An FPA-Optimized XGBoost Stacking for Multi-Class Imbalanced Network Attack Detection

Hui Fern Soon[1], Amiza Amir[2], Hiromitsu Nishizaki[3],
Nik Adilah Hanin Zahri[4], Latifah Munirah Kamarudin[5]
Faculty of Electronic Engineering & Technology, Universiti Malaysia Perlis, Arau, Perlis, Malaysia[1]
University of Yamanashi, Kofu, Yamanashi, Japan[1]
Faculty of Electronic Engineering & Technology-Centre of Excellence for Advanced Computing (ADVCOMP),
Universiti Malaysia Perlis, Arau, Perlis, Malaysia[2]
Integrated Graduate School of Medicine, Engineering & Agricultural Science, University of Yamanashi, Kofu, Yamanashi, Japan[3]
CoE Advanced Computing (ADVCOMP), Universiti Malaysia Perlis, Arau, Perlis, Malaysia[4]
Centre of Excellence for Advanced Sensor Technology (CEASTECH), Universiti Malaysia Perlis, Arau, Perlis, Malaysia[5]

***Abstract*—Network anomaly detection systems face challenges with imbalanced datasets, particularly in classifying underrepresented attack types. This study proposes a novel framework for improving F1-scores in multi-class imbalanced network attack detection using the UNSW-NB15 dataset, without resorting to resampling techniques. Our approach integrates Flower Pollination Algorithm-based hyperparameter tuning with an ensemble of XGBoost classifiers in a stacking configuration. Experimental results show that our FPA-XGBoost-Stacking model significantly outperforms individual XGBoost classifiers and existing ensemble models. The model achieved a higher overall weighted F1-score compare to the individual XGBoost classifier and Thockchom et al.'s heterogeneous stacking ensemble. Our approach demonstrated remarkable effectiveness across various levels of class imbalance, for example Analysis and Backdoor which is highly underrepresented classes, and DoS which is moderately underrepresented class. This research contributes to more effective network security systems by offering a solution for imbalanced classification without resampling techniques' drawbacks. It demonstrates that homogeneous stacking with XGBoost can outperform heterogeneous approaches for skewed class distributions. Future work will extend this approach to other cybersecurity datasets and explore its applicability in real-time network environments.**

***Keywords*—Intrusion detection; multi-class imbalanced classification; ensemble learning approaches**

I. Introduction

The proliferation of digital technology has led to a rise in cybersecurity concerns. The European Union Agency for Cybersecurity (ENISA) [1] reports that from the end of 2022 to the beginning of 2023, there was an increase in malware attack events. The increasing frequency of cyberattacks places sensitive data at danger of compromise. Researchers frequently use deep learning and machine learning algorithms to classify network traffic. The effectiveness of these algorithms in accurately classifying network traffic depends on the data used to train them. Typically, normal traffic constitutes the majority of training datasets, while abnormal traffic includes various potential attack types. Rare or unusual attacks are often underrepresented compared to regular or more common attacks, leading to an uneven class distribution. This imbalance can cause model bias towards the majority class [2], resulting in inaccurate predictions and difficulty in detecting rare network attacks.

Sampling approaches are commonly used to address data imbalance. Oversampling techniques, such as Random Oversampling [3] and SMOTE (Synthetic Minority Oversampling) [4], increase minority class samples to match the majority class. Conversely, undersampling methods, like Tomek-link [5] and Random Undersampling [3], reduce majority class samples to achieve a balanced distribution. Hybrid sampling combines both oversampling and undersampling techniques. However, oversampling can lead to overfitting [3], and undersampling may result in information loss.

Some researchers have proposed ensemble methods to address imbalanced datasets without resampling techniques [6], [7], [8]. However, these methods are typically applied only to binary classification. It is undeniable that some works involve ensemble approaches for multi-class imbalanced classification in network intrusion systems [9], [10], [11]. These works use a heterogeneous stacking ensemble approach, employing different algorithms as the base classifiers for the stacking model.

XGBoost model has shown superior ability to handle multi-class imbalanced classification [12] for network attack classification. It has also been widely used as a base learner in heterogeneous stacking models for multi-class imbalanced classification in network attack detection [13], [14], [15]. However, the use of XGBoost in homogeneous stacking ensemble approaches, where multiple instances of the same algorithm are used as base learners to solve imbalanced data, is limited and has not been applied to multi-class network attack detection [16]. The potential of homogeneous stacking with XGBoost, remains largely unexplored in the context of network attack classification. This gap in the literature is significant, as homogeneous stacking could potentially offer advantages in terms of model consistency and interpretability by leveraging the strength of XGBoost especially when dealing with the complex, multi-class nature of network attacks. Our work aims to address this research gap by investigating the effectiveness of homogeneous stacking with XGBoost for imbalanced multi-class network attack classification, without

resorting to resampling techniques.

Although XGBoost has shown remarkable success compared to other machine learning algorithms [12], some researchers [17], [18], [19], [20], [21] have applied hyperparameter tuning techniques to further enhance its performance in network attack detection. Additionally, researchers such as [22], [23], [24] have involved hyperparameter tuning such as grid search and random search with heterogeneous stacking models that use XGBoost as one of the base classifiers for network anomaly detection. While meta-heuristic algorithms have shown promising results in optimizing machine learning hyperparameters across various domains, there remains a significant gap in their application to network attack classification, particularly in conjunction with XGBoost. However, the integration of the Flower Pollination Algorithm (FPA) with XGBoost for hyperparameter tuning in the context of network security remains largely unexplored. Existing studies utilizing FPA and XGBoost have primarily focused on regression models in fields such as civil engineering and environmental science, leaving a gap in their application to classification tasks in cybersecurity. Furthermore, the performance of meta-heuristic algorithms can vary significantly depending on the specific machine learning model and dataset characteristics.

Given that no single meta-heuristic algorithm consistently outperforms others across all tasks, there is a clear need to investigate the efficacy of FPA in optimizing XGBoost specifically for network attack classification. This study aims to address this research gap by exploring the potential of FPA-optimized XGBoost in the context of multi-class, imbalanced network attack detection, potentially offering new insights into improving the accuracy and robustness of intrusion detection systems.

In our approach, we first identify the optimal XGBoost models by optimizing the hyperparameters to maximise the base learner's performance on imbalanced datasets. The selection of these optimal models as the base learners are performed using hyperparameter tuning. Other than Flower Pollination Algorithm, four hyperparameter tuning techniques were investigated in this study: Random Search (RS), Bayesian Optimization (BO), Genetic Algorithms (GA), and Cuckoo Search Algorithms (CSA). Then, we integrate the hyperparameter tuning process with ensemble learning techniques to create a robust and effective ensemble classifier.

To summarize, the paper's contributions are as follows. First, the integration of FPA for optimization of XGBoost classifier in network security. Second, the paper shows that the proposed homogeneous stacking ensemble model with hyperparameter tuning, specifically the FPA-XGBoost-Stacking model, achieves better results than heterogeneous stacking model [11] for multi-class imbalanced network attack classification. The homogeneous FPA-XGBoost-Stacking model has proven effective by improving both overall detection performance and class-specific metrics compared to a standalone XGBoost model. These findings pave the way for addressing multi-class imbalanced classification issues using ensemble learning without the need for resampling techniques, with potential applications extending beyond network attack detection.

The structure of this paper is as follows: Section II reviews related work. Section III discusses XGBoost hyperparameters. Sections IV and V cover hyperparameter tuning and ensemble learning techniques. Section VI describes the datasets and experimental setup. Sections VII and VIII present the results of the optimized XGBoost models and ensemble models. Finally, Section IX provides the conclusions.

II. Related Works

Imbalanced classification presents a significant challenge in machine learning (ML), characterised by a notable difference in the number of instances between classes. This imbalance can lead to biased models that produce poor results for the minority class. Ensemble learning, a technique that combines multiple models to enhance performance, has been studied to tackle this problem. Ensemble ML techniques involve the combination of multiple base learners using a specific combination rule to create improved predictive models. Base learners may include a wide range of ML algorithms, such as decision trees, Naïve Bayes, K nearest neighbours, artificial neural networks, and logistic regression. The ensemble topology can range from a basic collaboration of individual learners combined through a majority vote to more sophisticated mechanisms. Research has indicated that incorporating multiple classification methods enhances performance scores [25], [26].

The Geometric Structural Ensemble (GSE) [27], Hybrid Data-Level Ensemble (HD-Ensemble) [28] and `sBal_IH` [6] applied both resampling and ensemble approaches in their works. The Geometric Structural Ensemble (GSE) learning framework effectively tackles imbalanced classification issues by leveraging geometric structures to partition and eliminate redundant majority samples. GSE uses the Euclidean metric to create hyper-spheres that contain minority samples, improving training efficiency and interoperability. The framework also incorporates relaxation techniques to improve generalization [27]. The Hybrid Data-Level Ensemble (HD-Ensemble) uses both undersampling on the margins and oversampling to improve diversity and balance the distribution of data in order to get the best ensemble properties [28]. The HD-Ensemble effectively rebalances data distribution and enhances performance in binary classification tasks. Kaixiang Yang et al. [29] introduced a hybrid optimal ensemble classifier framework that integrates density-based undersampling with cost-sensitive techniques to address class imbalances. This method employs a multi-objective optimisation algorithm to choose informative samples and adjust the weights of misclassified minority samples. The dual-ensemble class imbalance learning method integrates resampling techniques with multi-classifier models. It uses evolutionary algorithms to optimize the combination of base classifiers, achieving better accuracy and simpler ensemble structures. This method outperforms other ensemble classification methods on human activity recognition datasets [30]. A medical diagnosis system uses an ensemble learning approach that combines SMOTE with cross-validated committee filters and utilises ensemble support vector machines (SVM). This approach utilises a simulated annealing genetic algorithm to optimize the weight vector [31]. However, most of these methods modify the initial distribution of classes to achieve a more balanced dataset [30], [29], [27], [28]. This is done through techniques such as over-sampling or under-sampling. These techniques can result in overfitting or

the removal of valuable data, which may eventually impede performance [6].

Without resampling, an ensemble method, `sBal_IH`, creates balanced splits of data based on instance hardness. This approach trains base learners on varied characteristics of the training data, significantly improving classification performance [6]. Chih-Fong Tsai et al. [7] and Hongle Du et al. [8] propose methods to solve imbalance class problems in network security. One-class classification (OCC) techniques, typically used for anomaly detection, are applied to two-class imbalanced datasets. Ensemble learning with OCC classifiers, both with and without feature selection, outperforms single OCC classifiers, demonstrating effectiveness in high imbalance ratio datasets [7]. An online ensemble learning algorithm for imbalanced data streams employs cost-sensitive techniques to dynamically adjust misclassification costs and sample weights. This method improves classification performance in imbalanced data streams, as demonstrated in network intrusion detection applications, reducing both false alarm and missing alarm rates [8]. These advancements highlight the potential of ensemble learning to address the challenges posed by imbalanced datasets effectively. However, these studies only applied to binary classification [6], [7], [8].

Recent research in network attack detection has explored ensemble methods to address imbalanced datasets in multi-class classification scenarios. Mansoor-ul-Haque et al. [9] proposed a heterogeneous stacking ensemble for multi-class network intrusion detection, utilizing resampling techniques with KNN, SVM, and RF as base learners and XGBoost as the meta-learner. Similarly, Thockchom et al. [11] developed a heterogeneous stacking ensemble model for multi-class intrusion detection, integrating Gaussian Naive Bayes, Logistic Regression, and Decision Tree as base classifiers with Stochastic Gradient Descent as the meta-classifier. Rajadurai and Gandhi [10] evaluated a stacked ensemble learning approach on the NSL-KDD dataset with four attack categories, employing Random Forest and Gradient Boosting. The majority of these approaches employ heterogeneous stacking ensembles. However, instead of using a variety of base learners as in [9], [10], [11], this study opts for using homogeneous learners by using XGBoost algorithm. This decision is based on the considering the ability of XGBoost in handling imbalanced multi-class classification effectively [12].

Meta-heuristic algorithms, which are optimisation techniques inspired by natural processes, have been increasingly used for ML parameter optimisation. Meta-heuristic algorithms have been successfully applied to optimize hyperparameters in diverse domains, including sentiment analysis, image reconstruction, and landslide susceptibility mapping [32], [33]. Meta-heuristic algorithms such as particle swarm optimisation (PSO), genetic algorithm (GA), and others have been shown to effectively optimize hyperparameters, leading to improved model performance across various ML tasks [32], [34], [33]. Meta-heuristics can outperform traditional methods like grid search and random search in terms of accuracy and computational efficiency [32], [34], [33]. The integration of meta-heuristics with ML models has led to significant improvements in predictive performance and robustness [35], [34], [33].

The integration of XGBoost and FPA algorithm for hyperparameter tuning has rarely been investigated. Flower Pollination Algorithm (FPA) has been used to optimize the regression XGBoost models in civil engineering [36], [37] and environmental science [38]. The existing models have mainly focused on building regression models. Studies indicate that no single meta-heuristic algorithm consistently outperforms others across all tasks. The performance can vary depending on the specific ML model and the nature of the dataset [34]. Hence, the current work attempts to fill these gaps in the literature by employing FPA in optimizing XGBoost for network attack classification.

III. XGBoost Hyperparameters

A. Overview of XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful and widely-used machine learning algorithm under the Gradient Boosting framework. It combines multiple weak learners (decision trees) to create a strong learner by iteratively training on the residuals of previous trees, minimising the loss function, and enhancing overall performance. Known for its speed and efficiency, XGBoost is popular for various ML tasks, including classification and regression.

The XGBoost algorithm can be expressed using the following equation:

$$\hat{y}_i = \sum_{k=1} f_k(x_i), \quad f_k \in \mathcal{F} \tag{1}$$

where: $\hat{y}_i$ is the predicted value for the i-th instance, K is the total number of trees (iterations), f_k is the k-th tree (weak learner), x_i is the input feature vector for the i-th instance, and $\mathcal{F}$ is the space of possible trees (weak learners).

The objective function of XGBoost consists of two parts: the loss function and the regularization term.

$$\text{Obj}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2}$$

Where $\text{Obj}(\theta)$ is the objective function to be minimized, n is the total number of instances, $l(y_i, \hat{y}_i)$ is the loss function, such as squared error or log loss, and $\Omega(f_k)$ is the regularization term for the k-th tree, which penalizes the complexity of the model.

The gain function in XGBoost determines the optimal split point for a decision tree node by measuring the improvement in the loss function after splitting the node into two child nodes. The split with the highest gain is selected as the best split for the current node. The split with the highest gain is chosen as the best for the current node, and the process is recursively repeated for the child nodes until a stopping criterion is met, such as maximum depth or minimum number of instances in a leaf.

B. Hyperparameters

The performance of an XGBoost model can be improved by tuning various hyperparameters. Below are several critical hyperparameters and their influence on the model's performance:

1) Number of estimators: The number of estimators refers to the number of decision trees (also known as weak learners or base learners) that are built and combined to create the final ensemble model. A hyperparameter determines the total number of trees to be trained in the XGBoost model. Each estimator is a decision tree that is trained on a subset of the training data and contributes to the final prediction.

2) Learning Rate (eta): The learning rate determines the step size at which the model's weights are updated. A lower learning rate leads to slower convergence but can help reduce overfitting.

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta \cdot f_t(x_i) \tag{3}$$

where $\hat{y}_i^{(t)}$ is the prediction for the i-th instance at iteration t, $\hat{y}_i^{(t-1)}$ is the prediction for the i-th instance at the previous iteration $t-1$, η is the learning rate, and $f_t(x_i)$ is the output of the t-th tree for the i-th instance.

3) Maximum depth (max_depth): The maximum depth of a tree controls the complexity and the ability of the model to capture interactions among features. Increasing the maximum depth can lead to overfitting while decreasing it can result in underfitting. A deeper tree can capture more complex patterns, but it may also overfit the training data.

4) Minimum child weight (min_child_weight): It defines the minimum sum of instance weights (hessian) needed in a child node. A higher value prevents overfitting by avoiding the creation of too many child nodes with low instance weights.

$$\sum_{i \in I_L} H_i \geq \texttt{min\_child\_weight} \tag{4}$$

where I_L is the set of instances in the left child node, and H_i is Hessian (second-order gradient) for instance i.

5) Subsample: Subsample is the fraction of observations to be randomly sampled for each tree. A value less than 1 introduces randomness and helps in reducing overfitting.

6) Colsample_bytree: It is the subsample ratio of columns (features) when constructing each tree. A value less than 1 introduces randomness and helps in reducing overfitting by considering only a subset of features for each tree.

7) Gamma: Gamma is the minimum loss reduction required to partition the leaf node of the tree further. Increasing gamma makes the model more conservative and can help reduce overfitting.

Tuning these hyperparameters can significantly impact the performance of an XGBoost model. The optimal values depend on the specific dataset and problem at hand. It is common to use techniques like grid search or random search to find the best combination of hyperparameters that maximize the model's performance on a validation set. Given the computational constraints and the exponential growth in possible parameter combinations, this investigation concentrates on a carefully selected subset of hyperparameters. The study emphasizes the optimization of four key parameters: the quantity of estimators, the rate of learning, the maximum depth of trees, and the minimum weight required for child nodes. These specific hyperparameters were chosen due to their substantial influence on both the model's efficacy and its capacity to generalize. Moreover, these parameters play a crucial role in determining the model's ability to learn from the data and in balancing the trade-off between bias and variance in its predictions.

IV. Hyperparameter Tuning

Hyperparameter tuning is a critical step in optimising the performance of ML models. Bayesian optimisation and Genetic Algorithm, have shown promising results in tuning support vector machines (SVM) and random forests (RF) [35], [33]. In this study, other than two commonly used hyperparameter tuning techniques: Random Search and Bayesian Optimization, we also applied three metaheuristics algorithms: Flower Pollination Algorithm (FPA), Cuckoo Search Algorithm (CSA), and Genetic Algorithm (GA), to tune the hyperparameters discussed in Section III-B.

A. Objective Function

In optimisation, an objective function is a mathematical function that needs to be minimised or maximised to find the optimal solution to a problem. In this study, the objective function is the weighted F1-score, a common metric for imbalanced classification problems. We focuses on the weighted F1-score rather than the macro-average F1-score because the weighted F1-score accounts for the imbalance in the dataset. It reflects the classifier's performance across all classes by considering the actual data distribution [39]. In contrast, the macro-average F1-score treats all classes equally [39], without accounting for class imbalance, which can be misleading for imbalanced datasets. Therefore, this study prioritises improving the weighted F1-score performance.

The F1-score (weighted) is calculated (see Eq. 5) as follows:

$$F1_{weighted} = \frac{\sum_{i=1}^{n} 2 \times \frac{precision_i \times recall_i}{precision_i + recall_i} \times w_i}{\sum_{i=1}^{n} w_i} \tag{5}$$

where: n is the number of classes, $precision_i$ is the precision for class i, $recall_i$ is the recall for class i, and w_i is the weight for class i, which is usually the number of instances in class i.

By using the weighted F1-score as the objective function, the hyperparameter optimisation ensures the resulting XGBoost model is optimized to perform well across all classes, considering the class imbalance in the dataset.

B. Random Search (RS)

Random search (RS) is a simple but effective method used for optimisation. In ML, random search is commonly used for hyperparameter tuning, which aims to find the best set of hyperparameters to maximize a model's performance [40]. With a set budget, random search randomly samples hyperparameters and assesses the model's performance for each hyperparameter combination. Unlike more complex methods

such as grid search or Bayesian optimisation, random search involves randomly sampling points in the parameter space and evaluating the objective function at these points [32]. It is easy to implement and does not require any prior knowledge about the problem's structure.

C. Bayesian Optimisation (BO)

Bayesian optimisation (BO) is a highly effective and efficient method for hyperparameter tuning in ML [41], [32]. It consistently outperforms traditional methods and is applicable across a wide range of models. Advanced techniques and practical tools enhance its utility, making it a preferred choice for optimizing complex ML algorithms. It leverages probabilistic models to make informed decisions about where to evaluate the objective function next, aiming to find the global optimum with as few evaluations as possible. It is efficient in high-dimensional spaces and works well with expensive-to-evaluate functions.

D. Flower Polination Algorithm (FPA)

The Flower Pollination Algorithm (FPA) is a nature-inspired optimisation technique that mimics the pollination process of flowering plants. Introduced by Xin-She Yang [42], it leverages the principles of flower constancy and pollination to solve complex optimisation problems. The FPA is an attractive choice for optimisation due to its simplicity and ease of implementation, having only one main control parameter, which makes tuning straightforward and less sensitive to settings [42]. FPA effectively balances exploration and exploitation through global pollination (Lévy flights) for exploration and local pollination for exploitation. It has been successfully applied to various optimisation problems, including continuous, discrete, and multi-objective cases [43].

E. Genetic Algorithm (GAs)

Genetic Algorithms (GAs) are evolutionary algorithms inspired by natural selection and genetics. Introduced by John Holland in the 1970s, GAs solve optimisation problems by evolving a population of candidate solutions through selection, crossover, and mutation operators [44]. They have since been widely applied across various domains. GAs are effective for solving optimisation problems [45]. They can explore a wide range of solutions and escape local optima due to the stochastic nature of the operators. GAs handle complex, non-linear problems without needing gradient information and are flexible, easily adaptable to various domains with appropriate representation schemes and operators.

F. Cuckoo Search Algorithm (CSA)

The Cuckoo Search Algorithm (CSA) is a nature-inspired optimisation technique introduced by Xin-She Yang and Suash Deb in 2009 [46]. It mimics the brood parasitism behaviour of some cuckoo species, which lay their eggs in the nests of other birds. Host birds may discard these eggs or abandon their nests, a concept CSA uses to search for optimal solutions in a problem space. The CSA is simple to implement and balances exploration and exploitation effectively through Lévy flights [46]. It has demonstrated strong performance in various optimisation problems and has been successfully applied across different domains.

V. Ensemble Combination Techniques

Rather than relying on a single model, combining several models, known as the ensemble technique, leads to more accurate classification predictions [47]. This method aims to enhance classification performance by integrating multiple models and has been widely adopted in numerous studies. Ensemble learning involves using the output of base classifiers as input for a new classifier. In this research, the stacking and voting ensemble learning approach were employed.

Stacking, introduced by Wolpert in 1992 [48], is an ensemble technique that minimises generalisation error in ML. It commonly involves two layers: multiple base classifiers are trained in the first layer (level 0), and their predictions are fed into a meta-classifier in the second layer (level 1) for further training. The effectiveness of stacking depends on the selection of both base and meta-classifiers, better prediction results from the base classifiers improving overall predictions [49]. The meta-classifier combines these predictions to make the final decision, often using a simple model [47].

There are two types of voting techniques: hard voting and soft voting. Hard voting is an ensemble method where the predicted outcomes from different models are averaged based on the majority. Each model makes a prediction, and the instances are classified into the most frequently predicted class. In contrast, soft voting relies on the probabilities output by the base classifiers. It calculates the average probabilities for each class across the models and assigns the instance to the class with the highest average probability [50]. For example, if class 1 has a higher average probability than class 2, the instance is classified as class 1.

A. Optimized XGBoost Ensemble

This section details the generation of optimized XGBoost ensemble models, divided into two main phases as shown in Fig. 1. The first phase involves hyperparameter tuning to identify the best configurations for the XGBoost models. The second phase applies ensemble learning techniques, specifically stacking and voting, to enhance performance.

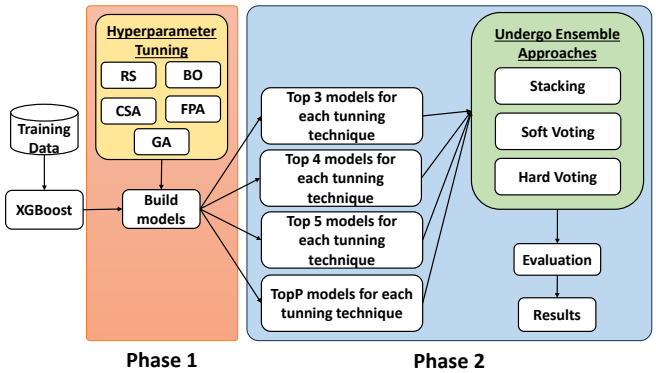

Fig. 1. The Framework for designing optimized XGBoost ensembles.

1) Phase 1 Hyperparameter tuning: Hyperparameter tuning is crucial for optimising ML models to achieve peak performance. In this research, it was used to refine XGBoost models for multi-class imbalanced classification in network attack detection. Initially, a broad range of hyperparameter

combinations was set, as detailed in Section VI-B. This ensured a thorough search for optimal settings. XGBoost models were tuned using five techniques: RS, BO, CSA, FPA, and GA. Each aimed to identify the top five models with the best hyperparameters for maximising the weighted F1-score, crucial for handling the imbalanced dataset. This process produced 25 optimized models, five from each technique. To ensure robustness and generalisability, stratified k-fold cross-validation was employed, splitting the training dataset into three subsets. This approach mitigated overfitting and provided reliable performance metrics. After tuning, the top five models from each technique were evaluated based on their weighted F1-scores, as shown in Table I.

2) Phase 2 Ensemble learning: After obtaining 25 optimized XGBoost models, the subsets of these models were combined through stacking and voting (soft and hard) ensemble techniques. Models from each hyperparameter tuning technique were grouped by performance into top three, top four, top five, and top-performing categories. This combination resulted in 18 optimized stacking ensemble models, 18 optimized soft voting ensemble models, and 18 optimized hard voting ensemble models.

The top n category includes the first n optimized XGBoost models from each hyperparameter tuning technique that achieved the highest weighted F1-scores. The top-performing category selects the best model from each tuning technique. For example in the FPA-Stacking ensemble models as shown in Fig. 2, FPA-Stacking 3 uses FPA-XGBoost 1, FPA-XGBoost 2, and FPA-XGBoost 3 as the base classifiers. This approach is similarly applied to the soft and hard voting ensemble models.

TopP-Stacking ensemble model utilises the top optimized XGBoost models from each hyperparameter tuning technique. For instance, TopP-Stacking3 combines the top-performing individual optimized XGBoost models: CSA-XGBoost 1 (TopP1), GA-XGBoost 1 (TopP2), and FPA-XGBoost 1 (TopP3) as the base classifiers for the stacking ensemble model. This approach is similarly applied to the TopP-SoftVoting and TopP-HardVoting ensemble models, where the same top-performing classifiers are used as the base models.

VI. Experiment

The performance of these optimized stacking and voting XGBoost ensemble models was evaluated on the testing dataset to assess their generalisation capability and classification accuracy for unseen data. The default XGBoost model was used as the meta-classifier for all stacking ensembles.

A. Dataset Description

The dataset used in this experiment is the UNSW-NB15 dataset, which is publicly accessible from the University of New South Wales (UNSW). This dataset consists of 257,673 records of network traffic, primarily categorized as either normal or attack traffic. It includes a total of 43 features, along with two label features [51]. The attack traffic is further classified into nine different types based on their characteristics: analysis, backdoor, DoS, exploits, fuzzers, generic, reconnaissance, shellcode, and worms, as illustrated in Fig. 3. The dataset is stratified and split into a 70:30 ratio, with 70% used as the training set and 30% as the test set. As shown in Fig. 3, attack classes such as "Analysis," "Backdoor," "Reconnaissance," "Shellcode," and "Worms" constitute less than 6% of the total instances, thus being identified as minority classes in this study. This reflects the complexity of real-life scenarios, where certain network attacks, despite their rarity, are of significant concern.

B. Experimental Setup

The XGBoost models were implemented in a Python environment. This model is capable of utilising GPU acceleration, which enhances the efficiency of hyperparameter tuning. The experiments compared the proposed approach with baseline individual default XGBoost model which also serves as the meta-classifier in each ensemble.

The MEALPY Python package [52] was utilised for hyperparameter tuning using metaheuristic algorithms. Due to the extensive combinatorial optimisation search space, it is necessary to establish a specific range for each parameter. For the four critical parameters mentioned in Section III-B, the tuning ranges are specified as follows: Number of estimators = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000], eta = [0.001, 0.01, 0.05, 0.1, 0.2, 0.3], minimum child weight = [1, 2, 3, 4, 5, 6], maximum depth = [3, 4, 5, 6, 7, 8, 9, 10, 11,12,13,14,15]. The other parameters are set as follows: $booster$= "$gbtree$", $gamma = 0$, $subsample = 1$, $\texttt{colsample\_bytree} = 1$, $\texttt{reg\_lambda} = 1$, $\texttt{tree\_method}$="$hist$", and $\texttt{random\_state} = 42$ $objective$="$multi : softmax$.

VII. Hyperparameter Tuning Results

Table I demonstrates the performance of the top five XGBoost models from five different hyperparameter tuning techniques, resulting in a total of 25 optimized models. The primary focus is on their weighted F1-scores during training, which are crucial for evaluating performance on imbalanced datasets. All models demonstrate strong training performance, with weighted F1-scores above 0.8180. The table organizes the models according to their weighted F1-scores within each hyperparameter tuning technique, facilitating comparison. For example, RS-XGBoost 1 refers to the model with the highest weighted F1-score from Random Search (RS) tuning, while RS-XGBoost 5 denotes the model with the lowest weighted F1-score among the top five from the same technique. This arrangement allows for a clear comparison of the effectiveness of each tuning method based on weighted F1-scores.

VIII. Optimized XGBoost Ensemble Models Results

A. Comparison of Optimized XGBoost-Stacking Models

Table II compares the performance of 18 stacking ensemble models against a baseline individual XGBoost model. The results show that all stacking ensemble models outperform the individual XGBoost model with improvements in accuracy, macro-average F1-score, and weighted F1-score. The weighted F1-scores of the stacking models range from 0.8307 to 0.8367, significantly higher than the individual XGBoost model's 0.8161. This highlights the effectiveness of the stacking ensemble approach in improving classification performance. Based on Table II, the FPA-stacking ensemble models outperform those using RS, BO, CSA, and GA-optimized models as

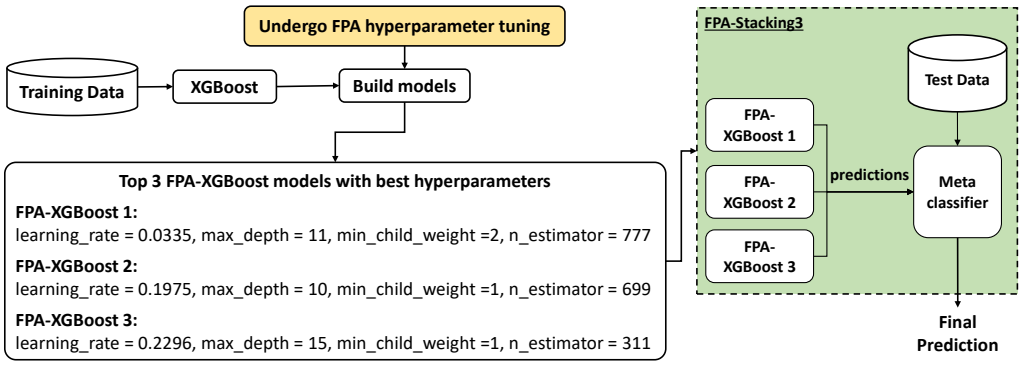

Fig. 2. FPA-XGBoost-Stacking of three models.

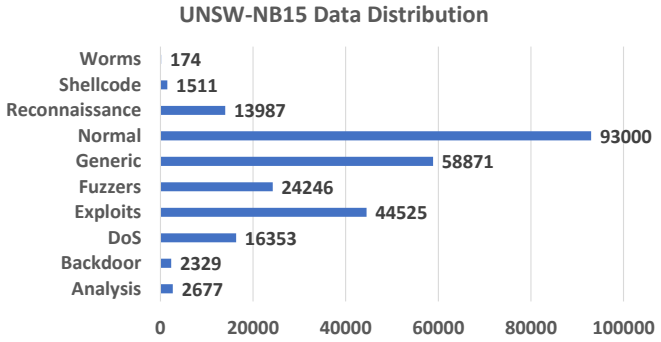

Fig. 3. UNSW-NB15 Data distribution.

base classifiers. They also surpass the TopP-stacking ensemble models, which use the top-performing optimized classifiers as base classifiers. Among the FPA-stacking ensemble models, FPA-Stacking3 achieved the highest weighted F1-score of 0.8367, representing a 2.524% improvement over the individual XGBoost model's score of 0.8161. Beyond achieving the highest weighted F1-score, FPA-Stacking3 also emerged as the top-performing stacking ensemble model among all 18 models, with a macro-average F1-score of 0.6266 and the highest accuracy of 0.8450.

B. Comparison of Optimized XGBoost-Voting Models

Tables III and IV show the results for 18 models of soft voting and hard voting, respectively. The tables reveal that all voting ensemble models showed slight improvements over the individual XGBoost model, particularly in terms of weighted F1-score. Soft voting models had weighted F1-scores from 0.8197 to 0.8224, while hard voting models ranged from 0.8197 to 0.8235, compared to 0.8161 for the individual XGBoost model. This demonstrates the effectiveness of the voting ensemble approach in enhancing classification performance. Based on Table III, the TopP-SoftVoting ensemble models outperform other soft voting ensembles that use RS, BO, CSA, FPA, and GA-optimized models as base classifiers in terms of weighted F1-score, with the CSA-SoftVoting5 ensemble model (0.8220) also emerging as a strong contender. Among the TopP-SoftVoting models, TopP-SoftVoting3 stands out, achieving the highest weighted F1-score of 0.8224.

TABLE I. HYPERPARAMETERS AND WEIGHTED F1-SCORES FOR 25 OPTIMIZED CLASSIFIER MODELS

Model	learning_rate	max_depth	min_depth	min_child_weight	Weighted F1-score (Train)
RS-XGBoost 1	**0.05**	**15**	**3**	**500**	**0.8196**
RS-XGBoost 2	0.2	6	2	200	0.8196
RS-XGBoost 3	0.05	9	2	800	0.8193
RS-XGBoost 4	0.05	13	6	400	0.8192
RS-XGBoost 5	0.1	7	5	800	0.8189
BO-XGBoost 1	**0.1**	**9**	**2**	**400**	**0.8194**
BO-XGBoost 2	0.1	8	4	800	0.8194
BO-XGBoost 3	0.1	8	4	400	0.8187
BO-XGBoost 4	0.1	8	3	400	0.8186
BO-XGBoost 5	0.1	8	6	400	0.8181
CSA-XGBoost 1	**0.0403**	**14**	**1**	**649**	**0.8201**
CSA-XGBoost 2	0.0655	10	2	892	0.8198
CSA-XGBoost 3	0.0589	12	1	936	0.8198
CSA-XGBoost 4	0.1964	10	4	398	0.8196
CSA-XGBoost 5	0.0656	10	2	893	0.8195
FPA-XGBoost 1	**0.0335**	**11**	**2**	**777**	**0.8198**
FPA-XGBoost 2	0.1975	10	1	699	0.8197
FPA-XGBoost 3	0.2296	15	1	311	0.8195
FPA-XGBoost 4	0.2737	9	2	265	0.8195
FPA-XGBoost 5	0.2263	10	2	347	0.8194
GA-XGBoost 1	**0.0898**	**12**	**2**	**727**	**0.8200**
GA-XGBoost 2	0.0898	11	3	660	0.8200
GA-XGBoost 3	0.1335	12	2	393	0.8198
GA-XGBoost 4	0.898	11	2	660	0.8198
GA-XGBoost 5	0.0898	12	2	660	0.8198

This represents a 0.7720% improvement over the individual XGBoost model's score of 0.8161. Additionally, Table III identifies FPA-XGBoost-SoftVoting3 as the ensemble model with the highest macro-average F1-score of 0.6010, while RS-XGBoost-SoftVoting3 stands out for achieving the highest accuracy of 0.8356.

Table IV shows that all hard voting ensemble models achieved a weighted F1-score above 0.8200, except for the BO-HardVoting3 ensemble model, which recorded a weighted F1-score of 0.8197. Among the 18 hard voting ensemble models, the FPA-HardVoting4 model stands out with the highest weighted F1-score of 0.8235, representing a 0.9068%

TABLE II. ACCURACY AND F1-SCORES PERFORMANCE FOR STACKING ENSEMBLES AND INDIVIDUAL CLASSIFIER MODELS

Algorithm	Accuracy	Macro-Average F1-score	Weighted F1-score
Individual XGBoost	0.8348	0.5945	0.8161
RS-XGBoost-Stacking3	0.8447	0.6205	0.8347
RS-XGBoost-Stacking4	0.8441	0.6194	0.8343
RS-XGBoost-Stacking5	0.8440	0.6197	0.8342
BO-XGBoost-Stacking3	0.8377	0.6146	0.8307
BO-XGBoost-Stacking4	0.8413	0.6151	0.8314
BO-XGBoost-Stacking5	0.8413	0.6151	0.8314
CSA-XGBoost-Stacking3	0.8430	0.6083	0.8314
CSA-XGBoost-Stacking4	0.8435	0.6062	0.8319
CSA-XGBoost-Stacking5	0.8433	0.6112	0.8325
FPA-XGBoost-Stacking3	**0.8450**	**0.6266**	**0.8367**
FPA-XGBoost-Stacking4	0.8441	0.6207	0.8355
FPA-XGBoost-Stacking5	0.8439	0.6228	0.8356
GA-XGBoost-Stacking3	0.8437	0.6153	0.8333
GA-XGBoost-Stacking4	0.8427	0.6147	0.8337
GA-XGBoost-Stacking5	0.8434	0.6115	0.8336
TopP-XGBoost-Stacking3	0.8436	0.6129	0.8343
TopP-XGBoost-Stacking4	0.8442	0.6120	0.8339
TopP-XGBoost-Stacking5	0.8444	0.6181	0.8339

TABLE III. ACCURACY AND F1-SCORES PERFORMANCE FOR SOFT VOTING ENSEMBLES AND INDIVIDUAL CLASSIFIER MODELS

Algorithm	Accuracy	Macro-Average F1-score	Weighted F1-score
Individual XGBoost	0.8348	0.5945	0.8161
RS-XGBoost-SoftVoting3	0.8356	0.5988	0.8207
RS-XGBoost-SoftVoting4	0.8355	0.5960	0.8208
RS-XGBoost-SoftVoting5	0.8350	0.5932	0.8204
BO-XGBoost-SoftVoting3	0.8353	0.5936	0.8197
BO-XGBoost-SoftVoting4	0.8351	0.5964	0.8199
BO-XGBoost-SoftVoting5	0.8353	0.5967	0.8199
CSA-XGBoost-SoftVoting3	0.8326	0.5975	0.8218
CSA-XGBoost-SoftVoting4	0.8330	0.5959	0.8219
CSA-XGBoost-SoftVoting5	0.8330	0.5965	0.8220
FPA-XGBoost-SoftVoting3	0.8320	0.6010	0.8219
FPA-XGBoost-SoftVoting4	0.8325	0.5994	0.8218
FPA-XGBoost-SoftVoting5	0.8324	0.6002	0.8218
GA-XGBoost-SoftVoting3	0.8323	0.5978	0.8219
GA-XGBoost-SoftVoting4	0.8324	0.5983	0.8218
GA-XGBoost-SoftVoting5	0.8324	0.5979	0.8219
TopP-XGBoost-SoftVoting3	**0.8336**	**0.5967**	**0.8224**
TopP-XGBoost-SoftVoting4	0.8335	0.5958	0.8221
TopP-XGBoost-SoftVoting5	0.8337	0.5945	0.8218

improvement over the individual XGBoost model's score of 0.8161. Additionally, Table IV indicates that the FPA-HardVoting3 model achieved the highest macro-average F1-score of 0.6010, while the RS-HardVoting3 model achieved the highest accuracy of 0.8356.

C. Per Class Comparison

We further perform a comparison of the proposed FPA-XGBoost-Stacking, focusing on performance improvements across classes achieved by the ensemble models compared to an individual XGBoost model. Considering FPA-XGBoost Stacking with three models performs the best, we compare the performance of the FPA-XGBoost Stacking with an individual XGBoost and another ensemble approach proposed by Thockchom et al. [11], This ensemble approach selected due to its relevance to the methods used in this research. Thockchom et

TABLE IV. ACCURACY AND F1-SCORES PERFORMANCE FOR HARD VOTING ENSEMBLES AND INDIVIDUAL CLASSIFIER MODELS

Algorithm	Accuracy	Macro-Average F1-score	Weighted F1-score
Individual XGBoost	0.8348	0.5945	0.8161
RS-XGBoost-HardVoting3	0.8356	0.6005	0.8209
RS-XGBoost-HardVoting4	0.8342	0.5971	0.8221
RS-XGBoost-HardVoting5	0.8353	0.5974	0.8210
BO-XGBoost-HardVoting3	0.8353	0.5931	0.8197
BO-XGBoost-HardVoting4	0.8352	0.5953	0.8211
BO-XGBoost-HardVoting5	0.8354	0.5930	0.8200
CSA-XGBoost-HardVoting3	0.8330	0.5986	0.8221
CSA-XGBoost-HardVoting4	0.8323	0.5962	0.8225
CSA-XGBoost-HardVoting5	0.8329	0.5979	0.8218
FPA-XGBoost-HardVoting3	0.8316	0.6010	0.8223
FPA-XGBoost-HardVoting4	**0.8316**	**0.5990**	**0.8235**
FPA-XGBoost-HardVoting5	0.8323	0.5991	0.8217
GA-XGBoost-HardVoting3	0.8320	0.5986	0.8217
GA-XGBoost-HardVoting4	0.8320	0.5993	0.8225
GA-XGBoost-HardVoting5	0.8320	0.5974	0.8216
TopP-XGBoost-HardVoting3	0.8333	0.5959	0.8223
TopP-XGBoost-HardVoting4	0.8325	0.5952	0.8226
TopP-XGBoost-HardVoting5	0.8339	0.5952	0.8221

al.'s [11] model is a heterogenous stacking ensemble involving Gaussian Naive Bayes, Logistic Regression, and Decision Tree as base classifiers, with Stochastic Gradient Descent as the meta-classifier, using the base classifiers' predictions as input.

TABLE V. F1-SCORES FOR DIFFERENT ATTACK CLASSES ACROSS MODELS

Attack class	Individual XGBoost (default)		FPA -XGBoost -Stacking		Thockchom et al. [11]	
Analysis	0.1861		0.2229	✓	0.0023	
Backdoor	0.1684		0.2085	✓	0.0336	
DoS	0.2041		0.4149	✓	0.1663	
Exploits	0.7420		0.7531	✓	0.7120	
Fuzzers	0.6395		0.6706	✓	0.6041	
Generic	0.9888		0.9894	*	0.9848	
Normal	0.9296		0.9350	✓	0.9135	
Reconnaissance	0.8384		0.8304		0.8114	
Shellcode	0.6763		0.6723		0.6099	
Worms	0.5714		0.5686		0.4296	
Weighted F1-score	0.8161		0.8367	✓	0.7934	
Accuracy	0.8343		0.8450	✓	0.8111	

(✓) Significant improvements, (*)Slight improvements

The results presented in Table V offer compelling evidence for the effectiveness of our proposed FPA-XGBoost-Stacking model in addressing the challenges of imbalanced network attack detection. One of the most striking findings is the model's performance on underrepresented attack classes. For instance, the F1-score for the DoS class improved dramatically from 0.2041 to 0.4149, a 103.3% increase. This is particularly significant given that DoS attacks comprise less than 2% of the samples compared to the majority class. Similarly, substantial improvements were observed for other rare attack types, with the Analysis class showing a 19.8% increase and the Backdoor class a 23.8% increase in F1-scores. The consistent improvements across almost all attack classes are noteworthy. Six out of ten classes showed significant enhancements, indicating that our model's performance boost is not limited to just a few categories. This broad-spectrum improvement suggests that the FPA-XGBoost-Stacking model has successfully captured

a wide range of attack patterns and characteristics.

When comparing our model to both the individual XGBoost classifier and Thockchom et al.'s heterogeneous stacking ensemble, the advantages of our approach become clear. The overall weighted F1-score of 0.8367 represents a 2.52% improvement over individual XGBoost and a 5.46% improvement over Thockchom et al.'s model. This demonstrates that our homogeneous stacking approach with XGBoost, combined with FPA-based hyperparameter tuning, outperforms both simpler and more complex heterogeneous models.

It is worth noting that there were slight decreases in performance for the Reconnaissance, Shellcode, and Worms classes compared to the individual XGBoost model. However, these decreases were minimal, and our model still outperformed Thockchom et al.'s approach in these categories. This suggests that while our model excels in most areas, there may be room for further optimization in detecting these specific attack types. The improved accuracy (0.8450 compared to 0.8343 for individual XGBoost and 0.8111 for Thockchom et al.'s model) further corroborates the overall enhanced performance of our approach. This indicates that our model not only improves the detection of underrepresented classes but also maintains high performance on more common attack types and normal traffic.

IX. Conclusion

The experimental results highlight the effectiveness of ensemble learning techniques, particularly homogeneous stacking and voting, in addressing imbalanced datasets for network attack detection. Homogeneous stacking tends to produce more stable and consistent results because all base models in the ensemble use the same algorithm. In contrast, heterogeneous ensembles might be more volatile. Different algorithms can react very differently to changes in the data, potentially leading to less stable overall predictions. Key findings reveal that combining hyperparameter tuning, XGBoost, and homogeneous stacking, enhances detection performance and class-specific metrics compared to standalone classifiers. Our research has also demonstrates that FPA effectively improves XGBoost's performance in the context of multi-class, imbalanced network attack detection offering new insights into enhancing the accuracy and robustness of intrusion detection systems. These findings not only fill a gap in the literature but also provide practical implications for developing more effective network security solutions.

The FPA-XGBoost-Stacking model outperformed both individual XGBoost and Thockchom et al.'s ensemble model, significantly improving F1-scores for six classes and slightly improving one additional class. These results demonstrate that homogeneous stacking with XGBoost achieves better predictions than Thockchom et al.'s heterogeneous stacking. Additionally, our approach effectively addresses class imbalance without resorting to resampling techniques, avoiding overfitting or information loss.

In summary, this research contributes significantly to the field by introducing a novel ensemble learning approach that effectively addresses imbalanced classification without resampling. The combination of hyperparameter tuning, XGBoost, and stacking shows superior performance, greatly improving network attack detection and offering a robust solution for similar challenges across various domains. This methodology can be adapted to other areas with imbalanced multi-class datasets, paving the way for future research and the broader application of ensemble learning techniques, underscoring its impact in developing effective classification models.

Acknowledgment

The author would like to acknowledge the support from the Fundamental Research Grant Scheme (FRGS) under a grant number of FRGS/1/2018/ICT02/UNIMAP/02/6 from the Ministry of Higher Education Malaysia.

References

[1] I. Lella, C. Ciobanu, E. Tsekmezoglou, M. Theocharidou, E. Magonara, A. Malatras, R. Svetozarov Naydenov *et al.*, "Enisa threat landscape 2023: July 2022 to june 2023," 2023.

[2] K. M. Hasib, M. S. Iqbal, F. M. Shah, J. Al Mahmud, M. H. Popel, M. I. H. Showrov, S. Ahmed, and O. Rahman, "A survey of methods for managing the classification and solution of data imbalance problem," *Journal of Computer Science*, vol. 16, no. 11, p. 1546–1557, Nov. 2020. [Online]. Available: http://dx.doi.org/10.3844/jcssp.2020.1546.1557

[3] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine learning with oversampling and undersampling techniques: Overview study and experimental results," in *2020 11th International Conference on Information and Communication Systems (ICICS)*, 2020, pp. 243–248.

[4] D. Elreedy, A. F. Atiya, and F. Kamalov, "A theoretical distribution analysis of synthetic minority oversampling technique (smote) for imbalanced learning," *Machine Learning*, vol. 113, no. 7, pp. 1–21, 2023.

[5] M. Kamaladevi, V. Venkataraman, and K. Sekar, "Tomek link undersampling with stacked ensemble classifier for imbalanced data classification," *Annals of the Romanian Society for Cell Biology*, pp. 2182–2190, 2021.

[6] Chongomweru Halimu and Asem Kasem, "A novel ensemble method for classification in imbalanced datasets using split balancing technique based on instance hardness (sBal_ih)," *Neural Computing and Applications*, pp. 1–22, Jan. 2021.

[7] Chih-Fong Tsai and Wei-Chao Lin, "Feature Selection and Ensemble Learning Techniques in One-Class Classifiers: An Empirical Study of Two-Class Imbalanced Datasets," *IEEE Access*, vol. 9, pp. 13 717–13 726, 2021.

[8] Hongle Du, Yan Zhang, Kee-Rae Gang, Lin Zhang, and Yeh-Cheng Chen, "Online ensemble learning algorithm for imbalanced data stream," *Appl. Soft Comput.*, vol. 107, p. 107378, Aug. 2021.

[9] M. Ali, M.-u. Haque, M. H. Durad, A. Usman, S. M. Mohsin, H. Mujlid, and C. Maple, "Effective network intrusion detection using stacking-based ensemble approach," *Int. J. Inf. Secur.*, vol. 22, no. 6, p. 1781–1798, jul 2023. [Online]. Available: https://doi.org/10.1007/s10207-023-00718-7

[10] H. Rajadurai and U. D. Gandhi, "A stacked ensemble learning model for intrusion detection in wireless network," *Neural Computing and Applications*, vol. 34, no. 18, pp. 15 387–15 395, Sep. 2022. [Online]. Available: https://link.springer.com/10.1007/s00521-020-04986-5

[11] N. Thockchom, M. M. Singh, and U. Nandi, "A novel ensemble learning-based model for network intrusion detection," *Complex & Intelligent Systems*, vol. 9, no. 5, pp. 5693–5714, 2023.

[12] H. F. Soon, A. Amir, H. Nishizaki, N. A. H. Zahri, L. M. Kamarudin, and S. N. Azemi, "Evaluating tree-based ensemble strategies for imbalanced network attack classification," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 1, 2024. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2024.01501111

[13] M. H. Kabir, M. S. Rajib, A. S. M. T. Rahman, M. M. Rahman, and S. K. Dey, "Network intrusion detection using unsw-nb15 dataset: stacking machine learning based approach," in *2022 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)*. IEEE, 2022, pp. 1–6.

[14] V. Sidharth and C. Kavitha, "Network intrusion detection system using stacking and boosting ensemble methods," in *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*. IEEE, 2021, pp. 357–363.

[15] R. Zhao, Y. Mu, L. Zou, and X. Wen, "A hybrid intrusion detection system based on feature selection and weighted stacking classifier," *IEEE Access*, vol. 10, pp. 71 414–71 426, 2022.

[16] T. Tao, Y. Liu, Y. Qiao, L. Gao, J. Lu, C. Zhang, and Y. Wang, "Wind turbine blade icing diagnosis using hybrid features and stacked-xgboost algorithm," *Renewable Energy*, vol. 180, pp. 1004–1013, 2021.

[17] M. Zivkovic, L. Jovanovic, M. Ivanovic, N. Bacanin, I. Strumberger, and P. M. Joseph, "Xgboost hyperparameters tuning by fitness-dependent optimizer for network intrusion detection," in *Communication and intelligent systems: Proceedings of ICCIS 2021*. Springer, 2022, pp. 947–962.

[18] N. AlHosni, L. Jovanovic, M. Antonijevic, M. Bukumira, M. Zivkovic, I. Strumberger, J. P. Mani, and N. Bacanin, "The xgboost model for network intrusion detection boosted by enhanced sine cosine algorithm," in *International Conference on Image Processing and Capsule Networks*. Springer, 2022, pp. 213–228.

[19] M. Zivkovic, M. Tair, K. Venkatachalam, N. Bacanin, Š. Hubálovskỳ, and P. Trojovskỳ, "Novel hybrid firefly algorithm: An application to enhance xgboost tuning for intrusion detection classification," *PeerJ Computer Science*, vol. 8, p. e956, 2022.

[20] X. Yong and Y. Gao, "Hybrid firefly and black hole algorithm designed for xgboost tuning problem: An application for intrusion detection," *IEEE Access*, vol. 11, pp. 28 551–28 564, 2023.

[21] N. Bacanin, A. Petrovic, M. Antonijevic, M. Zivkovic, M. Sarac, E. Tuba, and I. Strumberger, "Intrusion detection by xgboost model tuned by improved social network search algorithm," in *International Conference on Modelling and Development of Intelligent Systems*. Springer, 2022, pp. 104–121.

[22] M. Rashid, J. Kamruzzaman, T. Imam, S. Wibowo, and S. Gordon, "A tree-based stacking ensemble technique with feature selection for network intrusion detection," *Applied Intelligence*, vol. 52, no. 9, pp. 9768–9781, 2022.

[23] X. Zheng, Y. Wang, L. Jia, D. Xiong, and J. Qiang, "Network intrusion detection model based on chi-square test and stacking approach," in *2020 7th International Conference on Information Science and Control Engineering (ICISCE)*. IEEE, 2020, pp. 894–899.

[24] M. R. Ghazi and N. Raghava, "A scalable and stacked ensemble approach to improve intrusion detection in clouds," *Information Technology and Control*, vol. 52, no. 4, pp. 898–914, 2023.

[25] Hossein Ghaderi Zefrehi and H. Altınçay, "Imbalance learning using heterogeneous ensembles," *Expert Syst. Appl.*, vol. 142, Mar. 2020.

[26] Roshani Choudhary and Sanyam Shukla, "A clustering based ensemble of weighted kernelized extreme learning machine for class imbalance learning," *Expert Syst. Appl.*, vol. 164, p. 114041, Feb. 2021.

[27] Zonghai Zhu, Zhe Wang, Dongdong Li, Yujin Zhu, and W. Du, "Geometric Structural Ensemble Learning for Imbalanced Problems," *IEEE Transactions on Cybernetics*, vol. 50, pp. 1617–1629, Apr. 2020.

[28] Zhi Chen, Jiang Duan, Li Kang, and G. Qiu, "A hybrid data-level ensemble to enable learning from highly imbalanced dataset," *Inf. Sci.*, vol. 554, pp. 157–176, Apr. 2021.

[29] Kaixiang Yang, Zhiwen Yu, Xin Wen, Wenming Cao, C. L. P. Chen, H. Wong, and J. You, "Hybrid Classifier Ensemble for Imbalanced Data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, pp. 1387–1400, Apr. 2020.

[30] Yinan Guo, Yaoqi Chu, Botao Jiao, Jian-Bo Cheng, Zekuan Yu, Ning Cui, and Lianbo Ma, "Evolutionary Dual-Ensemble Class Imbalance Learning for Human Activity Recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, pp. 728–739, Aug. 2022.

[31] Na Liu, Xiaomei Li, Ershi Qi, Man Xu, Ling Li, and Bo Gao, "A Novel Ensemble Learning Paradigm for Medical Diagnosis With Imbalanced Data," *IEEE Access*, vol. 8, pp. 171 263–171 280, 2020.

[32] Enas Elgeldawi, Awny Sayed, Ahmed R. Galal, and Alaa M. Zaki, "Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis," *Informatics*, vol. 8, p. 79, Nov. 2021.

[33] Farkhanda Abbas, Feng Zhang, Muhammad Ismail, G. Khan, Javed Iqbal, A. Alrefaei, and M. Albeshr, "Optimizing Machine Learning Algorithms for Landslide Susceptibility Mapping along the Karakoram Highway, Gilgit Baltistan, Pakistan: A Comparative Study of Baseline, Bayesian, and Metaheuristic Hyperparameter Optimization Techniques," *Sensors (Basel, Switzerland)*, vol. 23, Aug. 2023.

[34] Ismail Damilola Raji, H. Bello-Salau, I. J. Umoh, A. Onumanyi, M. Adegboye, and Ahmed Tijani Salawudeen, "Simple Deterministic Selection-Based Genetic Algorithm for Hyperparameter Tuning of Machine Learning Models," *Applied Sciences*, Jan. 2022.

[35] Maryam Karimi Mamaghan, Mehrdad Mohammadi, P. Meyer, Amir Mohammad Karimi-Mamaghan, and El-Ghazali Talbi, "Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art," *Eur. J. Oper. Res.*, vol. 296, pp. 393–422, Apr. 2021.

[36] N.-D. Hoang, V.-D. Tran, and X.-L. Tran, "Predicting compressive strength of high-performance concrete using hybridization of nature-inspired metaheuristic and gradient boosting machine," *Mathematics*, vol. 12, no. 8, p. 1267, 2024.

[37] B. Xi, Z. Huang, S. Al-Obaidi, and L. Ferrara, "Predicting ultra high-performance concrete self-healing performance using hybrid models based on metaheuristic optimization techniques," *Construction and Building Materials*, vol. 381, p. 131261, 2023.

[38] S. Zhao, Y. Xiang, L. Wu, X. Liu, J. Dong, F. Zhang, Z. Li, and Y. Cui, "Simulation of diffuse solar radiation with tree-based evolutionary hybrid models and satellite data," *Remote Sensing*, vol. 15, no. 7, p. 1885, 2023.

[39] A. Rusli, A. Suryadibrata, S. B. Nusantara, and J. C. Young, "A comparison of traditional machine learning approaches for supervised feedback classification in bahasa indonesia," *IJNMT (International Journal of New Media Technology)*, vol. 7, no. 1, pp. 28–32, 2020.

[40] Nurshazlyn M. Aszemi and P. Dominic, "Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms," *International Journal of Advanced Computer Science and Applications*, 2019.

[41] A. K. Agrawal and G. Chakraborty, "On the use of acquisition function-based Bayesian optimization method to efficiently tune SVM hyperparameters for structural damage detection," *Structural Control and Health Monitoring*, vol. 28, Jan. 2021.

[42] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation*, J. Durand-Lose and N. Jonoska, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 240–249.

[43] T.-T. Nguyen, J.-S. Pan, and T.-K. Dao, "An improved flower pollination algorithm for optimizing layouts of nodes in wireless sensor network," *IEEE Access*, vol. 7, pp. 75 985–75 998, 2019.

[44] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992. [Online]. Available: http://www.jstor.org/stable/24939139

[45] G. D'Angelo and F. Palmieri, "Gga: A modified genetic algorithm with gradient-based local search for solving constrained optimization problems," *Inf. Sci.*, vol. 547, pp. 136–162, 2021.

[46] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 2009, pp. 210–214.

[47] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99 129–99 149, 2022.

[48] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[49] L. Chao, Z. Wen-Hui, L. Ran, W. Jun-Yi, and L. Ji-Ming, "Research on star/galaxy classification based on stacking ensemble learning," *Chinese Astronomy and Astrophysics*, vol. 44, no. 3, pp. 345–355, 2020.

[50] D. Burka, C. Puppe, L. Szepesváry, and A. Tasnádi, "Voting: A machine learning approach," *European Journal of Operational Research*, vol. 299, no. 3, pp. 1003–1017, 2022.

[51] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016.

[52] N. Van Thieu and S. Mirjalili, "Mealpy: An open-source library for latest meta-heuristic algorithms in python," *Journal of Systems Architecture*, 2023.