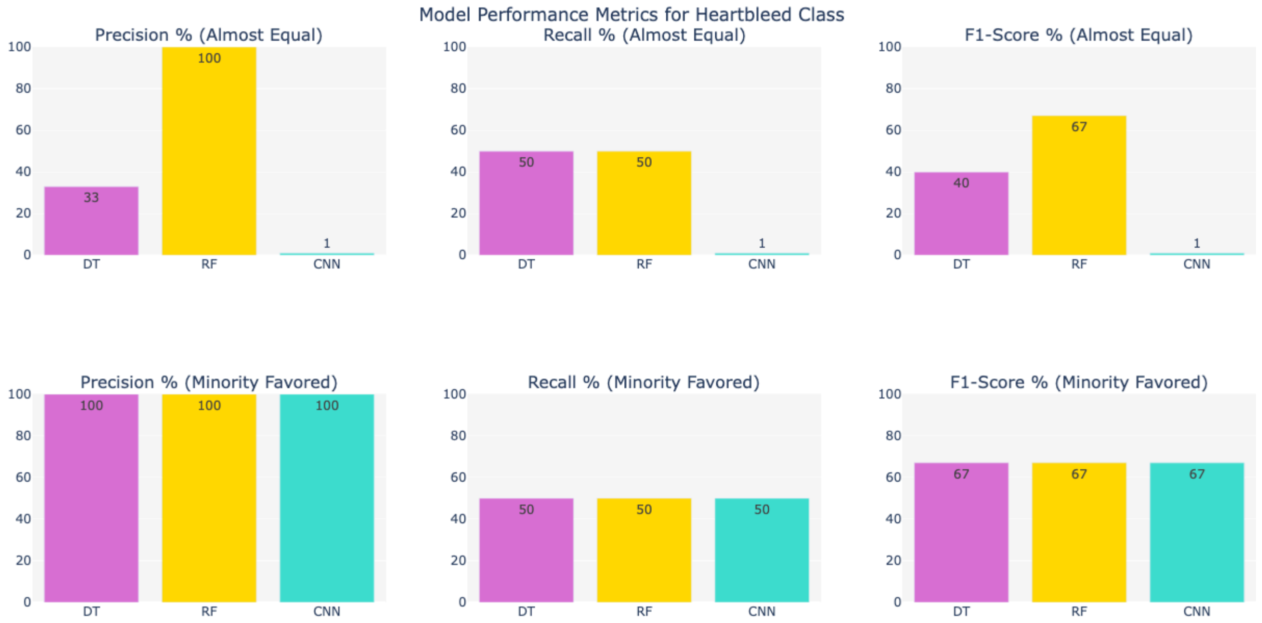
Precision % (Minority Favored)

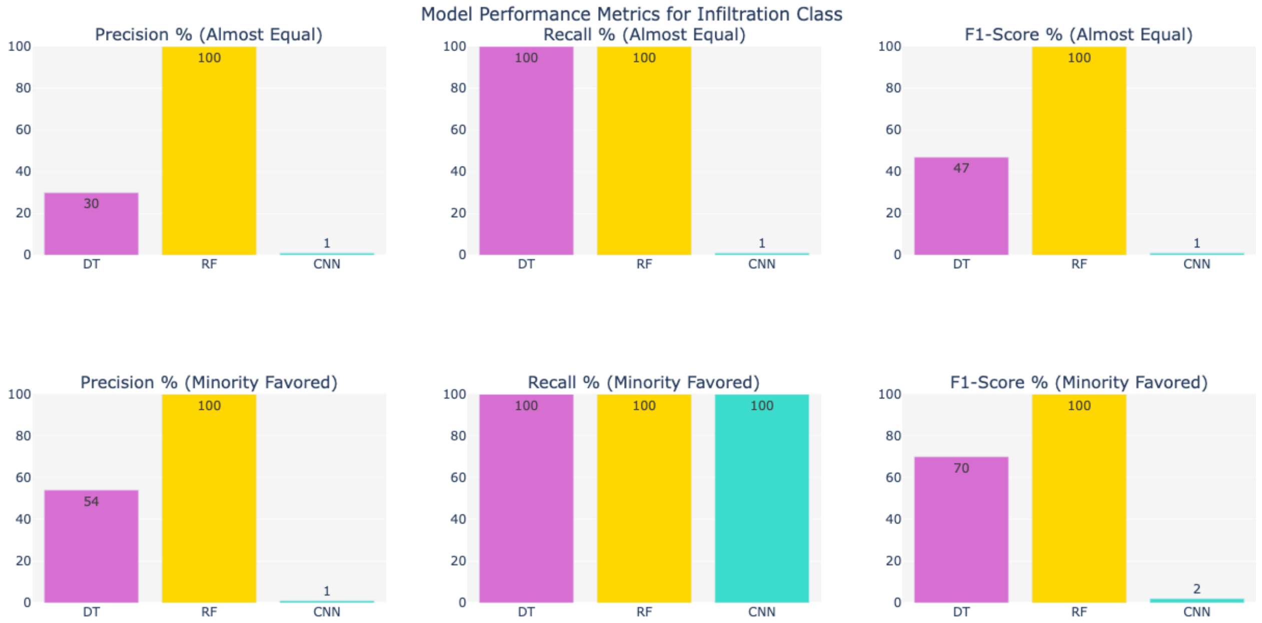
Recall % (Minority Favored)

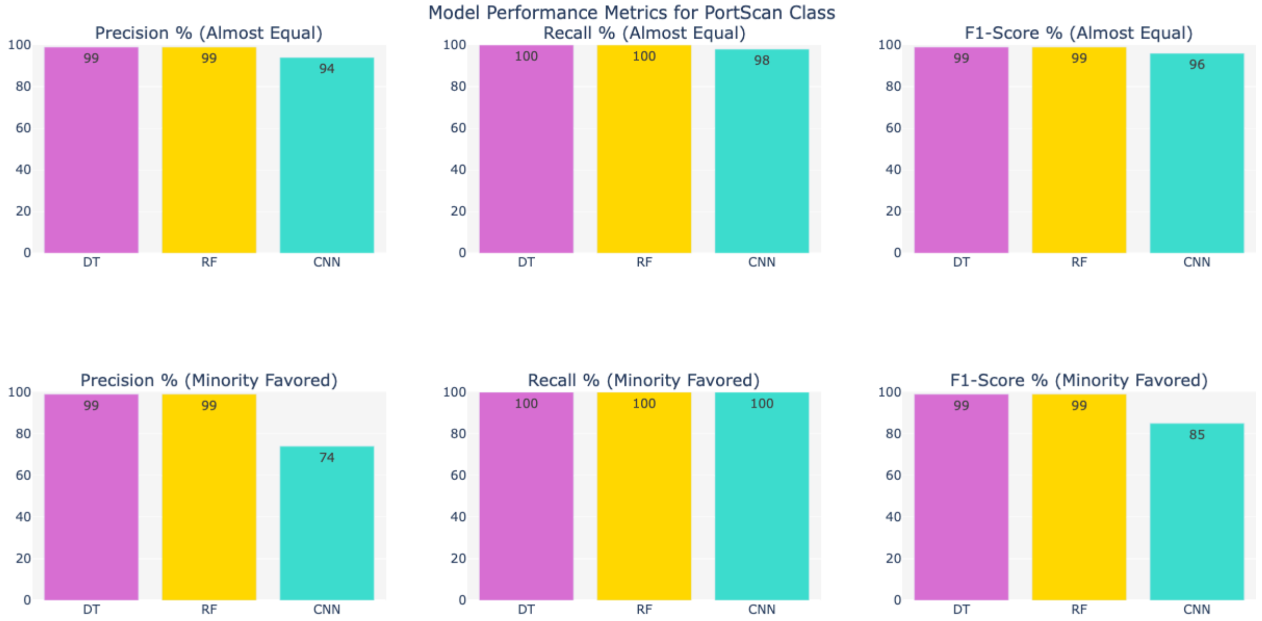
F1-Score % (Minority Favored)

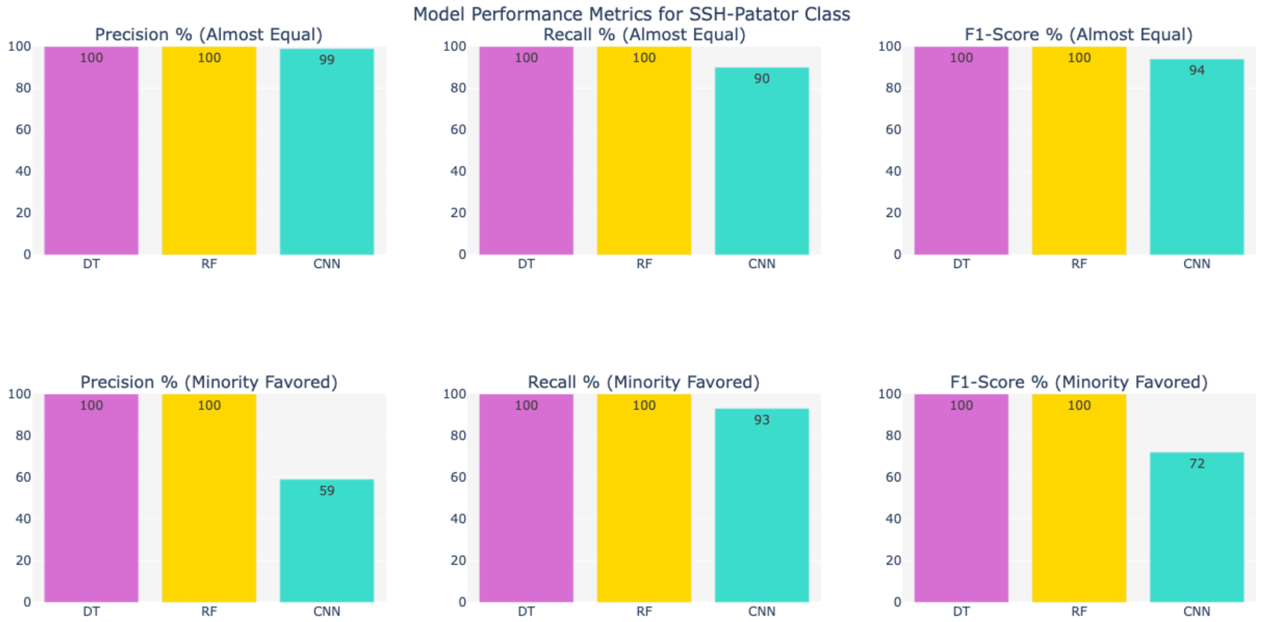

Model Performance Metrics for DoS slowloris Class
Precision % (Almost Equal)
Recall % (Almost Equal)
F1-Score % (Almost Equal)
Precision % (Minority Favored)
Recall % (Minority Favored)
F1-Score % (Minority Favored)
0
20
40
60
80
100
DT
RF
CNN
98
100
96
99
99
95
99
99
95
99
100
94
99
99
96
99
99
95

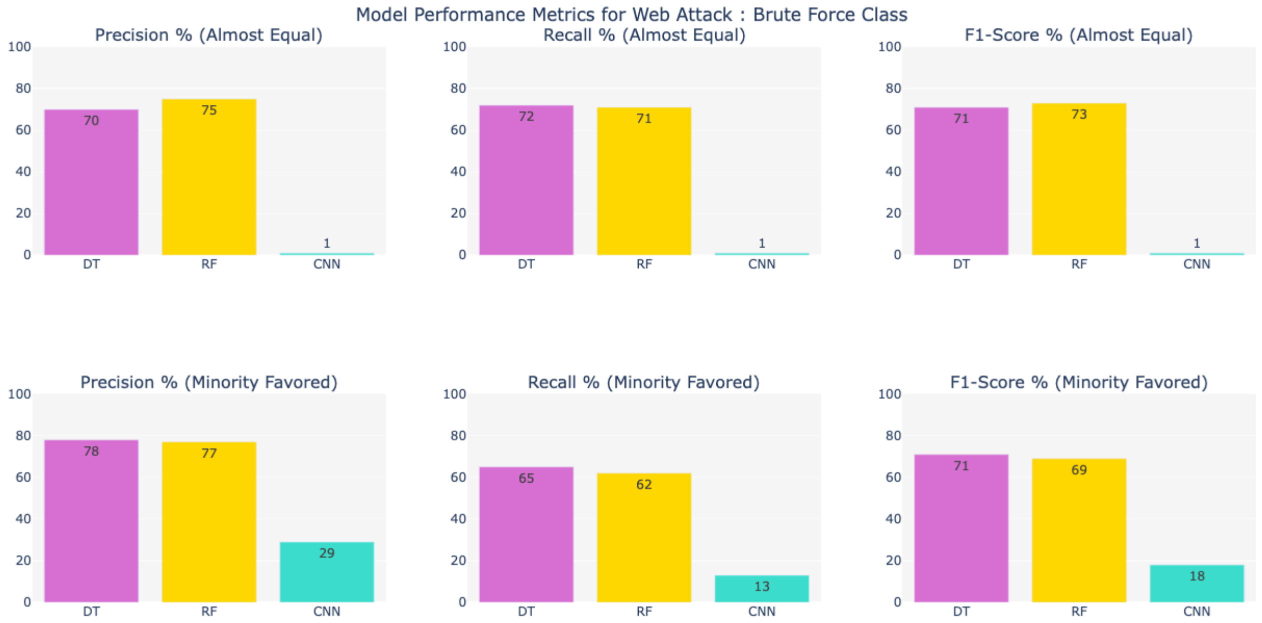

Model Performance Metrics for FTP-Patator Class
Precision % (Almost Equal)
Recall % (Almost Equal)
F1-Score % (Almost Equal)
Precision % (Minority Favored)
Recall % (Minority Favored)
F1-Score % (Minority Favored)
0
20
40
60
80
100
DT
RF
CNN
100
100
100
100
100
99
100
100
99
99
100
98
99
100
100
99
100
99

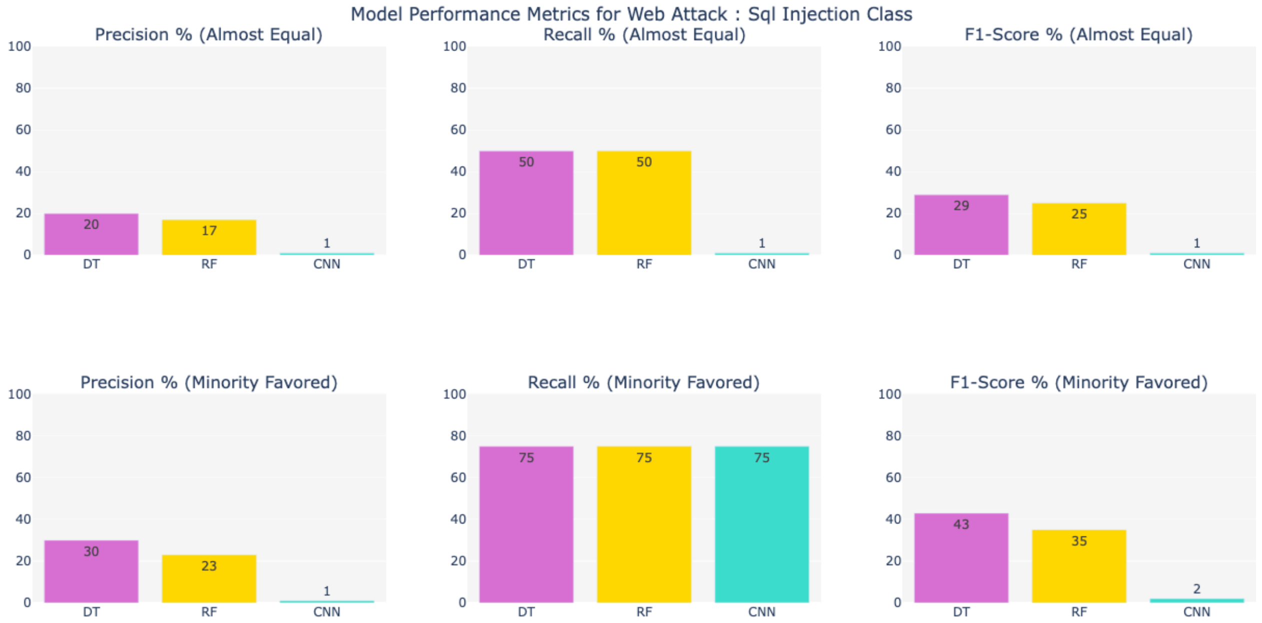

Model Performance Metrics for Heartbleed Class
Precision % (Almost Equal)
Recall % (Almost Equal)
F1-Score % (Almost Equal)
100
80
60
40
20
0
33
100
1
50
50
1
40
67
1
DT
RF
CNN
Precision % (Minority Favored)
Recall % (Minority Favored)
F1-Score % (Minority Favored)
100
100
100
50
50
50
67
67
67

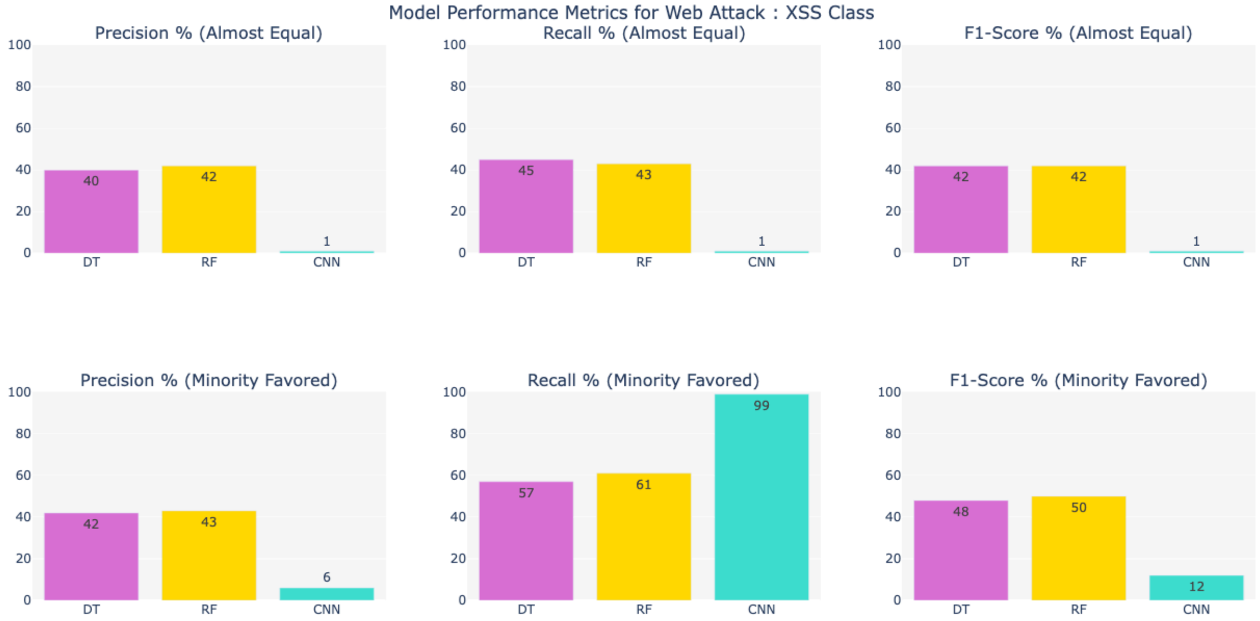

Model Performance Metrics for Infiltration Class
Precision % (Almost Equal)
Recall % (Almost Equal)
F1-Score % (Almost Equal)
100
80
60
40
20
0
30
100
1
100
100
1
47
100
1
DT
RF
CNN
Precision % (Minority Favored)
Recall % (Minority Favored)
F1-Score % (Minority Favored)
54
100
1
100
100
100
70
100
2


Model Performance Metrics for PortScan Class
Precision % (Almost Equal)
DT 99
RF 99
CNN 94
Recall % (Almost Equal)
DT 100
RF 100
CNN 98
F1-Score % (Almost Equal)
DT 99
RF 99
CNN 96
Precision % (Minority Favored)
DT 99
RF 99
CNN 74
Recall % (Minority Favored)
DT 100
RF 100
CNN 100
F1-Score % (Minority Favored)
DT 99
RF 99
CNN 85
0
20
40
60
80
100


Model Performance Metrics for SSH-Patator Class
Precision % (Almost Equal)
DT 100
RF 100
CNN 99
Recall % (Almost Equal)
DT 100
RF 100
CNN 90
F1-Score % (Almost Equal)
DT 100
RF 100
CNN 94
Precision % (Minority Favored)
DT 100
RF 100
CNN 59
Recall % (Minority Favored)
DT 100
RF 100
CNN 93
F1-Score % (Minority Favored)
DT 100
RF 100
CNN 72
0
20
40
60
80
100


Model Performance Metrics for Web Attack : Brute Force Class
Precision % (Almost Equal)
DT 70
RF 75
CNN 1
Recall % (Almost Equal)
DT 72
RF 71
CNN 1
F1-Score % (Almost Equal)
DT 71
RF 73
CNN 1
Precision % (Minority Favored)
DT 78
RF 77
CNN 29
Recall % (Minority Favored)
DT 65
RF 62
CNN 13
F1-Score % (Minority Favored)
DT 71
RF 69
CNN 18
0
20
40
60
80
100


Model Performance Metrics for Web Attack : Sql Injection Class
Precision % (Almost Equal)
DT 20
RF 17
CNN 1
Recall % (Almost Equal)
DT 50
RF 50
CNN 1
F1-Score % (Almost Equal)
DT 29
RF 25
CNN 1
Precision % (Minority Favored)
DT 30
RF 23
CNN 1
Recall % (Minority Favored)
DT 75
RF 75
CNN 75
F1-Score % (Minority Favored)
DT 43
RF 35
CNN 2
0
20
40
60
80
100



To sum up, the analysis of the CICIDS dataset shows the importance of sampling techniques in training models, especially for datasets with unequal class distribution. It points out that DT and RF models may not need big changes for handling class imbalance, but LR, SVM, NB, CNN, and LSTM models may benefit from customized approaches that boost their interaction with minority classes. The information obtained from this analysis should be used to improve future efforts in enhancing model training methods, so that all types of attacks, particularly those that are not as common, can be effectively identified by intrusion detection systems.

Results for UNSW

Table 3 states that the most models perform well, particularly Decision Tree and Random Forest which have accuracy scores exceeding 88%. High precision is consistent in all models, showing a strong performance in correctly predicting positive outcomes. Recall , which evaluates the capacity to locate all pertinent examples, aligns with accuracy, indicating that models are equally sensitive. The Decision Tree and Random Forest models are excelling in the F1 score, which combines precision and recall effectively.

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	84.81	83.32	84.81	81.93
Support Vector Machine	80.21	85.81	80.21	81.58
Naive Bayes	76.62	80.74	76.62	76.91
Decision Tree	88.05	87.92	88.05	87.68
Random Forest	88.07	87.73	88.07	86.92
Convolutional Neural Network	87.37	87.46	87.37	85.03
Long Short Term Memory	86.17	85.57	86.17	83.57

Table 3 - Accuracy, Precision, Recall and F1 Score Achieved for UNSW dataset before resampling

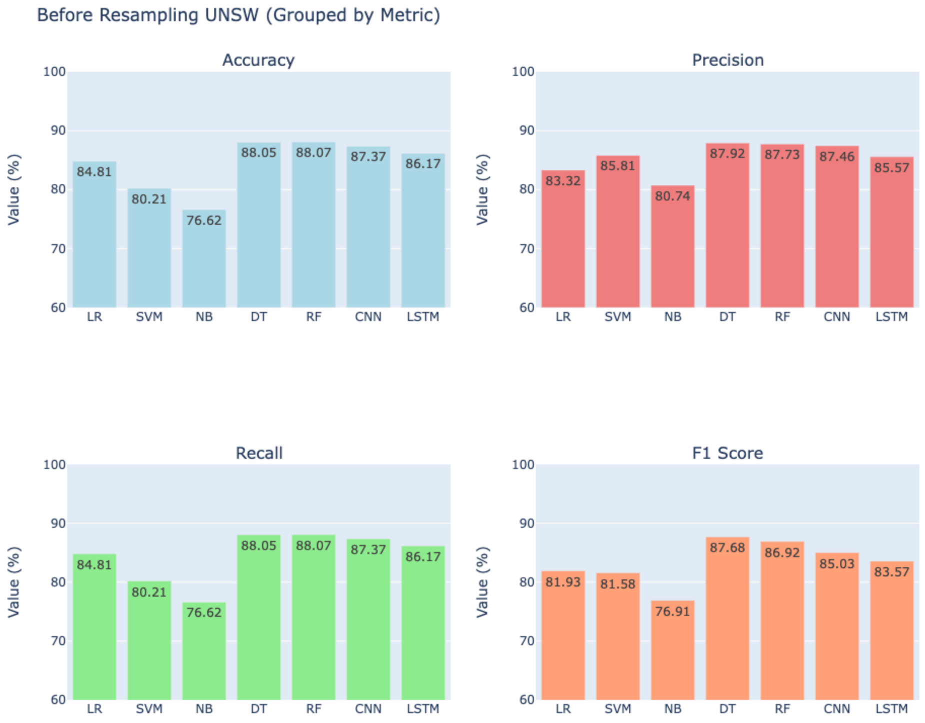
In table 4, after sampling, most models experience a small decrease in accuracy, except Random Forest, which demonstrates a slight improvement. LSTM and CNN have seen an improvement in accuracy, suggesting that they are better at predicting true positives post-sampling. Recall remains reliable in terms of precision, with only slight variations that do not deviate significantly from the original data. The F1 metrics show similarity, with certain models such as the Long Short-Term Memory network demonstrating enhancement post sampling, indicating an improved equilibrium of precision and recall.

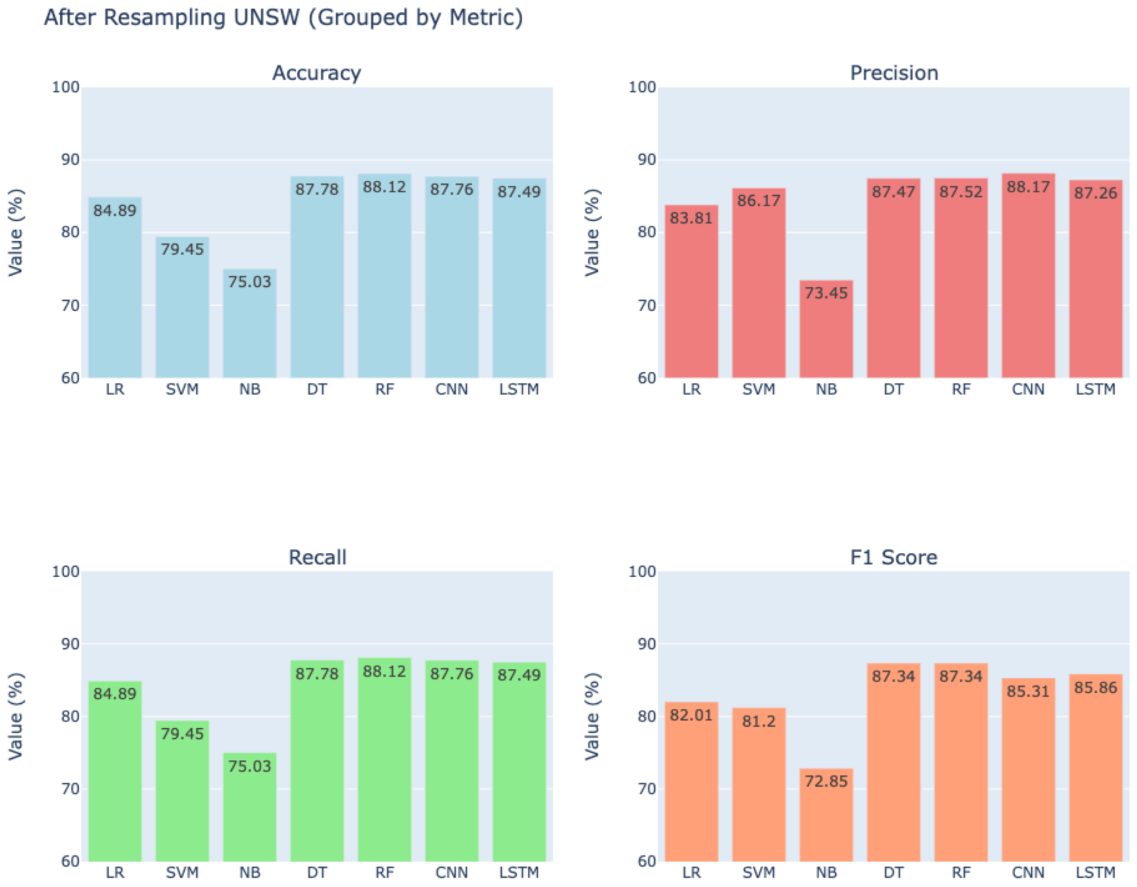
Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	84.89	83.81	84.89	82.01
Support Vector Machine	79.45	86.17	79.45	81.20
Naive Bayes	75.03	73.45	75.03	72.85

Decision Tree	87.78	87.47	87.78	87.34
Random Forest	88.12	87.52	88.12	87.34
Convolutional Neural Network	87.76	88.17	87.76	85.31
Long Short Term Memory	87.49	87.26	87.49	85.86

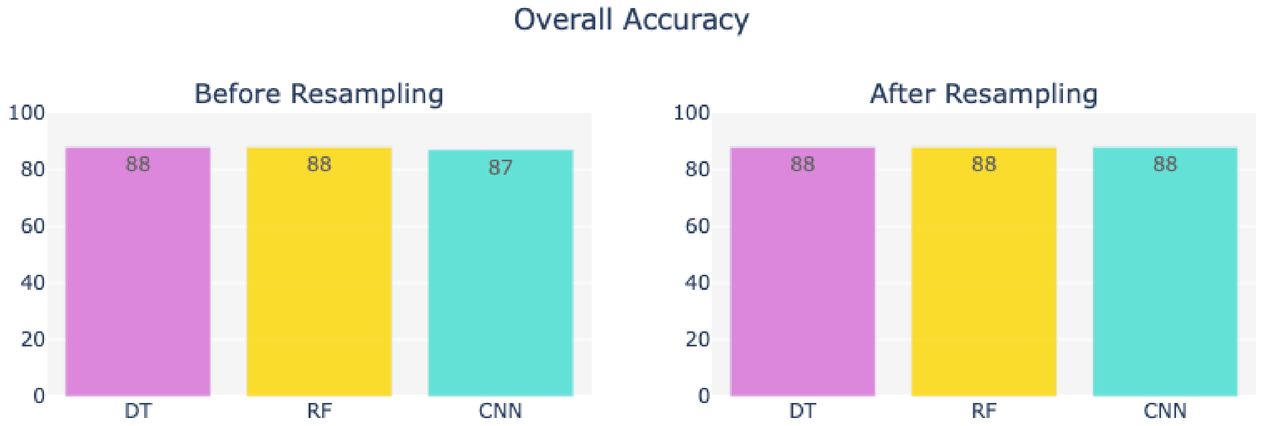
Table 4 - Accuracy, Precision, Recall and F1 Score Achieved for UNSW dataset after resampling

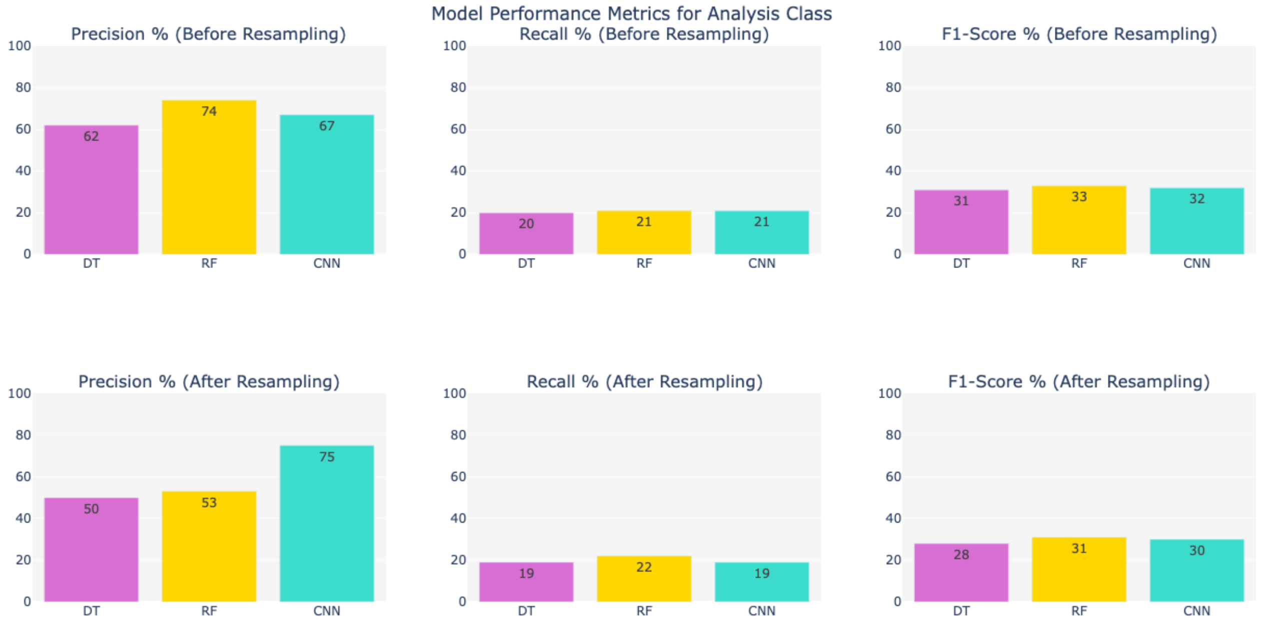
Despite the overall metrics being somewhat better without sampling, a closer examination of the confusion matrices revealed that the minority classes such as ‘Analysis’, ‘Backdoor’, ‘DoS’, ‘Shellcode’ and ‘Worm’ were frequently mislabeled or overlooked in each model. By utilizing sampling, there was enhanced detection of minority classes same as happened in CICIDS, explaining the small decrease in overall scores in exchange for better performance in minority classes. Sampling, while not making a huge difference in performance, did have a positive impact on identifying minority classes, which is crucial for a fair and balanced predictive model, especially in situations where misclassification costs can be significant.

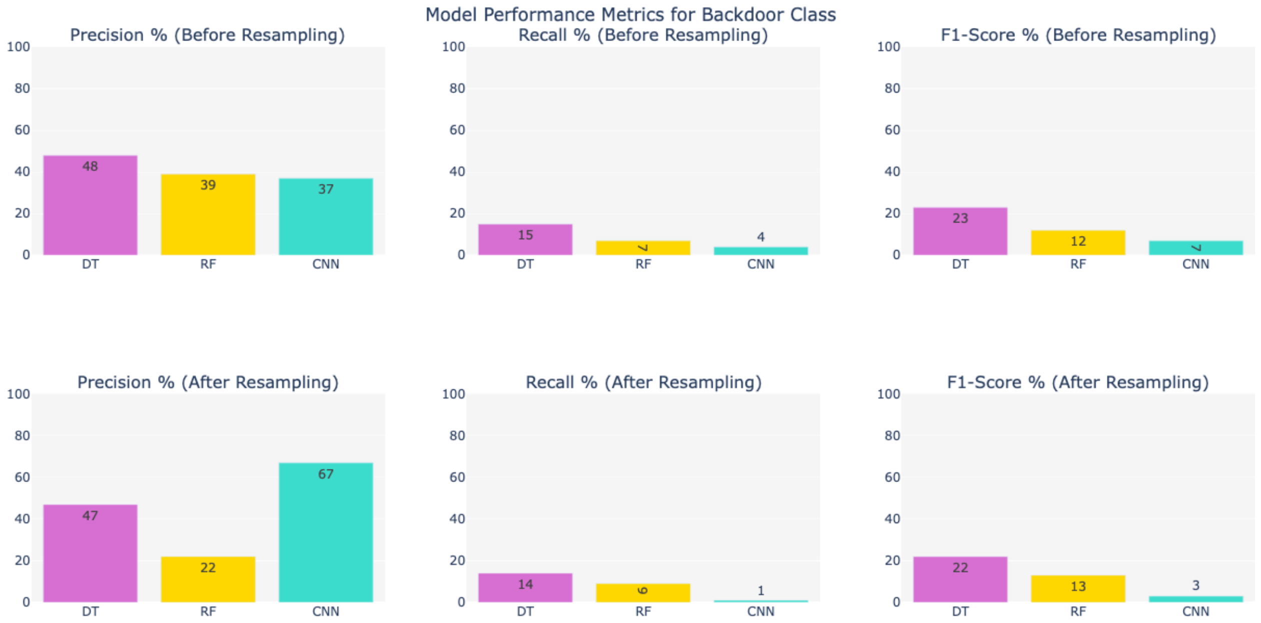


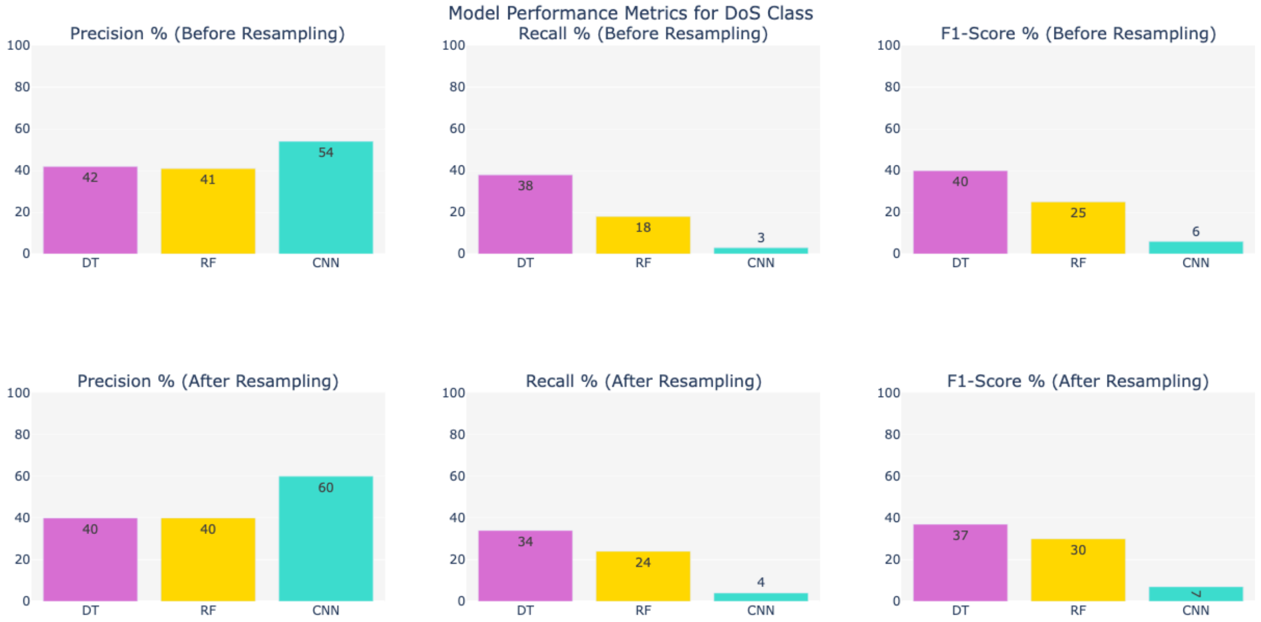

After Resampling UNSW (Grouped by Metric)
Accuracy
Value (%)
100
90
80
70
60
84.89
79.45
75.03
87.78
88.12
87.76
87.49
LR
SVM
NB
DT
RF
CNN
LSTM
Precision
Value (%)
100
90
80
70
60
83.81
86.17
73.45
87.47
87.52
88.17
87.26
LR
SVM
NB
DT
RF
CNN
LSTM
Recall
Value (%)
100
90
80
70
60
84.89
79.45
75.03
87.78
88.12
87.76
87.49
LR
SVM
NB
DT
RF
CNN
LSTM
F1 Score
Value (%)
100
90
80
70
60
82.01
81.2
72.85
87.34
87.34
85.31
85.86
LR
SVM
NB
DT
RF
CNN
LSTM

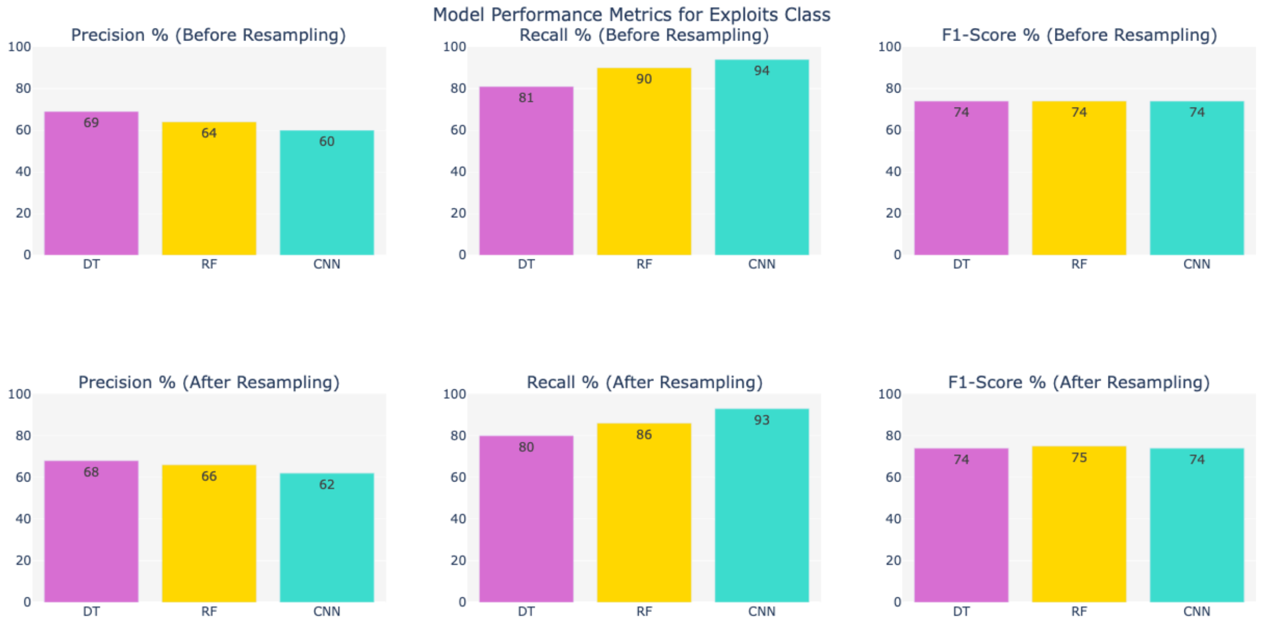
Class by Class Comparison for UNSW based on top 3 models

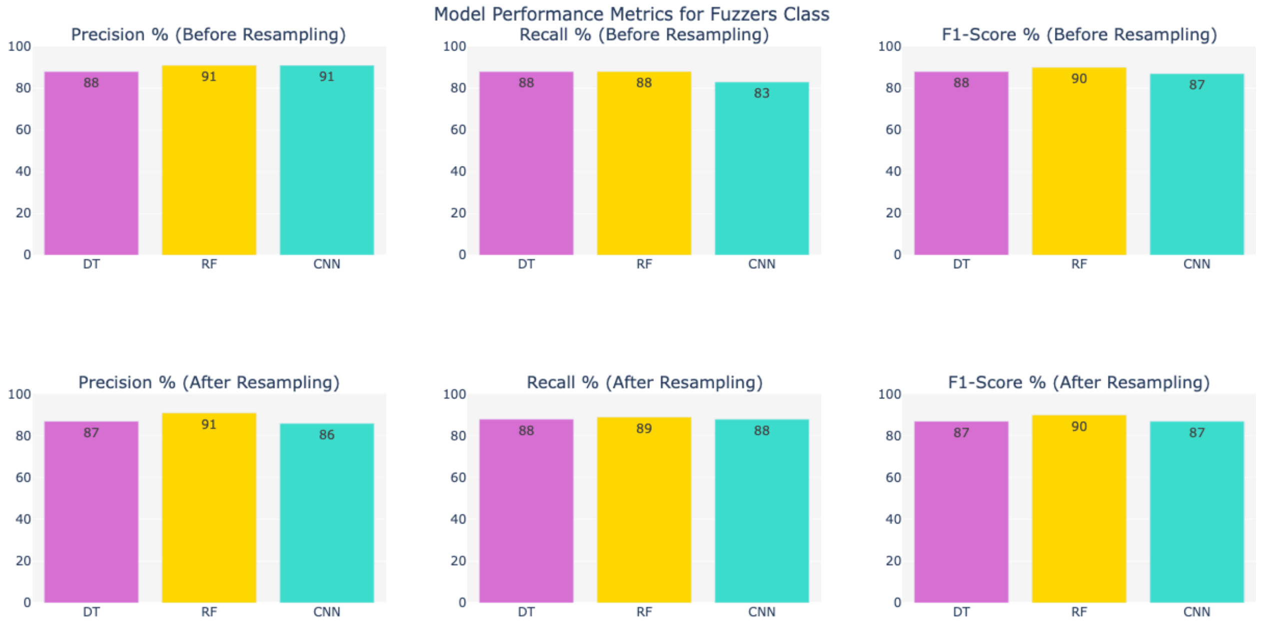


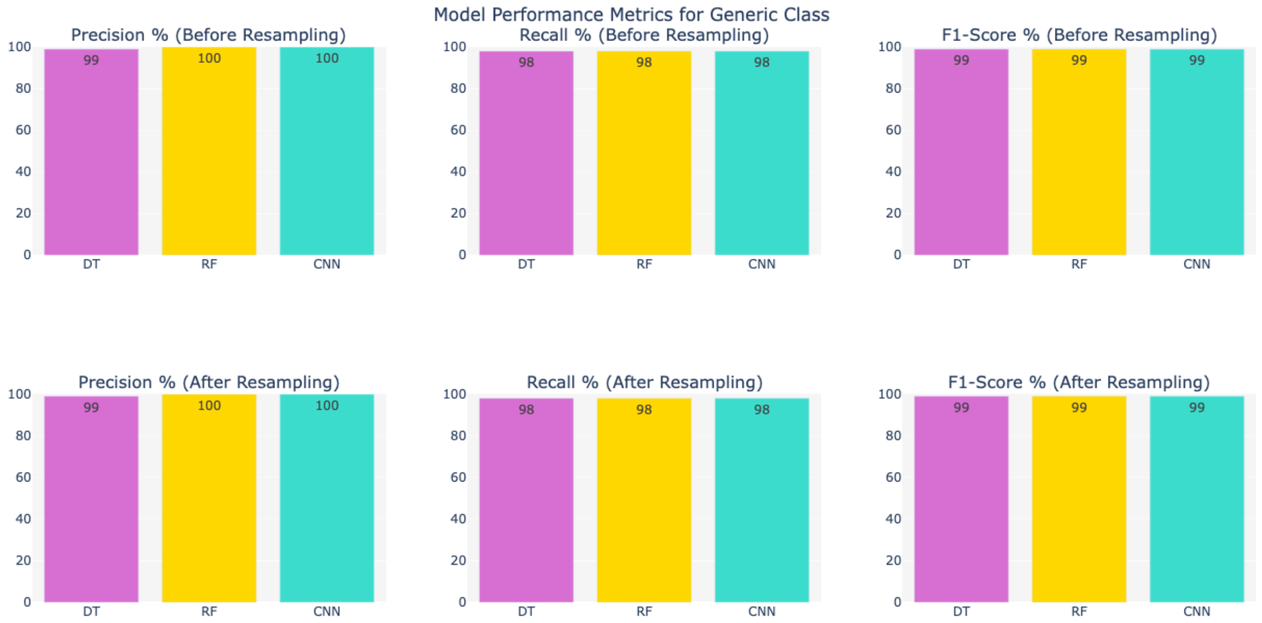


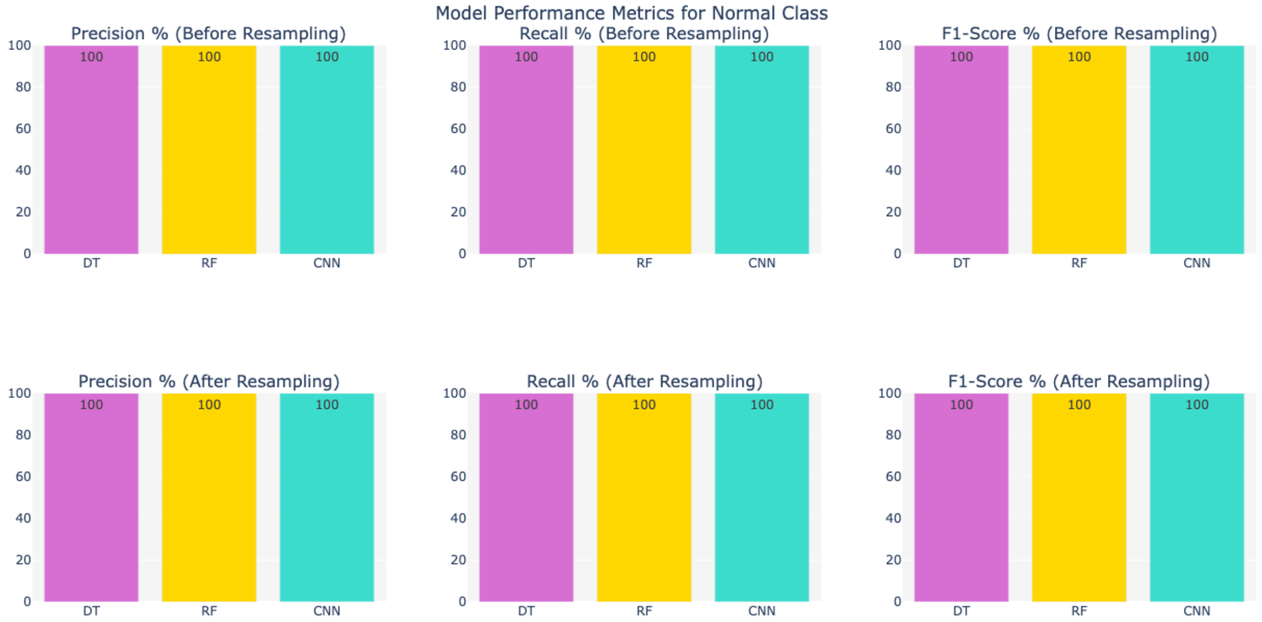

Model Performance Metrics for Backdoor Class
Precision % (Before Resampling)
0
20
40
60
80
100
DT 48
RF 39
CNN 37
Recall % (Before Resampling)
DT 15
RF 7
CNN 4
F1-Score % (Before Resampling)
DT 23
RF 12
CNN 7
Precision % (After Resampling)
DT 47
RF 22
CNN 67
Recall % (After Resampling)
DT 14
RF 9
CNN 1
F1-Score % (After Resampling)
DT 22
RF 13
CNN 3

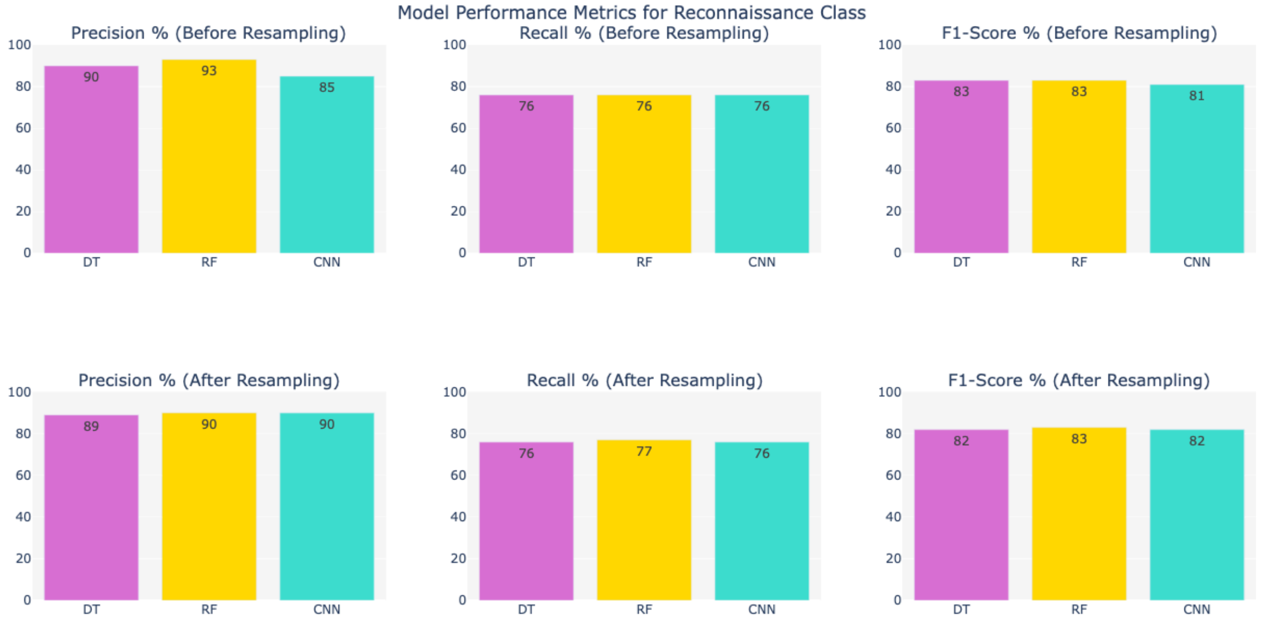

Model Performance Metrics for DoS Class
Precision % (Before Resampling)
0
20
40
60
80
100
DT 42
RF 41
CNN 54
Recall % (Before Resampling)
DT 38
RF 18
CNN 3
F1-Score % (Before Resampling)
DT 40
RF 25
CNN 6
Precision % (After Resampling)
DT 40
RF 40
CNN 60
Recall % (After Resampling)
DT 34
RF 24
CNN 4
F1-Score % (After Resampling)
DT 37
RF 30
CNN 7

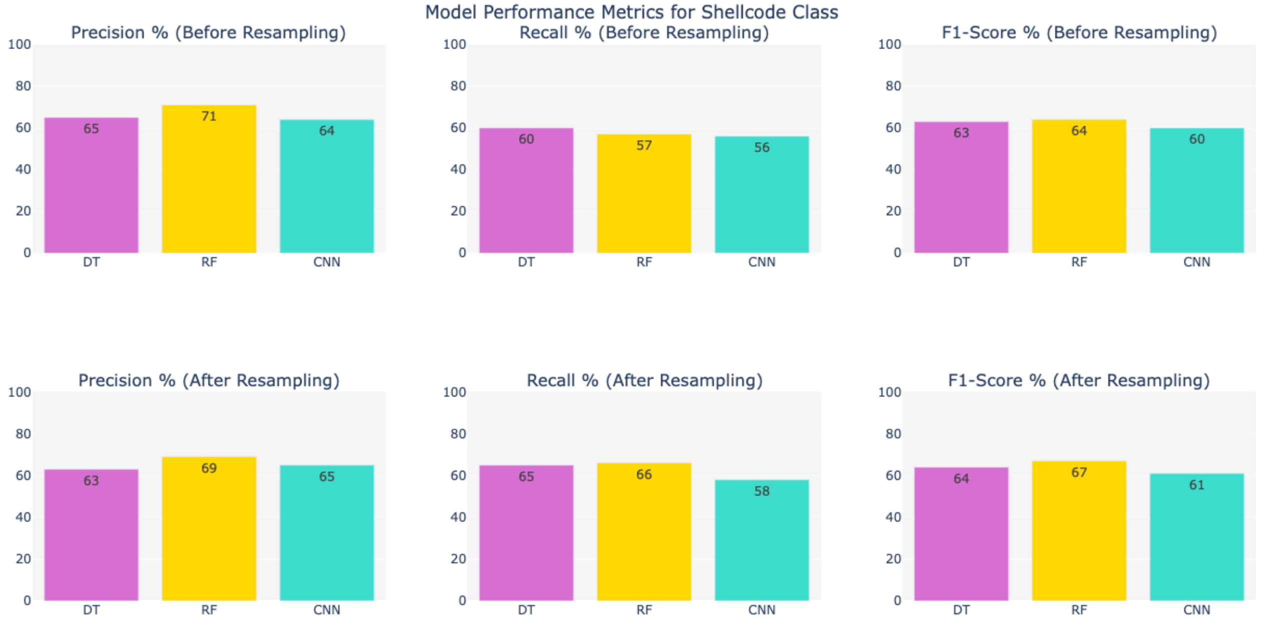

Model Performance Metrics for Exploits Class
Precision % (Before Resampling)
100
80
60
40
20
0
69
64
60
DT
RF
CNN
Recall % (Before Resampling)
81
90
94
DT
RF
CNN
F1-Score % (Before Resampling)
74
74
74
DT
RF
CNN
Precision % (After Resampling)
68
66
62
DT
RF
CNN
Recall % (After Resampling)
80
86
93
DT
RF
CNN
F1-Score % (After Resampling)
74
75
74
DT
RF
CNN

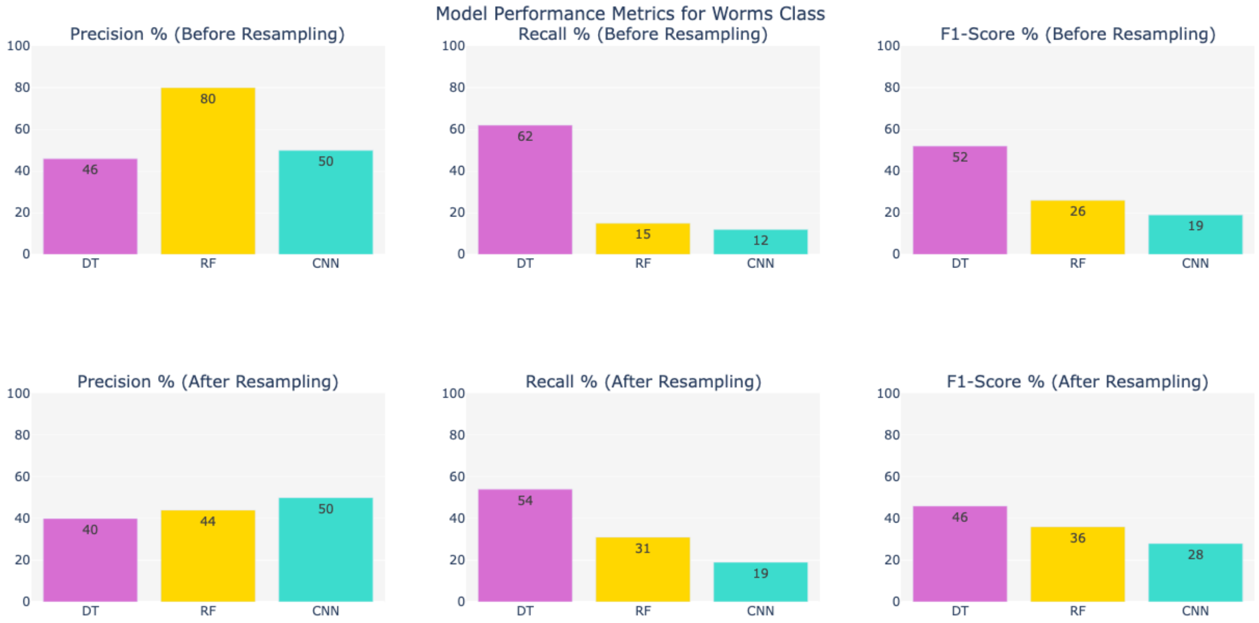

Model Performance Metrics for Fuzzers Class
Precision % (Before Resampling)
100
80
60
40
20
0
88
91
91
DT
RF
CNN
Recall % (Before Resampling)
88
88
83
DT
RF
CNN
F1-Score % (Before Resampling)
88
90
87
DT
RF
CNN
Precision % (After Resampling)
87
91
86
DT
RF
CNN
Recall % (After Resampling)
88
89
88
DT
RF
CNN
F1-Score % (After Resampling)
87
90
87
DT
RF
CNN


Model Performance Metrics for Generic Class
Precision % (Before Resampling)
Recall % (Before Resampling)
F1-Score % (Before Resampling)
100
80
60
40
20
0
DT
RF
CNN
99
100
100
98
98
98
99
99
99
Precision % (After Resampling)
Recall % (After Resampling)
F1-Score % (After Resampling)
99
100
100
98
98
98
99
99
99


Model Performance Metrics for Normal Class
Precision % (Before Resampling)
Recall % (Before Resampling)
F1-Score % (Before Resampling)
100
80
60
40
20
0
DT
RF
CNN
100
100
100
100
100
100
100
100
100
Precision % (After Resampling)
Recall % (After Resampling)
F1-Score % (After Resampling)
100
100
100
100
100
100
100
100
100


Model Performance Metrics for Reconnaissance Class
Precision % (Before Resampling)
Recall % (Before Resampling)
F1-Score % (Before Resampling)
Precision % (After Resampling)
Recall % (After Resampling)
F1-Score % (After Resampling)
100
80
60
40
20
0
DT
RF
CNN
90
93
85
76
76
76
83
83
81
89
90
90
76
77
76
82
83
82


Model Performance Metrics for Shellcode Class
Precision % (Before Resampling)
Recall % (Before Resampling)
F1-Score % (Before Resampling)
Precision % (After Resampling)
Recall % (After Resampling)
F1-Score % (After Resampling)
100
80
60
40
20
0
DT
RF
CNN
65
71
64
60
57
56
63
64
60
63
69
65
65
66
58
64
67
61



Moreover, it has been observed in the confusion matrix for each model that the issue of false negatives and positives is mainly associated with minority classes like 'Analysis', 'Dos', 'Backdoor', 'Shellcode', and 'Worms'. Models without sampling had a tendency to predict these minority classes as the majority class, leading to false negatives and false positives. With sampling, although there were still false negatives and positives, the model's ability to recognize these classes improved, suggesting that the strategy was partially successful.

Deciding whether to use a sampling strategy depends on the unique demands of the current task. Sampling can be advantageous in order to avoid missing any attack types, even if it results in a slight decrease in overall scores. To improve future model development, it is important to explore strategies that reduce false negatives and false positives, specifically in important minority groups, while still maintaining the overall performance of the model. This could involve the use of more advanced sampling techniques, feature engineering, or fine-tuning the model.

Chapter 6: Conclusion and Future Scope

This project compared various machine learning and deep learning models for network intrusion detection using the CICIDS 2017 and UNSW-NB15 datasets. The project focused on multi-class classification for categorizing network traffic into distinct attack classes.

- Model Performance: Decision Trees (DT) and Random Forests (RF) consistently showed better performance in both datasets. These models demonstrated high accuracy, precision, recall, and F1 scores, proving their effectiveness in handling class imbalance and accurately detecting majority and minority attack classes.
- Class Imbalance Strategies: Methods for dealing with class imbalance, such as the Minority Favored Strategy, were essential for enhancing the identification of minority attack classes. The method used for the CICIDS dataset included increasing the number of samples in minority classes to reach a ratio of approximately 1.12 malicious instances to 1 normal instance. One approach was used for the UNSW-NB15 dataset, leading to a sample ratio of approximately 2.21:1 between the minority and majority classes.
- Traditional ML Models: Classical machine learning models like Logistic Regression (LR), Support Vector Machines (SVM), and Naive Bayes (NB) faced challenges when dealing with minority classes like web attacks (Brute Force, SQL Injection, XSS), Bot, Heartbleed, and Infiltration in the CICIDS dataset. The Minority Favored Strategy showed better results in these courses, although with a slightly lower overall accuracy.
- Deep Learning Models: Advanced Deep Learning models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks demonstrated enhanced effectiveness in identifying minority classes following the utilization of sampling techniques. These models showed improved efficiency in identifying rare attack types compared to certain traditional machine learning methods.
- Challenges specific to the dataset: The CICIDS 2017 dataset presented more significant class imbalance challenges compared to the UNSW-NB15 dataset. This highlighted the need for dataset-specific strategies in model development and evaluation.

- Performance Trade-offs: The study revealed trade-offs between overall accuracy and the ability to detect minority classes. Sampling strategies often led to improved detection of rare attack types at the cost of slight decreases in overall performance metrics.
- Limitations of Overall Metrics: A critical finding of this study is that high overall accuracy and other metrics are not always indicative of a model's true performance, especially in imbalanced datasets. In many cases, these inflated metrics resulted from the model's failure to correctly classify minority classes. This observation underscores the importance of the sampling techniques employed in our study. For instance, in the CICIDS dataset, models like Logistic Regression, SVM, and Naive Bayes initially showed high overall accuracy but performed poorly on minority classes such as web attacks, Bot, Heartbleed, and Infiltration. After applying the Minority Favored Strategy, we observed a slight decrease in overall metrics, but a significant improvement in the correct classification of these crucial minority classes. Similarly, for the UNSW dataset, the confusion matrices revealed that minority classes like 'Analysis', 'Backdoor', 'DoS', 'Shellcode', and 'Worm' were frequently misclassified or overlooked, despite high overall accuracy scores. The application of sampling techniques led to enhanced detection of these minority classes, demonstrating that a small decrease in overall metrics can actually represent a more balanced and effective model. This finding highlights the necessity of looking beyond aggregate metrics and carefully examining the model's performance across all classes, especially in security-critical applications like intrusion detection.

6.1 Implications

- The study underscores the importance of addressing class imbalance in cybersecurity datasets. Appropriate sampling strategies can significantly improve the detection of rare but potentially severe attack types.
- The findings highlight the need for a nuanced approach to model evaluation in cybersecurity, where the ability to detect minority classes may be more crucial than overall accuracy metrics.
- The study provides practical insights for selecting and fine-tuning machine learning models for intrusion detection systems, emphasizing the strengths of ensemble methods like Random Forests.

- Our results caution against over-reliance on aggregate performance metrics, particularly in imbalanced datasets. A more holistic evaluation approach, considering performance on individual classes, is essential for developing truly effective intrusion detection systems.

6.2 Future Scope

1. **Advanced Sampling Techniques**: Explore more sophisticated sampling methods or hybrid approaches to further improve the balance between overall accuracy and minority class detection.
2. **Feature Engineering**: Investigate the impact of feature selection and engineering techniques on model performance, particularly for improving the detection of challenging minority classes.
3. **Ensemble Methods**: Develop and evaluate ensemble models that combine the strengths of different algorithms to enhance overall detection capabilities.
4. **Real-time Performance**: Assess the computational efficiency and real-time performance of these models in practical network environments.
5. **Model Interpretability**: Incorporate techniques to improve the interpretability of complex models like Random Forests and neural networks, which is crucial for building trust in AI-based intrusion detection systems.
6. **Class-Specific Performance Metrics**: Develop and standardize evaluation metrics that better reflect a model's performance across all classes, especially in imbalanced datasets.

In conclusion, this project has demonstrated the potential of machine learning and deep learning approaches in network intrusion detection while highlighting the complexities involved in handling imbalanced datasets. It emphasizes the importance of careful model evaluation, particularly in correctly identifying minority classes that may represent critical security threats. The development of robust, adaptive, and balanced intrusion detection systems remains a critical area of research in the field of cybersecurity, particularly as cyber threats continue to evolve in sophistication and frequency.

References

[1] Sharafaldin, Iman & Habibi Lashkari, Arash & Ghorbani, Ali. (2018). *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*. 108-116. 10.5220/0006639801080116.

[2] N. Moustafa, J. Slay, *UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), in 2015 Military Communications and Information Systems Conference (MilCIS).* (Springer, 2015), pp. 1–6. https://doi.org/10.1109/MilCIS.2015.7348942

[3] Ahmad, M., Riaz, Q., Zeeshan, M. *et al. Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set. J Wireless Com Network 2021, 10 (2021).* https://doi.org/10.1186/s13638-021-01893-8

[4] Besharati, E., Naderan, M. & Namjoo, E. *LR-HIDS: logistic regression host-based intrusion detection system for cloud environments*. *J Ambient Intell Human Comput* **10**, 3669–3692 (2019). https://doi.org/10.1007/s12652-018-1093-8

[5] Cervantes, J.; Garcia-Lamont, F.; Rodríguez-Mazahua, L.; Lopez, A. *A comprehensive survey on support vector machine classification: Applications, challenges and trends.* Neurocomputing 2020, 408, 189–215.

[6] Kumar, M.N.; Koushik, K.V.S.; Deepak, K. *Prediction of heart diseases using data mining and machine learning algorithms and tools.* Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol. 2018, 3, 887–898.

[7] Rahman, M.A.; Asyhari, A.T.; Leong, L.S.; Satrya, G.B.; Tao, M.H.; Zolkipli, M.F. *Scalable machine learning-based intrusion detection system for IoT-enabled smart cities*. Sustain. Cities Soc. 2020, 61, 102324