ENSEMBLE BASED NETWORK INTRUSION DETECTION SYSTEM (NIDS)

Mona A. Kamel
Computer Science and Engineering Department
The American University in Cairo
Cairo, Egypt
mona_ahmad@aucegypt.edu

Mohamed H. Seddik
Computer Science and Engineering Department
The American University in Cairo
Cairo, Egypt
mohamedheshamm@aucegypt.edu

Mohammed U. Elkholy
Computer Science and Engineering Department
The American University in Cairo
Cairo, Egypt
moelkholy@aucegypt.edu

Nada M. Moursi
Computer Science and Engineering Department
The American University in Cairo
Cairo, Egypt
nmoursi@aucegypt.edu

Mohamed A. SharafEldein
Computer Science and Engineering Department
The American University in Cairo
Cairo, Egypt
mohamedsharaf@aucegypt.edu

Kareem A. Talaat
Computer Science and Engineering Department
The American University in Cairo
Cairo, Egypt
kareemamr213@aucegypt.edu

Mohamed Sedky
Computer Science and Engineering Department
Cairo, Egypt
mohamed.sedky@aucegypt.edu

Hossam Sharara
Computer Science and Engineering Department
Cairo, Egypt
hossam.sharara@aucegypt.edu

ABSTRACT

In modern times, the proliferation of data, particularly sensitive information, has highlighted the extreme importance for robust cybersecurity measures. Any leakage of sensitive data can have severe consequences, whether financial or legal. This growing importance of safety has made the cybersecurity field more important than ever. Concurrently, advances in Machine Learning (ML), improvements in computational capabilities, and the complex nature of cyber attacks have made the applicability of ML techniques in the cybersecurity field an attractive area for research, particularly Intrusion Detection. In this paper, we discuss the current limitations in this area of research, potential solutions, and provide a novel Network Intrusion Detection System (NIDS) that utilizes ensemble learning to achieve results that rival or exceed other state-of-the-art models.

Keywords Ensemble Learning · Feature Engineering · Anomaly

1 Introduction

Intrusions, like unauthorized access and malicious activities, carry risks for sensitive information and services. Detecting such intrusions accurately and timely is extremely crucial. Traditional NIDSs are often rule-based, relying on known sequences and patterns that match specific attack signatures. Setting up such systems requires expert domain knowledge. Additionally, patterns for certain types of attacks may be complex or subtle, thus making it hard to match them properly.

Machine Learning and Deep Learning (DL) research have gained a lot of traction as of late due to drastic improvements in computational power. Advances in these fields have allowed many other fields to advance as well by utilizing techniques from ML and DL to automate or solve tasks that were very tedious or previously infeasible. Many of these techniques are able to find patterns in extremely high dimensional data that are otherwise not noticeable to human beings.

Machine Learning and Deep Learning have been extensively applied as of late to tackle the intrusion detection problem. However, they still leave a lot to be desired. Publicly available datasets for network intrusion detection are mostly synthetic, and thus might not be very reflective of real-life. Nonetheless, many ML- and DL-based systems have been proposed using these datasets for training, with varying degrees of success. In this paper, we discuss the limitations that many of these proposed systems face. We additionally tackle the different datasets from a cybersecurity perspective, allowing for better informed feature selection. We finally propose a novel NIDS that utilizes Ensemble Learning to tackle some of the problems addressed.

2 Literature Review

2.1 Network Intrusion Detection Systems

Network-based Intrusion Detection Systems (NIDS) proactively observe network traffic, looking for suspicious activities like unauthorized access attempts or network scans. They can be placed at various locations within a network, such as its boundary, internal points, vital junctures like servers and routers. There are different types of NIDSs based on the mechanism they use to detect intrusions or malicious activities. Some of them are highlighted below.

2.1.1 Rule-based Detection

These NIDSs utilize known patterns or signatures for certain intrusions/attacks to detect them. These systems are very systematic in the way they operate. They actively monitor data flowing through the network, trying to match the data to predefined rules in a rules database. If a match is found, an alert is generated for the network administrator [1, 2].

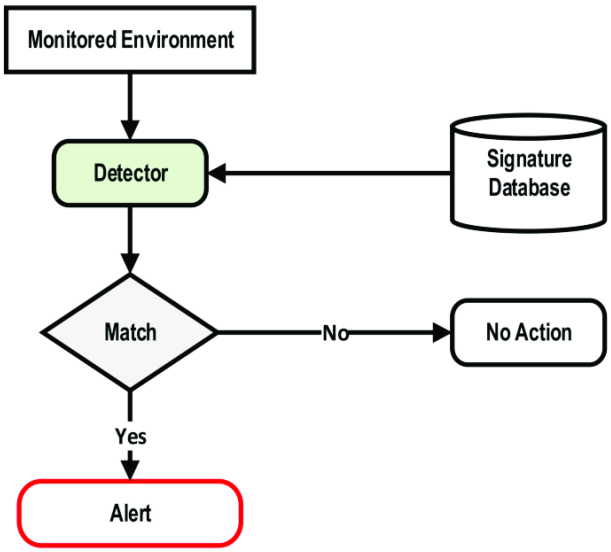

Figure 1: Work Flow of Rule-Based NIDS.
[3]

According to [4], rule-based detection has several advantages. They are often very easy to implement and use, and if the rules are well-defined and match signatures, then there is a low chance of having false positives. However, it still faces some challenges: it can't handle unknown attacks of unknown signatures. Additionally, attacks inside networks that involve privilege abuse are often hard to detect using it. In addition, rule-based NIDS shows a huge drawback as they are ineffective in detecting zero-day attacks. These attacks exploit previously unknown vulnerabilities, for which no detection rules exist. Thus, rule-based detection requires frequent updates to remain effective. Which would demand significant resources in terms of time and expertise. Finally, due to the frequent network changes like configurations, applications, and user behavior. Rule-based with their static nature may not adapt well to these changes, increasing the rate of false positives/negatives or failing to detect an attack as the system no longer aligns with the network.

Snort [5] is an open-source rule-based IDS that can be used for both network intrusion detection and prevention. Snort uses a combination of signature-based and anomaly-based detection methods to identify security threats, making it a flexible and efficient IDS. Snort is highly configurable, and its rules can be updated in real-time, allowing it to adapt to new and evolving threats.

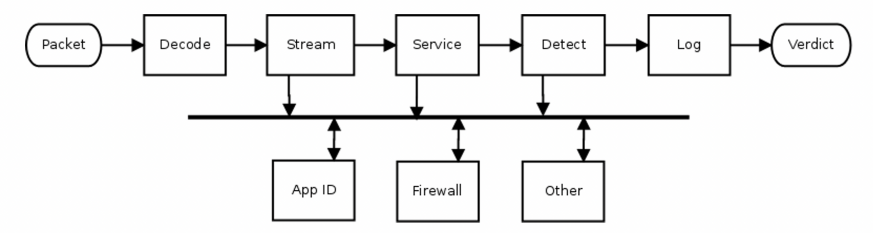

Figure 2: Work Flow of Snort 3. [6]

2.1.2 Anomaly-based Detection

Anomaly-based detection relies on modeling the normal behaviors of the system or network it operates on, and identifies any behaviors that deviate from those. This makes them a very suitable candidate for detecting zero-day attacks (new attacks/ malware) as such attacks are often trying to exploit newly found vulnerabilities in systems that haven't been exploited before. This however poses a problem, any behavior that deviates from the normal behavior of the system could be flagged as malicious, even if that was not the case. This is evident in the high number of false positives that such systems usually output[7]. Anomaly-based detection methods can be further classified according to the type of algorithms they use for anomaly detection, however, we will mainly focus on Machine Learning and Deep Learning techniques in our review. These techniques allow for a more accurate detection as the network evolves. Anomaly-based detection is particularly useful in environments where security needs are high, and where there is a potential for sophisticated, unknown attacks. It's often used in conjunction with other types of IDS, like signature-based systems, to provide a more comprehensive security posture.

2.2 Classical Machine Learning Approaches

Machine Learning approaches used in NIDSs differ greatly. They can be classified as follows into two main categories: Classical Machine Learning vs Deep Learning techniques. Both classes of algorithms have been used extensively for both anomaly-based NIDSs. We will be mainly focusing on classification-based NIDSs. Moreover, two of the selected state-of-the-art models were we conducted our analysis, Model multi-tiered hybrid intrusion detection system (MTH) [8] and Model Leader Class and Confidence Decision Ensemble (LCCDE) [9]. The LCCDE (Leader Class and Confidence Decision Ensemble) model, developed by Kiflay et al. represents a cutting-edge approach in Network Intrusion Detection Systems (NIDS) for Internet of Vehicles (IoV). This system uniquely combines three sophisticated machine learning algorithms: XGBoost, LightGBM, and CatBoost, each known for their gradient-boosting abilities. The LCCDE operates in two key phases: training and prediction. During training, the algorithms are applied to IoV traffic datasets to create 'leader models' for different attack classes. In the prediction phase, these models, along with their prediction confidence levels, are utilized to detect attacks accurately. The framework leverages a soft voting mechanism based on class probabilities, harmonizing the outputs of individual models to optimize overall attack detection.

The proposed Multi-tiered Hybrid Intrusion Detection System (MTH-IDS) for the Internet of Vehicles (IoV) combines various advanced techniques for comprehensive intrusion detection. It utilizes four supervised learners, including decision tree, random forest, extra trees, and XGBoost, for detecting known cyber attacks. These models are optimized using a stacking ensemble model and Bayesian optimization with a tree Parzen estimator method, focusing on

maximizing their detection efficiency. For unknown or zero-day attacks, the system employs an innovative anomaly-based IDS, using a cluster labeling k-means model to classify attacks based on data clustering and majority labeling. Additionally, the system incorporates Bayesian optimization and biased classifiers to refine the unsupervised learner, aiming to minimize classification errors. This integrated approach ensures robust detection capabilities for both known and novel threats in IoV environments.

2.2.1 Deep Learning Approaches

Deep Learning techniques have been used in different capacities in NIDSs. Sommer and Paxson's work highlights these challenges, which include the variability and diversity of network traffic, yet a lack of reliable public IDS datasets, with the evolving nature of cyber threats.

zVinayakumar et al. (2019) highlighted the effectiveness of recurrent neural networks (RNNs), especially Long Short-Term Memory (LSTM) networks which is used for analyzing sequential network data [5]. This approach underscored the potential of RNNs in handling the complexities of network intrusion data. However, the persistent issue of class imbalance in intrusion detection datasets has often skewed model performances.

To address this, Cao et al.'s model [10], integrating the capabilities of LSTM and Gated Recurrent Units (GRU) for network intrusion detection. The LSTM layers are instrumental in processing and understanding the sequence-based nature of network data.Then, the data is fed into a GRU layer, renowned for its ability to capture temporal dependencies effectively. This combination is particularly potent in the context of network intrusion detection, where understanding the sequence and temporal patterns in data is crucial.

Furthermore, we focus on balancing the dataset through data resampling techniques. This ensures that the training process is not biased towards the majority class, typically the normal traffic in intrusion detection datasets. A balanced training approach mitigates the risk of model skewness and enhances its ability to detect minority class instances effectively. Moreover, in the quest for robust and practical intrusion detection systems, there has been a consistent reliance on well-established machine learning models such as XGBoost, Decision Trees, and Random Forests. These models have gained widespread acceptance due to their interpretability, scalability, and relative ease of implementation. They are particularly valued for their ability to unravel complex data patterns, making them a staple in various industries for tasks including, but not limited to, intrusion detection.

2.3 Ensemble Learning

Ensemble models focus on combining individual meta-learing technique, these techniques include voting, stacking, bagging, and boosting. This section presents an overview of state-of-the-art methods that adopt ensemble learning. Xiao Et al. [11] demonstrated a new strategy which applied deep learning to an ensemble approach that combined multiple machine learning models. This deep learning model combines five classification models including SVMS, KNN, DT, RF, and gradient boosting decision trees. This model was used to increase the accuracy of detection cancer and in tumor conditions. The results indicated that the model had better results and the prediction got increased compared to using a single classification. Furthermore, Kiflay Et al. [12] proposed an innovative Network Intrusion Detection System (NIDS) utilizing a collective machine learning approach to distinguish between benign and malicious network traffic. they tested this ensemble-based NIDS with the NSL-KDD and UNSW-NB15 IDS datasets for binary traffic categorization, considering all available traffic attributes.

Their model adopts an ensemble machine learning approach to classify network traffic as benign or attack. This NIDS consists of three main components: the Packet Capture Unit (PCU) for traffic collection, the Data Processing Unit (DPU) for data conversion and normalization, and the Classifier Engine (CE) which is the core of their system. The CE uses various machine learning classifiers such as random forest, AdaBoost and gradient boosting decision tree (GBDT). A key feature of their system is the incorporation of a soft voting classifier, which integrates the outputs from individual classifiers based on class probabilities, thereby enhancing detection accuracy and reducing false alarms. The model's effectiveness was demonstrated using the NSL-KDD and UNSW-NB15 datasets, showing significant potential in accurately detecting cyber-attacks. Future expansions of this research involve testing the NIDS with a select set of traffic attributes and for multi-class attack classification. The outcomes indicate that our NIDS holds promise in enhancing the precision of cyber-attack identification and in reducing the frequency of false alerts.

Jakka et al. proposed a bagging ensemble, using the KDD'99 dataset, which is short for bootstrap aggregating. This ensemble technique generates multiple versions of a predictor and uses these to get an aggregated predictor. It was implemented using multiple sets of classifiers and tree-based algorithms, where each classifier is trained independently on a random subset of data, and all the predictions from these classifiers are combined through a majority voting system for classification.[13] This proposed method helps reduce over-fitting and improves the stability and accuracy of the model. In addition, the other proposed model is a stacked ensemble model, The proposed model combines

different classification models at the base level, and it references the use of Support Vector Machine (SVM) and Bayes Tree models. The final proposed model Spectral Clustering and Deep Neural Network (SCDNN), uses deep learning techniques in the stacking method. This stacking model leverages the diversity of different models to which it has achieved a very high weighted accuracy of 97.5%.

2.4 Research Gaps

2.4.1 Datasets

Advancements in Machine Learning (ML) for Network Intrusion Detection Systems (NIDS) have been hampered by the lack of uniform feature sets across different datasets, making it challenging to evaluate the generalizability of ML models across varied network environments and attack scenarios. Sarhan et al. addressed this limitation by transforming existing benchmark NIDS datasets - UNSW-NB15, BoT-IoT, ToN-IoT, and CSE-CIC-IDS2018 - into a standardized NetFlow format. This transformation resulted in five datasets, now available to the research community, which are suitable for both binary and multi-class traffic classification investigations. The standardization of these datasets into a common NetFlow-based format offers three key advantages. Firstly, it reduces the complexity and resource demands for data collection and storage. Secondly, it enables the consistent evaluation of proposed ML models across various datasets, ensuring comparability and robustness of the models. Thirdly, it allows for the amalgamation of different datasets, creating a richer and more diverse data source for training and assessing ML-based NIDS. Sarhan et al. also explored the preprocessing methods in NIDS, recommending the removal of specific columns like IPV4_SRC_ADDR, L4_SRC_PORT, IPV4_DST _ADDR, and L4_DST _PORT to mitigate biases and employing min-max normalization to treat categorical variables numerically. This approach, while somewhat controversial, aims to refine data for more accurate threat detection [14].

Furthermore, in their evaluation using an Extra Tree classifier, they demonstrated that the extended NetFlow-based feature set with 43 features outperforms proprietary feature sets in classification performance for all the benchmark NIDS datasets in binary and multi-class scenarios [15]. This finding underscores the practicality and scalability of NetFlow exporters and collectors, confirming the real-world applicability of the NetFlow-based feature sets.

Finally, we tried to adhere to the preprocessing recommendations by Sarhan et al. (2021), who advocated for a standard feature set in network intrusion detection datasets [14]. This standardization is important in removing irrelevant features, reducing data dimensionality, and finally improving the detection capability of the model. The normalization of selected features ensures uniformity in data scale, enhancing the model's ability to process and analyze data effectively.

2.4.2 Evaluation Metrics

The confusion matrix is a commonly used table when evaluating supervised machine learning models. A confusion matrix is a table that displays information about predicted and actual classes. This helps us calculate some metrics to evaluate the model's performance using some terms. Confusion matrix terms include:

Term	**Description**
True Positive (TP)	The data instances correctly predicted to be positive.
False Negative (FN)	The data instances wrongly predicted to be negative.
True Negative (TN)	The data instances correctly predicted to be negative.
False Positive (FP)	The data instances wrongly predicted to be positive.

To assess the different models, we chose precision, recall, and F1-score as evaluation metrics. Metrics can be deceptive, we discuss below the rational for choosing certain types of them over others:

- **Accuracy**

  Accuracy is one of the most used metrics in the evaluation of models; that being said, we chose not to rely on accuracy as one of the metrics in model evaluation because of the following reasons:

  - Imbalanced Datasets: most datasets are imbalanced, as the occurrence of intrusions (positive class) is often significantly lower than normal network behavior (negative class). Accuracy can be misleading because a model might achieve high accuracy by predicting the majority class all the time. This is problematic because it may not effectively capture the model's ability to detect intrusions.
  - Misleading High Accuracy: models can achieve high accuracy even when they perform poorly in the minority class. If the primary concern is detecting intrusions, a high accuracy score might give a false

sense of confidence in the model's performance, and a crucial part of this research paper is to fill the gap in the literature, which is taking into account minority attack detection.

- **Precision**

  Precision is the ratio of true positive predictions to the total number of positive predictions (true positives + false positives). We are choosing precision as one of the evaluation metrics: Precision is crucial when the cost of false positives (misclassifying normal behavior as an intrusion) is high. In NIDS, false positives can lead to unnecessary alerts or actions, making precision a critical metric for assessing the model's ability to provide accurate and reliable intrusion alerts.

- **Recall**

  Recall is the ratio of true positive predictions to the total number of actual positive instances (true positives + false negatives). The reason we are choosing recall as one of the evaluation metrics is that recall is essential when the cost of false negatives (missing actual intrusions) is high. In other words, recall shows the model's ability to detect even minority attacks (which is crucial in our case). In NIDS, failing to detect all types of intrusions can have severe consequences, so a high recall indicates that the model is effective in capturing the majority of intrusions.

- **F1-score**

  F1-score is the harmonic mean of precision and recall. It provides a balanced measure that considers both false positives and false negatives. The reason we are choosing F1-score as one of our evaluation metrics is that the F1-score is useful when there is a need to strike a balance between precision and recall. It becomes especially relevant in situations where there is an inherent trade-off between minimizing false positives and false negatives.

On top of using those three specific metrics, one can either choose macro precision, recall, and f1-score or weighted precision, recall, and f1-score. To understand why one would be better than the other, a comparison between weighted and macro metrics is needed.

2.4.3 Macro Metrics (Precision, Recall, F1-Score)

Macro metrics are computed by calculating the metric independently for each class and then taking the average. This treats all classes equally, regardless of their frequency in the dataset. Macro metrics are particularly useful in datasets with class imbalances. They highlight the performance of minority classes. By treating each class equally, macro metrics provide a clearer picture of a model's performance across all classes, including those that are underrepresented. Macro metrics are crucial for applications where detecting minority classes is as important as majority ones. For example, in fraud detection or rare disease diagnosis, failing to correctly identify the rare cases (minority class) could be very costly.

2.4.4 Weighted Metrics (Precision, Recall, F1-Score)

Weighted metrics are calculated by taking the average of the metrics for each class, weighted by the number of true instances for each class. This approach takes class imbalance into account. Weighted metrics are suitable for datasets where class distribution is not balanced, and the focus is on overall performance, taking into account the prevalence of each class. These metrics give a more accurate view of the model's performance in terms of the most common classes. They can provide a false sense of high performance if the model is only good at predicting the majority class while failing at minority classes. In cases where minority class detection is critical, weighted metrics might be misleading. They might indicate good performance overall, even though the model performs poorly on less frequent but important classes.

2.4.5 Choosing Macro vs Weighted Metrics

Macro metrics should be chosen when the ability of a model to detect minority classes is crucial. They ensure that the performance of smaller classes is not overshadowed by the larger ones. This is vital in scenarios where missing out on rare but significant events (like rare attacks in cybersecurity) can have serious consequences. Weighted metrics, on

the other hand, are more appropriate when the focus is on overall performance, especially in situations where class frequencies are representative of their importance or impact.

Furthermore, choosing macro precision, recall, and f1-score as evaluation metrics is particularly valuable in the context of imbalanced datasets, which are common in intrusion detection. Since the occurrence of intrusions is often much lower than normal network behavior, these metrics provide a more accurate representation of the model's performance, especially concerning the positive class. While precision, recall, and F1-score each focus on specific aspects of model performance, considering all three metrics provides a comprehensive understanding of the model's strengths and weaknesses. Choosing macro precision, recall, and f1-score is essential to evaluate a model's performance in depth and to envisage the model's ability to detect not only major attacks but also less frequent but highly impactful attacks. This holistic assessment is essential for making informed decisions about the model's suitability for real-world deployment.

3 Methodology

We propose an ensemble-based model to enhance the predictive accuracy and handle the imbalance problem. Instead of relying on the outputs of a single model, ensemble methods leverage the diversity among multiple models to improve overall performance. By combining diverse models, ensemble methods compensate for the individual weaknesses of each model, resulting in more accurate and reliable predictions. Common ensemble techniques include bagging, boosting, and stacking. We will be using stacking.

3.1 Proposed Ensemble Methods

Stacking is an ensemble learning technique that combines the predictions of multiple base models to create a meta-model for making final predictions. The process involves training diverse base models independently on a given dataset, generating predictions for each instance. These predictions, together with the dataset itself, become the input features for a meta-model, which is trained to learn how to effectively combine or weigh the predictions of the base models. The idea is that by leveraging the strengths of different models, stacking can capture a broader range of patterns and relationships present in the data, leading to improved predictive performance.

3.2 Binary Classifiers

Individual binary classifiers focus on specific classes, mitigating challenges posed by imbalanced datasets. Each classifier is tailored to recognize patterns associated with a particular class, improving sensitivity. For each of the fifteen classes, we create a binary classifier. In each binary classifier, one class is treated as the positive class (labeled as 1), and the rest of the classes are treated as the negative class (labeled as 0). Then, we train each binary classifier independently, optimizing for recall. This targeted approach allows for more specialized and accurate detection of each attack type. However, some attacks like SQL Injection, might be less frequent (minority class), the Synthetic Minority Over-sampling Technique (SMOTE) is used to oversample these minority classes. This helps in balancing the dataset, ensuring that the classifier is not biased towards the majority class.

4 Performance Evaluation

4.1 Experimental Setup

4.1.1 Dataset

The dataset used was a modified version of the NF-CIC-2018 v2 dataset. To minimize the time spent on computational tasks and to focus on processing crucial attribute information, we needed to convert certain attributes into numerical form, as they were not available in their original dataset format. Additionally, it was imperative to carry out data normalization procedures on the essential data elements. Our research aims to tackle this imbalance and enhance the detection of minority attacks.

Furthermore, Feature selection is crucial in our approach, where we go through our Dataset to pick the most relevant features by cutting through the noise and focusing on the more meaningful features. Sarhan Et al. recognize the need for a standardized approach across various NID datasets and advocate for a NetFlow-based feature set. One significant advancement Sarhan Et al. done was the recognition that the NetFlow feature set, comprising 43 features, is notably more streamlined than the CICFlow 83 features. [14] By considering the cybersecurity domain in feature selection, employing appropriate techniques for handling categorical features, and validating the dataset. These efforts collectively contribute to developing a more robust and accurate network intrusion detection model.

4.1.2 Dataset limitations

The NetFlow datasets, pivotal for ML-based NIDS evolution, deploy the NetFlow network meta-data collection protocol. Originating from datasets like UNSW-NB15, BoT-IoT, ToN-IoT, and CSE-CIC-IDS2018, they aim to standardize NetFlow-based feature sets. However, a pervasive anomaly emerged – the presence of floating-point values in the L7_Proto column. L7_Proto serves as a crucial categorical feature, responsible for identifying application layer protocols such as HTTP, SQL, YouTube, Tor, and more. This problem was found by our methodical analysis of several datasets, including the Netflow-based datasets. As a result, we proactively contacted the dataset authors to request explanations for the unexpected floating-point values, protecting the integrity of the NetFlow datasets.

Understanding the significance of L7_Proto in explaining the network behaviour was a crucial component of the feature selection process. Moreover, it is essential for differentiating between different application layer protocols and provides important information about the type of network traffic. The unforeseen existence of floating-point values in this characteristic casts doubt on the precision of protocol identification and, in turn, the dependability of machine learning-based intrusion detection systems that depend on this data for intrusion detection. The unexpected presence of floating-point values in this feature raises concerns about the accuracy of protocol identification and, consequently, the reliability of ML-based NIDS relying on this information for intrusion detection.

A significant flaw lies in the exclusive reliance on a flow's source and destination IP, coupled with a specific data collection window, for labeling purposes. This strategy categorizes a traffic flow as malicious solely based on the timing of data collection, lacking verification of content and characteristics.

Examination of these attacks revealed a reliance on an established TCP connection for malicious functionality. However, instances primarily consist of flows lacking data transfer in the forward direction, indicating an absence of an actual attack. Bot traffic worsens the issue with numerous failed or empty TCP connections devaluating meaningful botnet traffic. Difficulties arise in dealing with DoS attacks, especially when a web server is brought down. Numerous TCP connections without an HTTP payload are mislabeled as instances of an HTTP-based DoS attack, presenting a misclassification issue.

In conclusion, the labeling strategy play a impmortant role in sole reliance on individual flow analysis is deemed inappropriate for a flow-based NIDS. The inability to discern whether observed failed or empty TCP connections are part of a larger malicious campaign calls for a reevaluation. Recommendations include refining time frames for accurate labeling, augmenting the strategy with content verification within flows, and implementing class- specific adjustments for a more nuanced and accurate NIDS dataset. Addressing these concerns is pivotal for upholding the credibility of NIDS datasets, ensuring precise evaluations and advancing the field of ML- based NIDSs.

4.1.3 Data Preprocessing

Data preprocessing is a crucial phase in the construction of robust machine learning models, particularly in the context of cybersecurity where accurate identification of malicious attacks is paramount. The quality and effectiveness of the preprocessing steps applied to the datasets significantly influence the performance of the subsequent machine learning algorithms. In the prevailing literature, a notable deficiency has been observed regarding the comprehensive documentation of preprocessing methodologies employed by researchers. This gap raises concerns about the replicability and transparency of research findings, hindering the advancement of the field. Recognizing the importance of addressing this gap, our research places a distinct emphasis on elucidating the preprocessing steps taken to refine the NF-CSE-CIC-IDS2018 dataset, thereby contributing to the enhancement of reproducibility and comparability in cybersecurity research.

A fundamental aspect of data preprocessing involves a meticulous examination of dataset columns to identify those whose inclusion may introduce bias or hinder the accurate identification of attacks. In our approach, a well-founded decision was made to exclude the first four columns, namely 'IPV4_SRC_ADDR,' 'L4_SRC_PORT,' 'IPV4_DST_ADDR,' and 'L4_DST_PORT.' The rationale behind this exclusion is rooted in the objective of mitigating potential bias towards identifying attacking and victim end nodes. This knowledge-based exclusion aligns with best practices in cybersecurity research, ensuring that the subsequent machine learning models are trained on a dataset that fosters unbiased and reliable attack detection.

Addressing Categorical Data Challenges The handling of categorical data poses a significant challenge in existing literature, potentially compromising the accuracy of attack identification. Our research addresses this challenge through meticulous preprocessing steps. For instance, the 'L7_PROTO' column, denoting Layer 7 protocol information, was transformed into numerical values to facilitate compatibility with certain machine learning models. Furthermore, the 'L4_DST_PORT' and 'L7_PROTO' columns underwent binning processes to condense categorical values, allowing deep learning models to effectively train on this information.

Specifically, the top 100 ports in 'L4_DST_PORT' and 'L7_PROTO' were identified and binned, with the remaining values categorized as 'other.' This strategic binning not only reduces dimensionality but also ensures that the deep learning models are trained on the most relevant and informative categorical data. The subsequent application of one-hot encoding further enhances the representation of categorical features, enabling the models to capture intricate patterns associated with specific attacks.

In conclusion, our research underscores the significance of transparent and well-documented data preprocessing in cybersecurity research. By meticulously justifying the exclusion of specific columns and implementing strategic binning and encoding techniques, we aim to contribute to the establishment of best practices in the field. These preprocessing steps pave the way for the development of machine learning models that not only exhibit high performance but also foster reproducibility and comparability in cybersecurity research endeavors.

4.2 Experiments

4.2.1 Binary Classifiers

Random forest, XGB Classifier, and LightGBM model were implemented for each attack type. After several trials, while the three models have high metrics values for most attacks, random forest has been proven to be better than the other two in terms of its recall as shown in figure 3. However, SMOTE was then recognized as a main factor that contributes to overfitting the data. Therefore, the balanced random forest model was utilized where the same approach was used with an exception that no undersampling or oversampling takes place. Better results were shown by the balanced random forest, and the overfitting problem was resolved (Figure 4). Consequently, we proceeded building the ensemble model using the balanced random forest binary classifiers.

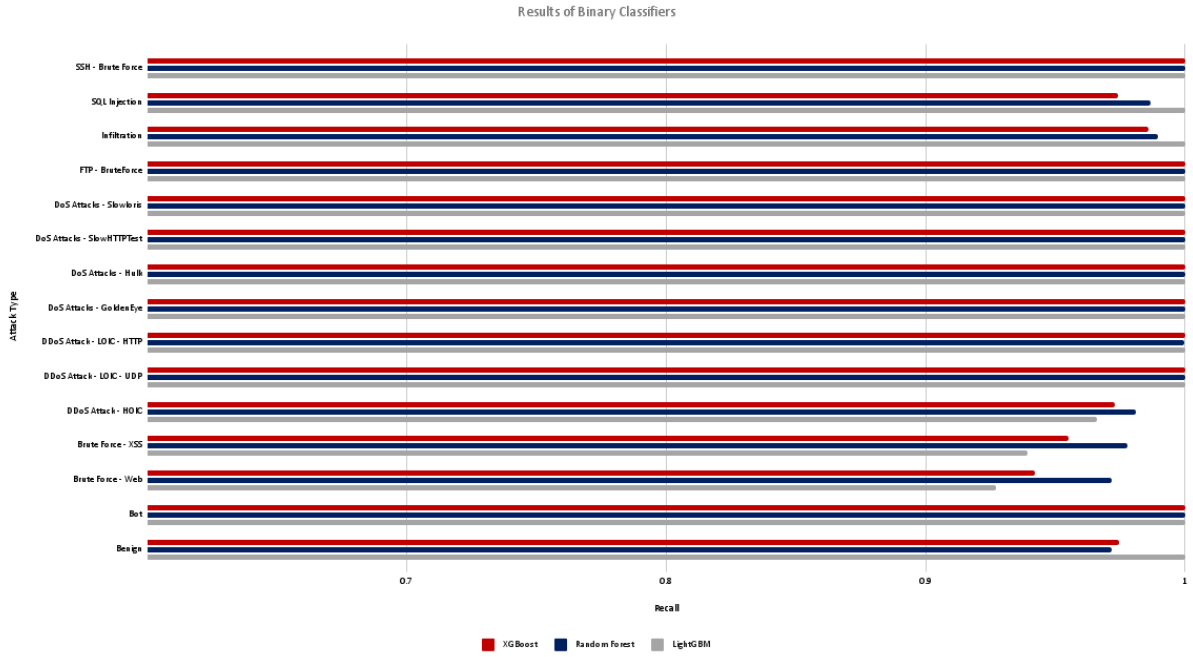

Figure 3: Recall of the Three Models as Binary Classifiers.

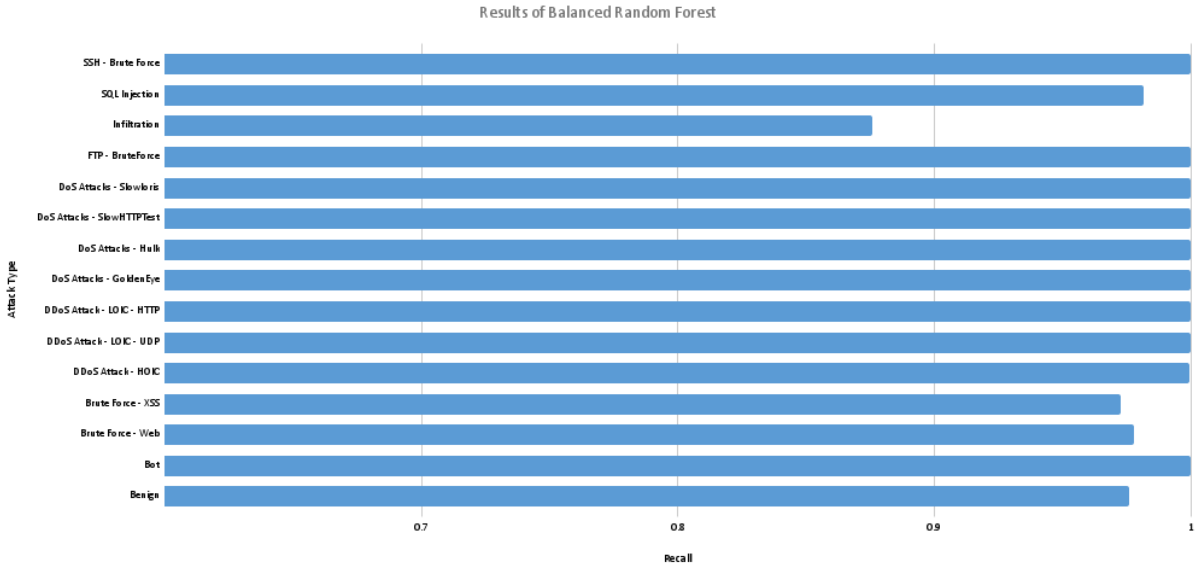

Figure 4: Recall of Balanced Random Forest for different classes.

4.2.2 Sequential Ensemble Model (EnsSeq)

After developing the individual binary classifiers, experiments were developed to opt the best architecture for the proposed ensemble model. One of the proposed architectures is the sequential architecture where one model examines the data at a time. Since the nature of this ensemble is sequential, the order of the binary classifiers is crucial and might lead to the improvement or deterioration of the ensemble model. Furthermore, the running time and memory utilized in this implementation was expected to be extremely high, which was another problem.

Consequently, the distribution of attacks was taken into consideration while integrating the binary classifiers. For example, as shown in figure 5, the benign data as expected is the most dominant, followed by the Hulk DoS attacks. In this case, the models are arranged in the same order of the distribution, starting from the most frequent data to the least frequent records. As shown in figure 6, the processed data will be provided to the benign binary classifier since it represents the majority of the data. The output of the binary classifier can be divided into two main groups: classified data as benign and other data. As for the classified data as benign, they will not be passed into any other classifiers and they are labeled “Benign” in the results of the ensemble model. Regarding the remaining records, they are passed to the second dominant attack which is Hulk DoS attack. The same process applies to the second binary classifiers, and only the data points that are not classified as Hulk DoS attack will be passed to the third model. This process continues until the SlowHTTPTest DoS attack binary classifier is reached and classifies the remaining attacks since it is the least frequent attack type. In case there are still remaining data points after the last binary classifier, they were classified as benign.

Unfortunately, the results were not promising as the macro average was approximately 49.98%. After several discussions, the reason for the inefficiency of the sequential architecture was identified to be due to the propagation of the errors. Each model has its own misclassified data points, whether false positive or false negative with respect to the target attack of the current binary classifier. Those false negative data points will be passed to the next classifier increasing the potential of its misclassification. Similarly, the false positives will halt and will not be passed to the next model increasing the error. For more elaboration, consider the first model which is the benign binary classifier. If a record was wrongly classified as “not benign”, although it is actually benign, it will be passed to the next model and eventually, it will be classified as one of the 14 attacks. On the other hand, if the record was originally any attack data and it was misclassified as benign, it will not be provided as an input to the next model. In that sense, the error of each classifier is passed to the next, causing the accumulation of the error. This has resulted in the deterioration of the ensemble model’s performance with only a recall of 49.98%.

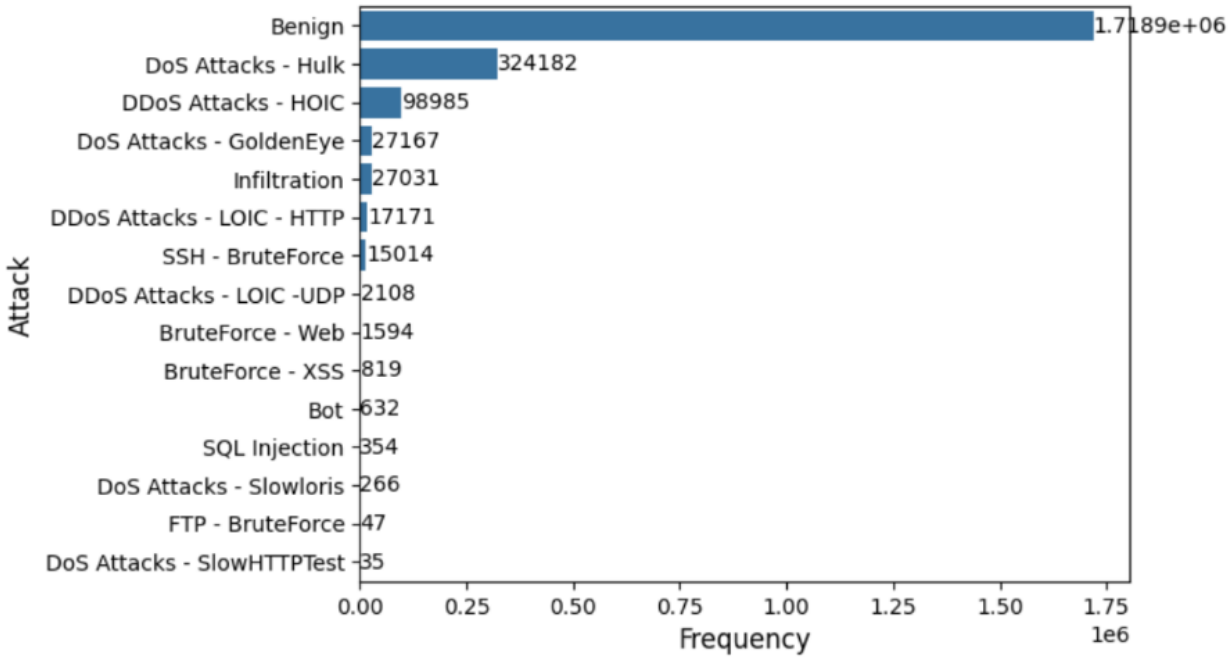

Figure 5: Bar Chart Showing the Distribution of Attacks.

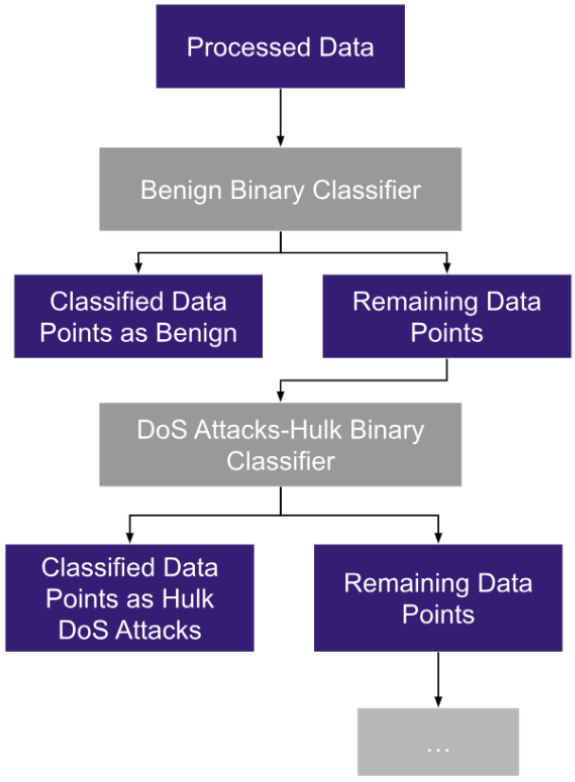

Figure 6: Flowchart of the Sequential Architecture of the Ensemble Model.

4.2.3 Random Forest Ensemble Model (EnsRF)

The proposed ensemble model consists of constructing a new dataset from the current available data. The processed dataset, which consists of 44 features apart from the label of the data points, was combined with the output of the 15 individual binary classifiers to create a new dataset with 59 features. That is, in addition to the original data of a record, this dataset also has the predictions carried out by each of the fifteen binary classifiers with respect to that record. The new dataset is referred to as the BIN_NF dataset. The BIN_NF dataset was then fed to a random forest model with a parameter of 200 trees being implemented in this random forest. An elaborative diagram of the proposed architecture is provided in Figure 7. This ensemble model is referred to as the EnsRF model.

As shown in Table 1, the macro average recall has significantly increased to approximately 0.78. Furthermore, the macro F1-score is approximately 0.77. Therefore,it was concluded that the original dataset along with the binary classifiers outputs provide a more meaningful feature set rather than just the original 44 features.

4.2.4 XGB Ensemble Model (EnsXGB)

The same experiment was carried out using an XGB classifier, which is illustrated in Figure 8, since it is a commonly used model in recent work. Thus, the aim here is to compare the performance of both models. The results are similar to the random forest with the respect to the recall since it produced 78%. However, the XGB model has significantly higher precision as it has 94% and F1-score of 80% approximately. A comparison between the two ensemble models are shown in table 1.

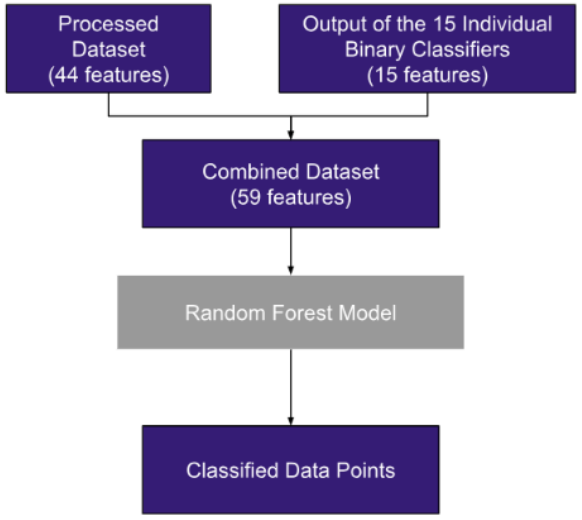

Figure 7: Architecture of the Proposed Ensemble NIDS Model using Random Forests.

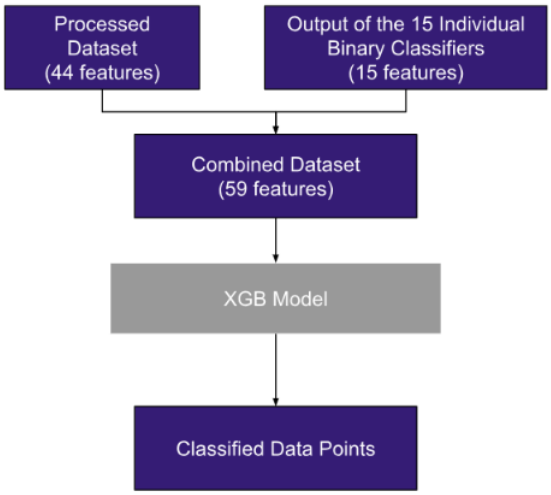

Figure 8: Architecture of the Ensemble Model Using XGB.

4.3 Results and Discussion

4.3.1 Baselines

We compare our proposed models with current state-of-the-art models, along with commonly used machine learning models. MTH [8] and LCCDE [9] models are current state-of-the-art models. For the commonly used ML models, we compare against Decision Trees (Decision Trees-BL), Random Forests (Random Forests-BL). We report the macro precision, recall and F1-scores.

4.3.2 Results and Discussion

Table. 1 contains the results from the baselines and our proposed models. The commonly used ML models show similar performance across all metrics whereas the different variants of LCCDE and MTH show a boost in recall as compared to the other models. This highlights that they are able to detect attacks much better than other models, however low precision suggests that they are not able to classify those attacks properly even if detected. Our proposed EnsXGB

model scores a much higher 94% for precision, with slightly lower recall than LCCDE and MTH variants. This suggests that our model properly classifies attacks when they are detected, lowering the chance of alarm fatigue due to False Positives. ross all classes, but at the cost of slightly lower precision.

4.3.3 Key Insights

Based on these results, we can divide the different models as follows:

- **High Recall Models:** Most models (LightGBM-LCCDE to XGBoost After Stacking-MTH) prioritize recall over precision. This is beneficial in scenarios where failing to detect a true positive (such as a fraudulent transaction in finance) is more costly than dealing with false positives.
- **Balanced Models:** Decision Trees-BL, Random Forest-BL, and EnsRF show a more balanced approach between precision and recall, which can be preferable in situations where both false positives and false negatives have a significant impact.
- **Precision-Focused Model:** The proposed EnsXGB model is distinct in its very high precision, making it an excellent choice in situations where false positives are particularly undesirable, even if it means a slightly lower recall.

	Metrics (Macro)		
Model	**Precision**	**Recall**	**F1-score**
Decision Trees-BL	0.76	0.76	0.76
Random Forest-BL	0.77	0.76	0.77
LightGBM-LCCDE	0.74	0.82	0.75
XGBoost-LCCDE	0.74	0.82	0.76
CatBoost-LCCDE	0.74	0.82	0.75
Ensemble-LCCDE	0.75	0.819	0.75
XGBoost-MTH	0.74	0.82	0.74
Random Forest-MTH	0.74	0.82	0.74
Decision Trees-MTH	0.73	0.82	0.74
Extra Trees	0.74	0.82	0.74
Stacking-MTH	0.74	0.82	0.74
XGBoost After Stacking-MTH	0.74	**0.82**	0.74
EnsRF	0.77	0.78	0.77
EnsXGB	**0.94**	0.78	**0.8**

Table 1: Baseline Models Vs. Proposed Models

5 Conclusion

Our research presents a leap in cybersecurity, particularly in Network Intrusion Detection Systems (NIDS), with the introduction of the EnsXGB model. This model, a result of our focus on ensemble learning as a solution, effectively addresses the limitations of traditional NIDS. Ensemble learning, by integrating multiple algorithms, enhances the model's ability to accurately detect a wide range of intrusions, including minority attacks, which are often overlooked yet critical. EnsXGB, emphasizing precision and effectiveness, excels in identifying genuine threats while minimizing false positives. This aspect is particularly vital in cybersecurity, where accurately pinpointing rare but dangerous attacks is as crucial as avoiding false alarms. Tested rigorously against various models on a consistent dataset, EnsXGB has demonstrated its superiority, underscoring our contribution to advancing NIDS through innovative, ensemble-based machine learning techniques. The EnsXGB model thus stands as a testament to the potential of ensemble learning in creating more secure digital environments adept at detecting even the most elusive threats.

References

[1] Jesús Díaz-Verdejo, Javier Muñoz-Calle, Antonio Estepa Alonso, Rafael Estepa Alonso, and Germán Madinabeitia. On the Detection Capabilities of Signature-Based Intrusion Detection Systems in the Context of Web Attacks. *Appl. Sci.*, 12(2):852, January 2022.

[2] Mohammad Masdari and Hemn Khezri. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. *Applied Soft Computing*, 92:106301, 2020.

[3] Ammar Aldallal and Faisal Alisa. Effective intrusion detection system to secure data in cloud using machine learning. *Symmetry*, 13:1–26, 12 2021.

[4] Vinod Kumar. Signature based intrusion detection system using snort. *International Journal of Computer Applications and Information Technology*, 1:7, 11 2012.

[5] Martin Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX Conference on System Administration*, LISA '99, page 229–238, USA, 1999. USENIX Association.

[6] Snort Rules and IDS Software Download, December 2023. [Online; accessed 6. Dec. 2023].

[7] Anna L. Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.

[8] Li Yang, Abdallah Moubayed, and Abdallah Shami. Mth-ids: A multitiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet of Things Journal*, 9(1):616–632, 2022.

[9] Li Yang, Abdallah Shami, Gary Stevens, and Stephen de Rusett. Lccde: A decision-based ensemble framework for intrusion detection in the internet of vehicles. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 3545–3550, 2022.

[10] Bo Cao, Chenghai Li, Yafei Song, Yueyi Qin, and Chen Chen. Network intrusion detection model based on cnn and gru. *Applied Sciences*, 12(9), 2022.

[11] Xiaodong Feng, Zhi Xiao, Bo Zhong, Jing Qiu, and Yuanxiang Dong. Dynamic ensemble classification for credit scoring using soft probability. *Applied Soft Computing*, 65:139–151, 04 2018.

[12] Aklil Zenebe Kiflay, Athanasios Tsokanos, and Raimund Kirner. A network intrusion detection system using ensemble machine learning. In *2021 International Carnahan Conference on Security Technology (ICCST)*, pages 1–6, 2021.

[13] Geethamanikanta Jakka and Izzat Alsmadi. Ensemble models for intrusion detection system classification. *International Journal of Smart Sensor and Adhoc Network.*, pages 83–95, 01 2022.

[14] Mohanad Sarhan, Siamak Layeghy, Nour Moustafa, and Marius Portmann. *NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems*, pages 117–135. 04 2021.

[15] Mohanad Sarhan, Siamak Layeghy, and Marius Portmann. Towards a standard feature set for network intrusion detection system datasets. *Mob. Netw. Appl.*, 27(1):357–370, feb 2022.