Stacked Based Ensemble Learning Approach for Computer Network Traffic Classification using Data Mining

***Abstract*—In this work, we explore different data sets for Network Intrusion detection. Classify the attack signals from normal ones. With the increase in network connections, computer security, and intrusion detection became a challenging problem. In our research, we used different feature selection techniques: Information gain, Gain ratio, Chi2, and Relief. We train different models based on ten thresholds with a quantization range of 10% to 90% for each feature selection algorithm we used. We train Decision trees, Random Forest, Extra trees, GradiantBoosing, XGBoost, AdaBoostm, Linear, SVM, and Logistic Regression. For each of these models, we use cross-validation and features selected with different thresholds. Then, we compare all of these models based on the validation accuracy obtained from cross-validation. We obtained 15 features selected with the best accuracy of 97.79%. We obtain this model by stacking the best models using the stacking model.**

***Index Terms*—Network Intrusion, Network Security, Stacking Algorithm, Ensemble, Machine Learning**

I. INTRODUCTION

A. Motivation

Most of the other researchers' work focuses on predicting the type of attack more accurately [7; 9; 18; 19; 24], but this step could be the second step for further analysis of classifying the type of attack for a more robust model that could achieve higher security we focus on solving the problem from a binary classification perspective.

Accurately classifying the attack signals is a crucial step before communication is done between users. So, the first step is to determine network intrusion detection as a binary classification before further analysis of the type of these attacks. Dealing, with network intrusion detection as a binary classification problem would enable us to achieve better results in security because this is the most crucial step that needs to be tackled before classifying the type of attack itself.

Improving the accuracy and building a robust Machine Learning model that could effectively classify the attack signals from normal ones is what motivated us to start our work and provide a state-of-the-art approach that could be considered later for more research.

In the last years, the computer network and their security become an important issue because of the increase in people that are using the internet in all fields of life. The security problem becomes a big issue because hackers try to hack users' information such as bank information, e-government information, etc...

The way we work, communicate, operate our businesses and go about our daily lives have changed significantly as a result of the Internet. However, as network connections and services proliferate, network attacks are now a significant threat to human society. A network assault would happen globally every 39 seconds, according to Norton's 2021 annual security report [64]. Network assaults can be divided into active attacks and passive attacks in terms of attack types [37].

Active attacks, such as denial of service attacks, which are the most common type, may have significant effects on a system's usefulness. The goal of passive assaults is to obtain crucial data from computer systems. Intrusion detection systems have been designed to identify malicious actions in the network to reduce the danger of various sorts of attacks [27; 36].

A Denning presented an early intrusion detection system in 1987, describing a model based on audit data and statistical techniques to spot system irregularities. The primary category we focus on is Machine Learning model based [16; 57].

B. Aim and Objectives

The aim of this study is to present a new method based on machine-learning techniques to classify network traffic into normal and abnormal cases and present a machine-learning model with an accurate detection rate and low execution time, to reduce the number of issues related to the security of the internet.

To achieve a robust approach we pick up several data sets and apply our pipeline for each one of them independently. There are several Network Intrusion Detection data sets available online for researchers to deal with network security issues, for our research we chose the three most well-known ones in this field: UNSW-NB15, NSL-KDD, and KDD Cup 99.

To achieve a robust model that could achieve reasonable accuracy with the best number of features we go through different objectives listed as follows.

- Apply different Feature selection methods to get a robust model with the best number of features that are more representative of the problem, and we can depend on them for prediction. Among these feature selection methods, we pick the most promising methods depending on different feature selection algorithms like Chi2, Relief, Information Gain, and Gain Ratio, we determine the best method for each data set independently.

- Apply different machine learning algorithms among single-based machine learning methods and Ensemble techniques and compare the results of each method of them to achieve the best model architecture.

- Picks the most promising algorithms that could be improved to achieve state-of-the-art performance on the network intrusion detection problem.

C. Research Questions

Based on the main problem we aim to improve is accurately predicting the attack from normal communication. We study the single-based approach that depends on one model on a hand like SVM (Support Vector Machine), Logistic Regression, Decision Tree, Lasso, and Ridge. Other researchers use also ensemble methods using three different approaches Bagging, Boosting, and Stacking. We apply all these boosting techniques to pick the most appropriate algorithm.

Firstly we use all these models without any feature selection, to select the best three models for further analysis and to consider them as level-0 models of the stacking algorithm which will be our final test algorithm. After selecting the best three models we apply the feature selection techniques.

The feature selection algorithm also is one of the main questions of our research which makes us depend on different algorithms that use statistical methods to give a score for each feature. We describe each method of this feature selection and the model we used in later sections. We briefly present the results that other researchers obtain and describe their limitations and what we offer in our work to increase the accuracy.

We use the best subset of features that get the highest scores with the best three selected models to represent our final stacking model. The following list of questions emphasizes our work and describes our interest area of research and in the following sections, we go through the answers in detail.

- RQ1: What is the best Feature selection method that could achieve the best number of features?

- RQ2: Do single-based algorithms better than ensemble learning algorithms for attack detection?

- RQ3: What is the limitation of a single-based machine learning model for Network intrusion detection problems?

- RQ4: Does bagging and boosting algorithms better than stacking?

- RQ5: Does stacking algorithms are better than others?

D. Contributions

We provide a new machine learning promising approach to achieve state of art accuracy on network intrusion detection our stacking models prove great results on the three data sets we use UNSW-NB15, NSL-KDD, and KDD Cup 99. Because there aren't many available research articles concerning stacking models and how well they perform in network intrusion detection tasks. We provide proof of concept that stacking models are a great approach that should be taken into consideration in further development and research. The bullet points show the contributions of this thesis.

- Firstly, we employ the 11 learning models including Decision-Tree, RandomForest, ExtraTrees, GradientBoosting, XGBoost, AdaBoost, SVC, LogisticRegression, Lasso, Ridge, and ElasticNet on all features. Based on the outcomes, we select the top three models for our stacking model, which will act as level-0 models of the stacking algorithm.

- We focus on applying stacking models which construct two-layer models level-0 models and level-1 models which are called the top model which may be Logistic Regression or Neural network. While utilizing the best models selected from our pipeline from all these 11 models to serve the second stage of our work.

- The second stage of our work is feature selection. We use 4 statistical methods Information Gain, Gain Ratio, Chi2, and Relief. we describe each method of them and introduce the results with comparisons to select the best method based on the models' results on different thresholds to select the best number of features that give us the best accuracy.

- Random Forest, Extra Trees, and XGBoost models get the best accuracy in the first stage of our pipeline so, we take them for further analysis and feature selection. After we obtain the best number of features, we train our stacking models using cross-validation.

- Using cross-validation, we train the three models Random Forest, Extra Trees, and XGBoost on the 15 features which obtain the highest scores on feature selection algorithms. We then use these models as a level-0 models to train the stacking models. And on top, logistic regression. The stacking model outperforms the three models separately and outperforms all baseline results, achieving 97.79% test accuracy with only 15 features.

- Then, we apply two more stacking models that use the same level-0 models overall but distinct meta-classifiers. One employs Multilayer perceptron (MLP) with two layers of 20 neurons each in neural networks, while the other uses five layers of 15, 15, 20, and 20. When compared to the logistic regression model that was previously utilized as the top model, accuracy doesn't significantly increase, but model complexity does.

II. LITERATURE REVIEW

A. Introduction

In order to accurately anticipate network data characteristics within a given time frame, industrial Network Intrusion Detection Systems (NIDS) often use quantitative measurements or even derived specifications on feature collections including packet size, inter-arrival time, stream length, and other network data parameters [6]. Both high false positive and false negative rates affect them. A high number of false positives suggests that the NIDS will be mistakenly alerted when there is no actual attack, whereas a considerable rate of false negatives suggests that the NIDS may frequently misidentify threats. Therefore, those industrial methods are insufficient to counteract contemporary challenges.

One of the effective methods for defending against modern threats is the auto-learning approach. With such a large repository of both the Normal and threat network, including events at the host site, they employ supervised, semi-supervised, and unsupervised machine learning algorithms to recognize the trends of various traditional and hostile behaviors.

Numerous publicly accessible data sets are available for studying intrusion detection systems (IDS). The most popular ones among them are UNSW-NB15, NSL-KDD, KDD Cup 99, and DARPA 98. In February 1998, DARPA 98 was first made available. This data collection contains audit logs and several weeks' worth of network data, but it does not represent actual traffic. Developed from DARPA 98, KDD Cup 99 was released to the public in 1999. However, this data set has issues like redundant and duplicate records. IDS is where this data set is most frequently used. NSL-KDD is a modified version of the KDD 99 data set that fixes the issues with the original data set. In 2009, it was made public. The most recent NIDS data collection, UNSW-NB15, was released in 2015. The most complete attack scenarios can be found in this data set.

The UNSW-NB15 data set is by far the most comprehensive one available for evaluating malicious traffic. For this reason, we also use this data set to train, test, and assess the suggested solution. It has 82,332 records in total. 49 features make up each album.

Another important data set is KDD Cup 99 this is the biggest data set available it has 1,074,992 records after removing duplicate records. Each record consists of 43 features.

NSL-KDD is the last data set we care about in our proposal work. This data set consists of 125,973 records. Each record consists of 42 features.

There are 4 types of attack that are mostly used for research some papers also use only binary classification for attack and normal connections. The four different categories of attack patterns are as follows.

Probing In a class of attacks known as probing, an attacker scans a network for information or to identify known weaknesses. An attacker can use the information to seek exploits if they have a map of the machines and services that are accessible on a network. There are various kinds of probes; some of them make use of social engineering strategies while others take advantage of the computer's genuine features. The majority of attacks fall under this category and only moderate technical knowledge is needed [44].

Denial of service attacks A DoS attack is a type of attack where the attacker makes a computer's memory or processing power overloaded or unusable for valid requests, preventing genuine users from accessing the system. DoS attacks can be launched in a variety of methods, including by misusing a computer's legitimate functionality, focusing on implementation flaws, or taking advantage of incorrect system setups. DoS attacks are categorized according to the services that an attacker prevents from being used by authorized users [44].

User to root attacks User-to-root exploits are a type of attack in which the attacker first gains access to a user account on the system and then uses a vulnerability to take control of it. The most frequent exposures in this type of attack are typical buffer overflows, which are brought on by everyday programming errors and erroneous assumptions about the environment [44; 48].

Remote to user attacks An attack known as a remote-to-user (R2U) attack involves sending packets to a target machine over a network and then using that target's vulnerability to gain unauthorized access to the machine's local network as a user. R2U assaults come in various forms, with social engineering attacks being the most prevalent in this category [44; 48].

B. Feature Selection Techniques

The process of classifying usually begins with eliminating redundant attributes. Feature selection and feature extraction techniques are employed to eliminate these redundancies. For increased efficiency, feature selection rather than feature extraction has been used in this paper [56].

Feature selection is a crucial step in machine learning that can aid in the elimination of low-value features, the avoidance of overfitting, the reduction of detection time, and the improvement of model accuracy. Methodologies can be used to categorize feature selection techniques into three groups: filter methods, embedding methods, and wrapper methods [54].

- Model-independent filter methods consistently calculate feature importance without needing to recalculate it, unlike feature correlation approaches.
- Embedded approaches acquire feature importance scores using logistic regression and tree-based machine learning algorithms like the random forest, C4.5, Xgboost, etc. Similar to filter approaches, after rating the features by importance, forward feature search or backward feature elimination can be used to choose feature subsets [28].
- Select feature subsets can be eliminated using Wrapper approaches. Wrapper approaches assess the quality of feature subsets based on the models' actual performance. Wrapper methods are not model-independent, although they can be based on any model. Wrapper approaches are more time and computationally-intensive than filter methods because they would actually train the model for each evaluation of a feature subset. The wrapper method can typically be combined with random search algorithms or other techniques to speed up selection [17].

Authors have described numerous feature selection techniques that are applied to data sets in articles [4; 34] in order to choose the pertinent characteristics that are utilized to assess the classification accuracy of classifiers. Feature selection strategies fall into various categories, such as the Filter method, Wrapper method, and Embedded method. The feature selection techniques described in the study include Cfs subset [35], Information Gain [39], Gain Ratio [29], Filtered Attribute [8], Randomized Hill Climbing [58], and Genetic Algorithms [38]. They examined those algorithms using a variety of search techniques, and some relevant features have been chosen after the irrelevant features have been eliminated. Different classifiers are applied using the best attributes to demonstrate how performance and accuracy have improved.

In our work, we use Embedded approaches relying on some statistical approaches Information Gain [39], Gain Ratio [29], Chi2 [41], and Relief [61] to calculate the importance of features based on score values for each feature and we rank these scores and specify the top features with different threshold values.

Before using the framework and applying feature selection algorithms we first train the models on all features as a baseline for the next development of our work.

C. Computer Network Classification Approaches

The literature review on ML algorithms is presented in this part. The primary goal of this part is to provide an overview of the research that has been done in the area of intrusion detection. Researchers have made significant contributions to ML algorithms, some of which are summarised below, according to the literature.

We divide our classification approaches into two different categories each one of them relying on different techniques. Also, have a slight change in the preprocessing steps with or without the normalization step using standard scaling. Firstly the Single based approaches depend on a single model present. Secondly, Ensemble learning combines more than one model. the combination of them could be boosting, bagging, or stacking. We illustrate these approaches in the next few sections.

1) Single based approaches: Single-based approaches rely on a single model only on the hand. In this section, we explore different research which use single base approaches to solve the problem of intrusion detection.

Vapnik and Cortes first created support vector machines (SVM) [14], which have since been widely used in machine learning, statistics, pattern recognition, and other fields. The fundamental goal of SVM is to locate an input space hyperplane of the largest margin type that divides the training data set.

When the time of the event is not critical, logistic regression is used to forecast the likelihood of a binary result as a function of some predictive variable. When the values of risk variables are known, logistic regression, for instance, can be used to forecast the likelihood that a person would get a disease. Despite being most frequently employed for binary outcomes, logistic regression can also be used for a variety of categorical outcomes But many choices and events have a clear-cut binary outcome (yes/no, succeeded/failed, alive/dead, etc.). Furthermore, unlike discriminant function analysis, logistic regression does not demand that the predictor variables be normally distributed, linearly connected, or of equal variance [21].

In [47], limiting the number of features in the data, authors were able to detect DoS assaults using this data set. They reduced the UNSW-NB15 test set to 68264 packets and the training set to 24596 packets of internet traffic, which includes both regular traffic and DoS attack traffic. By eliminating content features, general purpose features, time fields, and nominal type features in flow and basic field class, the feature set is decreased from 47 to 27. To identify DoS traffic, they suggested using Deep Radial Intelligence with Cumulative Incarnation (DeeRaI with CuI). DeeRaI uses Radial Basis Functions (RBF) to extract intelligence at various abstraction levels. The information gathered is then forwarded to the following level once the weights have been optimized using CuI. They were able to get the best accuracy of 96.15 percent with DeeRaI and CuI. This approach suffers from the decreased number of the training set, also the number of features could be decreased more using different techniques of features selection to obtain a more robust model with good accuracy relative to the same accuracy obtained in this paper.

Another paper [46] uses Bayesian family classifiers. They are Average One Dependence Estimator (AODE), Bayesian Network (BN), and Naive Bayes (NB), and they are used to assess the effectiveness of classifier models to categorize the data instances in the UNSW-NB15 data set in terms of classification rate and processing time. These classifiers use tenfold cross-validation to validate the training set before the model is tested, with the settings set to default as in WEKA tools. they use all the available features in the UNSW-NB15 data set. The best accuracy was obtained using AODE with a percentage of accuracy equal to 94.37%, and the BN algorithm obtains 92.70%. The limitation of this paper is the number of features used they didn't apply any type of feature selection techniques which is a very essential step to remove noisy features that don't help or even degrade the performance of the models.

In [3] described a multilayer hybrid SVM and EVM intrusion detection models. KDD 99 data set was used for the evaluation. In this suggested model, accuracy was attained at 95.75% with less training time. Only known attacks benefit from this strategy; unexpected attacks demand the use of effective classifiers.

Paper [2] uses binary classification to classify either normal or attack data. They obtained an accuracy 97.69% using SVM with $penalty = l1$, $loss = squaredhinge$, and $C = 10^{10}$. Also, they used ANN (Artificial Neural Network) and obtain an accuracy of 94.78%.

In [45], for the purpose of modeling quick and effective intrusion detection systems, they looked into decision trees, support vector machines, linear genetic programming, and an ensemble approach. To investigate these models they used the DARPA data set with its 41 features. They use a subset of 494,021 data points. They obtain an average accuracy of 98.85% using SVM, 92.75% using MARS, and 99.3% using ANN. These experiments' result is obtained using only five classes Normal, Probe, DoS, U2Su, and R2L. Each type of these attack is described in the introduction part of the literature review.

Further, single-classifier-based models are inefficient to detect all types of intrusions efficiently under all scenarios. Most of the researchers obtain the highest accuracy using ensemble learning approaches like the random forest, extra tree, and XGBoost classifiers. These models combine multiple decision tree classifiers together to obtain the final detection. We take about the ensemble classifiers in the next section.

2) Ensemble learning approaches: The fundamental structure of a classifier ensemble is shown in (Figure 1). Although the primary classification technique used in this example is a neural network, any classification technique (such as decision trees) may theoretically be used in its place. The training instances for each network in the ensemble model seen in Figure 1, (networks 1 through network N in this case) are used. The ensemble output is then created for each example by combining the expected outputs from each of these networks [49].

The ensemble models seem to be a great promise in our problem. we investigate more details of ensemble techniques Bagging, Boosting, and Stacking. Also, we show more related works and state-of-the-art ensemble model approaches in the next subsections of bagging, boosting, and stacking.

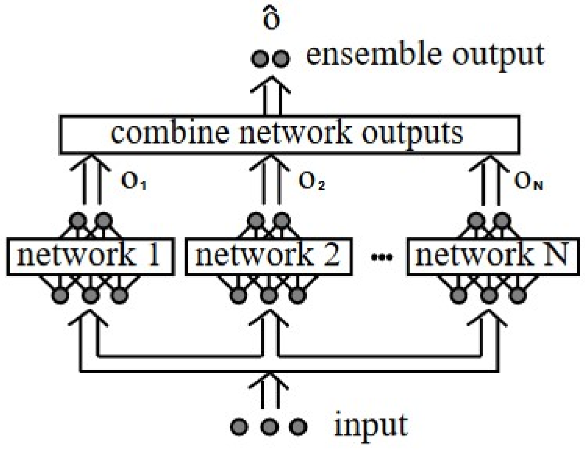


Fig. 1. Ensemble neural networks classifier [49]

3) Bagging: A "bootstrap" ensemble method called bagging trains each classifier on a random redistribution of the training set to produce individuals for the ensemble. The training set for each classifier is created by drawing N examples at random and replacing them with new instances. Depending on the size of the original training set, some of the original examples may be repeated while others may be left out. The ensemble's individual classifiers are created using various random samples from the training set [49].

In [7], they used a variety of supervised machine learning classifiers, and a NIDS was created. The performance of several classifiers was evaluated using the NSL_KDD data set. Random Forest (RF) classifier outperforms other classifiers, according to the results. It yields the lowest FPR and maximum TPR, with an accuracy of 99%. However, there is still a demand for classifiers that can be applied to multiclass categorization.

The idea for Intrusion Detection System (IDS) was put up by [50], as a Hybrid Intelligent Approach. In order to enhance the performance of the resulting model, the authors combined various classifiers. They employed a 10-fold cross-validation classification approach. The NSL-KDD data set is used to perform the experimental outcomes. They achieved an overall accuracy of 84.24% on 6 labels representing 5 types of attacks in addition to the normal label. The limitation of this paper is the accuracy of the Dos attack label, they achieved an 0.1109 f1 score on this label which is very bad.

In [45], they investigated the effectiveness of Support Vector Machines (SVMs), Multivariate Adaptive Regression Splines (MARS), and Artificial Neural Networks (ANNs). they demonstrate that, for intrusion detection, an ensemble of ANNs, SVMs, and MARS outperforms individual approaches in terms of classification accuracy. They use the DARPA data set with it is all 41 features. They use s subset of 494021 data instances. They get an average accuracy of 99.82% on five types of attacks Probe, DoS, U2Su, and R2L and the fifth label is Normal connection.

In [19], fifteen features are selected from the NSL_KDD data set using a Genetic Algorithm. These fifteen selected features are passed into a Bagging of classifiers. They achieved the best accuracy of 99.7166% using bagging with a partial Decision Tree, and they achieved a classification accuracy of Bagging C4.5 trees at 99.175%. This accuracy is relatively high but in the test set, it degraded so much. These algorithms suffer from overfitting problems and some regularization techniques should be added to solve this problem.

Another paper [18] uses the NSL_KDD data set. The authors of this paper select 29 features based on the method proposed in [23] for experimental work. WEKA data mining tool is used to manually select these 29 relevant features. They achieved 99.661% using the RandomForest algorithm applied using cross-validation and achieved 76.2287% accuracy on the test set which is suffering from overfitting. Some regularization techniques should be used to improve the accuracy of the test set. They also used Bagging of (REPTree). The REPTree minimizes the false positive rate and increases the classification accuracy. They achieved 99.6761% on the validation set with the cross-validation algorithm and achieved 81.2988% on the test set which is better than Random forest with respect to the test set. We aim to work more on this issue of overfitting by proposing another technique of Ensemble models which is the Boosting Algorithm we take about this in the next section.

4) Boosting: Techniques concentrate on creating a number of classifiers. Based on the effectiveness of the preceding classifier(s) in the series, the training set used for each subsequent member is selected. In boosting, examples are selected more frequently than those that were properly predicted by the prior classifiers in the series [49]. In order to improve the performance of the present ensemble for examples for which it is unable to anticipate, boosting aims to create additional classifiers. Note that in contrast to the bagging ensemble, boosting takes the effectiveness of the previous classifiers into account while resampling the training set.

In [9], they used XGBoost on KDDCup99 and achieved an accuracy of 99.95%. This model is too good to classify the attack type, but we can achieve more accuracy to classify either attack or normal connection. In our work, we care more about binary classifications without giving attention to the type of attack and that is important for the real environment we want to capture each attack connection without caring about its type. We can classify the type after classifying whether the connection is normal or an attack one which is the most important step for security.

Boosting models is a common supervised learning algorithm technique. Adaboost and stochastic gradient boosting are just two examples of boosting models. These types of boosting models have been employed in numerous studies [51; 62] to discover the best outcomes.

A boosting Anti-Colony optimization technique for finding an intrusion was proposed in [55]. This algorithm, which is an enhanced version of the Anti-Miner algorithm, is used to generate classification criteria.

According to [24], the Adaboost technique uses basic overfitting to discover network intrusions. To increase the detection rate for the UNM data set, Chen et al. [13] suggested an ensemble strategy that combines the Adaboost and Incremental Hidden Markov models. By merging classifiers, Schapire [55] describes a strategy that can transform a weak learner into a strong learner.

Another paper [18] uses the NSL_KDD data set. The authors of this paper select 29 features based on the method proposed in [23] for experimental work. WEKA data mining tool is used to manually select these 29 relevant features. They use the AdaBoost algorithm and achieved 94% on the validation set, and get 74.583% on the test set. This model is suffering from overfitting and some regularization parameters can use to improve the accuracy of the test set.

In paper [26], they used AdaBoosting Classifier on the NSL_KDD data set. They divide the data into four 4 data sets based on the type of attack. They used 4 types of attacks which are Dos (39 features), Probe(32 features), R2L, and U2R (39 features). After they cluster the data set into four sub-datasets and select the most relevant features for each data set independently, they apply the Adaboosting Classifier to each data set. They achieved an accuracy of 97.61% for DOS attacks and 97.15% for the other types of attacks Probe, R2L, and U2R. The accuracy is relatively high, but the number of selected features is so high. The number of features could be reduced more using other feature selection techniques.

In paper [65], they achieved an average accuracy of 99.897% on four types of labels DoS, Normal, Probe, and R2L for the NSL_KDD data set using the Adaptive boosting (AdaBoosting) model. They also achieved 99.99% on the AWID data set using the same model AdaBoosting. AdaBoosting achieved the highest accuracy on the two data sets. The author of this paper describes briefly how this AdaBoosting works.

The following section presents the Stacking algorithm which is a kind of ensemble learning approach.

5) *Stacking:* Not many research papers are available online about Stacking models and their performance in Network intrusion detection problems, so, in our approach, we focus on applying stacking algorithms.

An array of estimators called base models or level-0 models and a classifier, in the end, called the level-1 model. A classifier is used to produce the final prediction in stacked generalization, which stacks the output of each separate estimator (base model). By using each estimator's output as an input for a final estimator (meta-classifier), stacking enables the utilization of each estimator's strongest features.

Another strategy is to train a number of classifiers and then utilize the results as features to train a generalizer, which is a new classifier. The classifiers that were developed using the initial data distributions work as dimensionality reduction maps in this way [57].

The distinctions between the original distributions are clearly visible in the new one-dimensional binned distribution, which is the classifier output, thanks to good performance maps. However, it is possible that some classifiers are more adept than others at recognizing connections between particular kinematic distributions; for this reason, combining diverse maps is likely to improve performance, as was mentioned in the Introduction [5].

In paper [52], they select 11 features from the UNSW-NB15 data set using Information gain (IG). They train a stacking classifier pipeline that employs base models and meta-classifier, called logistic regression (meta-classifier), K-nearest neighbor (KNN), random forest (RF), and support vector machine (SVM), respectively. They obtained an accuracy of 94% on binary classification (Attack or normal connections). The limitation of this paper is the choice of the base models which is not dependent on an effective way, we aim to solve this issue in our work and pick the most appropriate base models, and meta-classifier effectively.

III. PROPOSED STACKED COMPUTER NETWORK TRAFFIC APPROACH

A. Introduction

This section introduces our pipeline from data collection, pre-processing, feature selection, training, and evaluation of the models as shown in (Figure 2). We select the best models to represent the level-0 models of the stacking algorithm and then compare the best models obtained from the first step, training all 11 models on all features without the feature selection step with the stacking models trained on the subset selected features.

We apply the pipeline to the three collected data sets which are UNSW-NB15, NSL-KDD, and KDD Cup 99. But the cleaning step is different for each data, we will illustrate these differences in pre-processing step.

We split our pipeline into different steps the first one is pre-processing which applies cleaning, one-hot encoding, and normalization for models that require normalized data like Lasso, and Logistic Regression.

Then the data is ready for training so, we split the data for training and testing as shown in (Figure 2). Then we apply a cross-validation algorithm to gain a robust result that describes each model. This step for all features is just a baseline result for our next step, which applies feature selection using four different statistical methods: Information Gain, Gain Ratio, Chi2, and Relief on the best three models selected from this step.

The output of this pipeline is the best-selected 3 models representing our level-0 models of the stacking algorithm. Also, we apply the regularization step to select the best hyperparameters for each selected model before we use them for stacking.

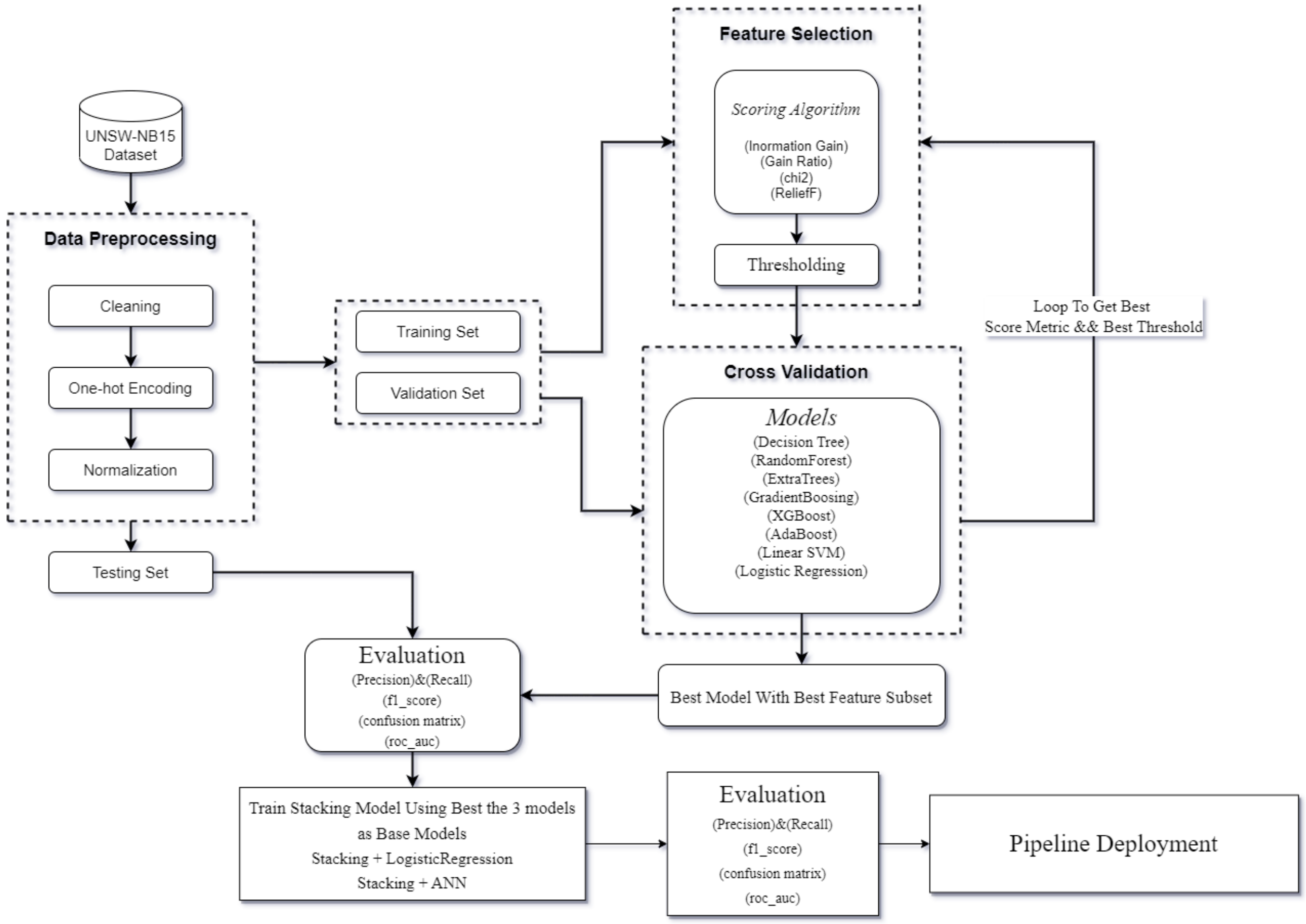


Fig. 2. Proposed network intrusion detection approach

B. Data Collection

There is a lot of Network Intrusion Detection data sets available online but we select the most popular ones among them which are UNSW-NB15 [25], NSL-KDD [60], and KDD Cup 99 [15].

The most extensive data collection for evaluating malicious traffic is UNSW-NB15, by far. Due to this, we additionally train, test, and evaluate our suggested pipeline using this data set. There are 82,332 records total in it. Each record has 49 features.

Another important data set is KDD Cup 99 this is the biggest data set available it has 1,074,992 records after removing duplicate records. Each record consists of 43 features.

NSL-KDD is the last data set we care about in our proposal work. this data set consists of 125,973 records. Each record consists of 42 features.

There are 4 types of attack that are mostly used for research some papers also use only binary classification for attack and normal connections. The four different categories of attack patterns are as follows Probe, DoS, U2Su, and R2L illustrated in the literature review section.

C. Data Pre-processing

We apply multiple preprocessing steps and apply data analysis to get more intuition about different aspects of the problem we deal with. Figure 3, divides the preprocessing into three modules we illustrate each module of them in detail. Firstly we apply the Clean module the first module which checks for null values, redundant rows, and unnecessary columns, and checks for outliers. The clean module is the simple one because of the data simplicity.

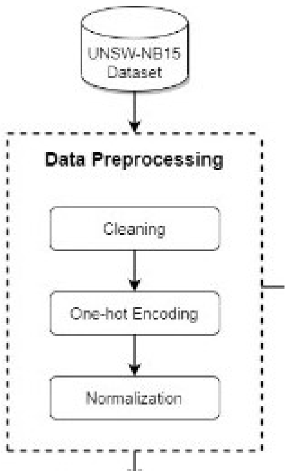


Fig. 3. Pre-processing sub framework

Figure 4, illustrates the distribution of the label along the data set. label 1 means attack and label 0 mean normal communication. Figure 4 shows that the data don't suffer from data imbalance there's a slight difference in the distribution.

For one hot encoding, we will first count the unique values for each categorical feature. Figure 5, shows that the protocol feature has 131 unique values and the other 2 categorical features service and state have 13, and 7 unique values respectively. These values are too much for one hot encoding which will explore the number of features too much so, will reduce these values by considering the top 2 values for each feature, and the other values will be represented as OTHER values. This method will reduce the number of one hot encoding feature to just 9 encoded features which will replace these 3 categorical features then, the increasing number of features due to one hot encoding will be just 6 features.

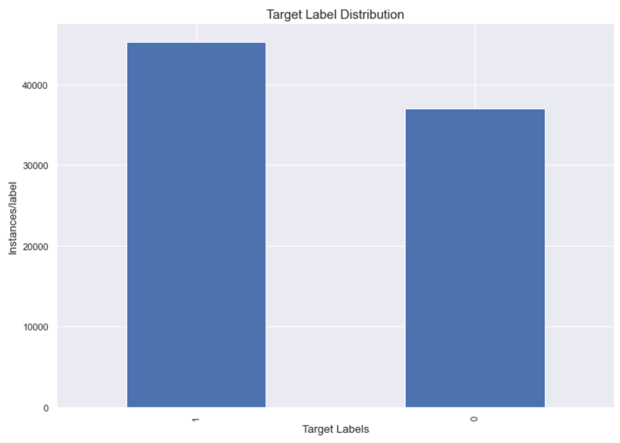
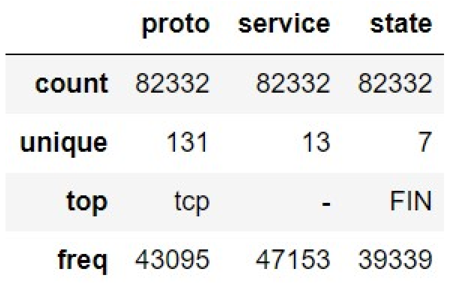


Fig. 4. Label distribution 1 for attack 0 for normal

The chosen number of values top 2 is chosen by manual tuning to get the best performance for the base model which



	proto	service	state
count	82332	82332	82332
unique	131	13	7
top	tcp	-	FIN
freq	43095	47153	39339

Fig. 5. Categorical features occurrences

will be described later. The percentage of occurrences for the top 2 values along the data. For each categorical feature, it's around 88% which it's a high percentage that makes the top 2 values represent our data set more wisely.

Standard Scaling is the last module in our pre-processing pipeline. The module used only with the models required scaling as a main pre-processing before training. The tree-based models don't require any type of scaling because these models rely on statistical measures which don't affect the scaling the performance is the same.

Standardize features by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as seen in Equation (1).

$$Z = (X - U)/S \tag{1}$$

where U is the training samples' average, and S denotes their standard deviation. Standardization of data sets is a common criterion for many machine learning estimators since individual features may perform poorly if they do not more or less reflect standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

For instance, several components of a learning algorithm's objective function, such as the Radial Basis Function (RBF) kernel of an SVM or the L1 and L2 regularization methods of a linear model, presuppose that all features are centered around 0 and have a variance that is distributed in the same order. One variable may dominate the objective function and prevent the estimator from properly inferring from other characteristics as planned if its variance is orders of magnitude bigger than those of the others.

As seen in Figures 6, 7, and 8, the distribution of the unique values for each feature of the categorical features "protocol", "service", and "state" respectively, there are about three unique values that are represented in most of the data set instances. So, we reduce the number of unique values into just three values the most represented ones as seen in (Figure 9).

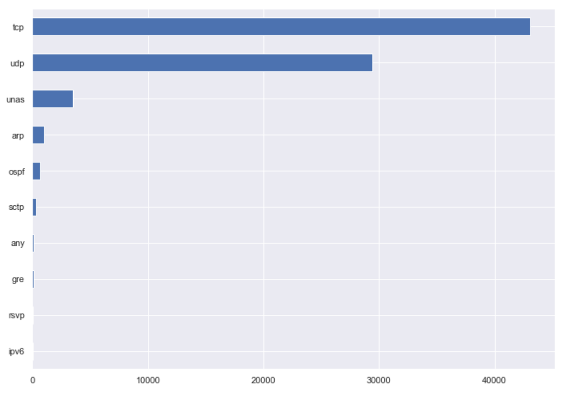


Fig. 6. Feature ”protocol” top 10 unique values with highest frequencies

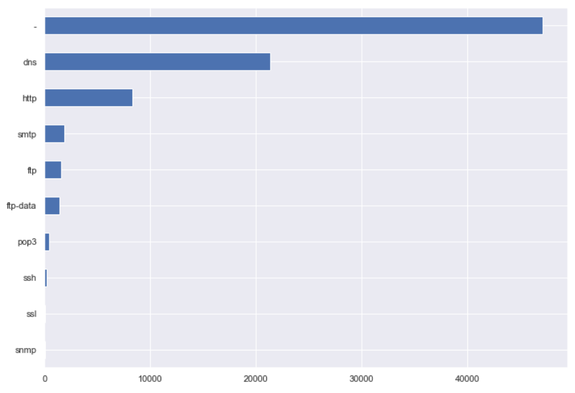


Fig. 7. Feature ”service” top 10 unique values with highest frequencies

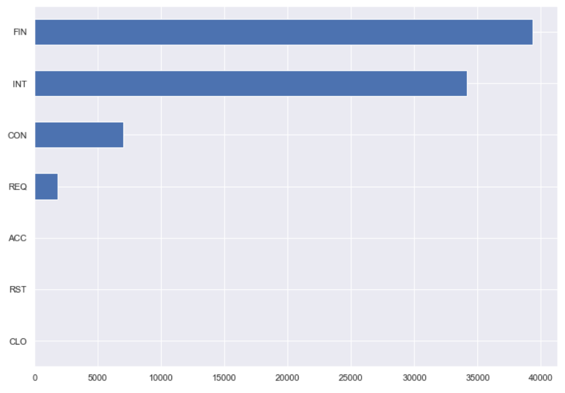
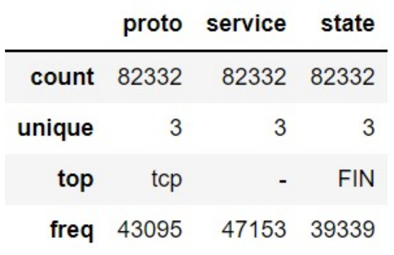


Fig. 8. Feature ”state” unique values frequencies



	proto	service	state
count	82332	82332	82332
unique	3	3	3
top	tcp	-	FIN
freq	43095	47153	39339

Fig. 9. The categorical features after removing the lowest frequency values

D. Feature Selection

In many cases, reducing the input features of some data sets may improve the model’s performance while also eliminating the noisy features that reflect our model’s accuracy and require less processing power to run the model.

When employing statistically based feature selection methods, each input variable’s relationship to the target variable is assessed, and the input variables with the strongest relationships are chosen. Even though the choice of statistical measures depends on the data type of the input and output variables, these techniques can be simple and effective.

We use 4 statistical methods Information Gain, Gain Ratio, Chi2, and Relief. In this section, we will describe each method of them but in the next section, we introduce the results with comparisons to select the best method based on the models’ results. We apply these feature selection algorithms for the 3 selected data sets UNSW-NB15, NSL-KDD, and KDD Cup 99.

For each data set, we select the best algorithm independently based on the validation accuracy obtained from applying thresholds in the scores of each metric to get the best number of features with the highest evaluation metric.

1) Information gain: To determine the amount of reduction in entropy or surprise, Information Gain (IG), divides a data set according to a certain value of a random variable. A set of groups of samples with a lower entropy and hence less surprise imply a higher information gain meaning that this variable has a higher importance and a strong relation to the target variable.

You may remember that information quantifies an event’s surprise in bits. Events with a lower likelihood contain more information than those with a greater probability. Entropy measures the amount of information contained in the probability distribution of a random variable. A distribution with unequal probabilities of events has higher entropy than one with a skewed distribution [30; 39].

In the information Gain method, we commonly talk about the ”surprise” or the ”entropy” of an event. Since they are more surprising, low-probability events contain more information. Probability distributions with equally likely events have higher entropies and are more startling.

-Skewed Probability Distribution (unsurprising): Low entropy .

-Balanced Probability Distribution (surprising): High entropy.

For example, in a binary classification problem we have only two classes, we can calculate the entropy of the data sample as seen in Equation (2). P is the probability distribution of a specific label occurrence in the data set [30].

$$Entropy = -(p(0) * log(P(0)) + p(1) * log(P(1))) \tag{2}$$

Entropy can be calculated in this way to determine a data set's purity, such as how evenly distributed the classes are.

An entropy of 0 value indicates that there is only one class present in the data set, whereas an entropy of 1 or more bits indicates the maximum entropy for a balanced data set (depending on the number of classes). Values in between these extremes suggest levels that fall in between these two extremes.

Entropy can be used to determine how a change to the data set, such as the distribution of classes, affects the data set's purity using information gain. A lower entropy indicates greater purity or decreased surprise.

The expected decrease in entropy brought on by dividing the instances into groups based on this property is known as information gain. By dividing a data set S by a random variable that has a range of values, for instance, we could want to assess the effect on purity. This can be determined as seen in Equation (3).

$$IG(S,a) = H(S) - H(S|a) \tag{3}$$

Where $H(S)$ is the data set's initial entropy (described above) and $H(S|a)$ is the data set's conditional entropy in the presence of the variable a. Information for the data set S for the random variable an is contained in the expression $IG(S,a)$.[30].

The gain for the variable an in data set S is described by Equation (4). It is the number of bits saved after the data set transformation. Each observed value of the data set may be divided into groups, and the conditional entropy can be determined by adding the ratio of examples in each group out of the whole data set multiplied by the entropy of each group.

$$H(S|a) = \sum_{v} (\frac{Sa(v)}{S}) * H(Sa(v)) \tag{4}$$

The ratio of instances in the data set where variable a has the value v is represented in this formula by $Sa(v)/S$, and the entropy of a set of samples where variable a has the value v is represented by $H(Sa(v))$ [30].

2) Gain ratio: The gain Ratio is the same as Information gain but it has some improvement. It introduces a new balancing term called Intrinsic Information in an effort to reduce the bias of Information Gain on heavily branching predictors that have many unique values.

The entropy of sub-data set proportions is referred to as intrinsic information (II). In other words, it refers to how difficult it is for us to determine which branch a randomly chosen sample is placed in [29; 41].

Intrinsic Information's mathematical formula is given by Equation (5). Then the gain ratio can be calculated by Equation (6).

$$II = -(\sum_{j} \frac{|D_j|}{|D|} * \log_2 \frac{|D_j|}{|D|}) \tag{5}$$

$$Gain\ \ Ratio = \frac{Information\ \ Gain}{Intrinsic\ \ Information} \tag{6}$$

3) Chi2: The chi-square method is used to assess categorical characteristics in a data set. The Chi-square between each feature and the desired result is calculated, and the features with the highest Chi-square scores are then selected. It determines if the relationship between two categorical variables in the sample is consistent with the relationship between them in the population [42].

Chi-square results are given by Equation (7).

$$X^2 = \frac{(Observed\ \ frequency - Expected\ \ frequency)^2}{Expected\ \ frequency} \tag{7}$$

Where the number of observations for the class is the observed frequency. If there was no relationship between the feature and the target, the expected frequency would be the number of expected observations for the class.

4) Relief: The original Relief algorithm was created by Kira and Rendell [31; 32], and was influenced by instance-based learning [1; 12]. Relief creates a proxy statistic for each feature as a means for selecting features for individual evaluation filtering. This statistic can be used to gauge the "quality" or 'relevance' to the target concept (i.e. predicting endpoint value).

These feature statistics are known as feature weights (W[A] = weight of feature "A") or, more colloquially, feature "scores," which can vary from -1 (worst) to +1. (best). Notably, the original Relief technique had no mechanism for handling missing data and was only applicable to binary classification issues. Instead of going into depth here about how to apply Relief to multi-class or continuous endpoint problems [33].

The Relief algorithm cycles through m randomly chosen training examples (R_i), chosen without replacement, where m is a user-defined parameter, as shown by the pseudo-code in Algorithm 1. The feature score vector W is updated each cycle using feature value discrepancies between the target and nearby instances as the "target" instance R_i. As a result, the separation between the "target" instance and every other instance is determined in every cycle. Relief specifies the target's two closest neighbors: one belonging to the same class is known as the nearest hit (H), while the other is known as the nearest miss (M). If a feature's value differs between the target instance R_i and either the closest hit H or the nearest miss M, the cycle's final step adjusts the feature's weight in W [61].

Algorithm 1 Original Relief algorithm pseudo-code [32]

Require Input each training example requires a vector of feature values and the target value.

```
1:  $n \leftarrow$ total number of training examples;
2:  $a \leftarrow$ total number of features in the training examples;
3:  Parameter: $m \leftarrow$ selected number of random training examples out of n total examples used to update the value of feature weights $W$;
4:  initialization step $W[A] := 0.0$
5:  for $i \leftarrow 0$ to m do
6:      select a random example out of m random examples $R_i$;
7:      find a nearest hit $H$ and nearest miss $M$ examples for the select example $R_i$;
8:      for $A \leftarrow 1$ to a do
9:          $W[A] := W[A] - diff(A, R_i, H)/m + diff(A, R_i, M)/m$
10:     end for
11: end for
12: return the feature weights vector $W$ has the feature scores that estimate the importance of each feature in the data set.
```

E. Proposed Machine Learning Algorithms

Not many research papers are available online about Stacking models and their performance in Network intrusion detection problems, so, in our approach, we focus on applying stacking algorithms but using the best models selected from our pipeline.

We employ 11 different learning models DecisionTree, RandomForest, ExtraTrees, GradientBoosting, XGBoost, AdaBoost, SVC, Logistic Regression, Lasso, Ridge, and ElasticNet. based on the results we select the best three models for our stacking model to represent the base models (level-0 models).

1) Single based approach: We employ 6 different Single Based models which are decision tree, SVM, Logistic Regression, Lasso, Ridge, and ElasticNet. We give a summary of each model algorithm and the structure of each approach of them.

Support Vector Machine Classification and regression issues are resolved using a Support Vector Machine (SVM), one of the most popular supervised learning algorithms. However, it is widely applied in Machine Learning Classification problems.

With the help of the SVM algorithm, we can quickly categorize incoming data points in the future by determining the best line or decision boundary that can divide n-dimensional space into classes with maximum margin boundaries that can regularize well for unseen data points. The name for this ideal decision boundary is a hyperplane.

The extreme vectors and points that help create the hyperplane are chosen by SVM. The SVM technique is based on support vectors, which are utilized to represent these extreme situations [63].

The hinge loss function is shown in Equation (8) useful for extending SVM to instances when the data are not easily separable linearly.

$$max(0, 1 - y_i(W^T x_i - b)) \tag{8}$$

It should be noted that y_i is the i-th target and $W^T x_i - b$ is the i-th output.

Therefore, the optimization objective we aim to reduce is defined as shown in Equation (9)[63].

$$\lambda||W||^2 + [\frac{1}{n}\sum_{i=1}^{n} max(0, 1 - y_i(W^T x_i - b))] \tag{9}$$

Logistic Regression A supervised classification algorithm is essentially what the logistic regression Machine Learning algorithm categorized. For a given set of characteristics or inputs X, in a classification issue, the target variable or output y can only take discrete values.

Contrary to popular belief, logistic regression is a regression model. To predict the likelihood that a particular data entry will fall into the category denoted by the number "1," the model builds a regression model. Similar to linear regression, which assumes that the data follows a linear function, logistic regression models the data using the sigmoid function described in (Figure 10). The sigmoid function is defined as shown in Equation (10).

$$\phi(z) = \frac{1}{1+e^{-z}} \tag{10}$$

By definition of the sigmoid function, Logistic Regression (LR) employs the hypothesis function used for prediction given by Equation (11).

$$h(x_i) = \phi(W^T x_i) = \frac{1}{1+e^{-W^T x_i}} \tag{11}$$

where W are the learning parameter weights.
The logistic regression cost function is inversely proportional to the parameter likelihood. Therefore, using the log-likelihood equation, we can derive the following formula for the cost function, J given by Equation (12).

$$J(W) = \sum_{i=1}^{n} -y_i log(h(x_i)) - (1-y_i) log(1-h(x_i)) \tag{12}$$

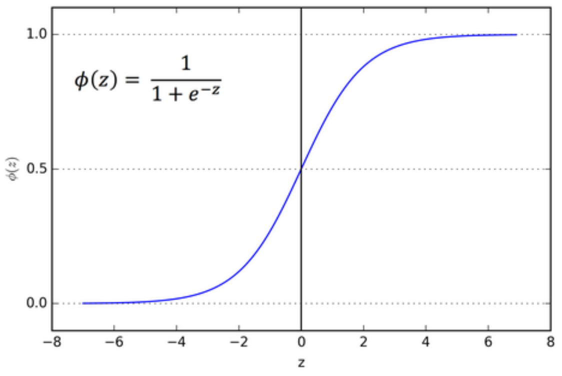


Fig. 10. Sigmoid function

Lasso A well-known model by the name of LASSO [59] successfully combines a simple linear model with a constraint with an l1-penalty component to the objective function.

It's the same as Logistic Regression but by adding the l1-penalty to the cost function. The new loss function after adding the penalty term given by Equation (13).

$$J(W) = \sum_{i=1}^{n} -y_i log(h(x_i)) - (1-y_i) log(1-h(x_i)) + \lambda \sum_{j=1}^{P} |w_i| \tag{13}$$

The algorithm reduces the sum of squares under a restriction. A classification model is created by shrinking some Beta to zero. The L1 regularization penalty's severity is controlled by the tuning parameter lambda. Basically, shrinkage is measured by lambda. When lambda is equal to 0, no parameters are dropped. As lambda rises, bias rises as more and more coefficients are deleted and set to zero. The coefficients are all removed when lambda equals infinity. Variance rises as lambda lowers [66].

Ridge To avoid overfitting, regularization that penalizes model coefficients is used. When a model is overly complicated and captures noise in the data rather than the underlying signal, it is known as overfitting, a prevalent problem in machine learning. On new data, this may result in poor generalization performance. Ridge classification fixes the issue by introducing a penalty element that discourages complexity in the cost function. As a result, the model's ability to generalize to new data is improved.

Ridge classification functions by introducing a penalty component that discourages complexity in the cost function. Typically, the penalty term is the sum of the squared coefficients of the model's parameters. This reduces overfitting by requiring the coefficients to remain small. By altering the penalty term, regularization can be regulated to a certain extent. More regularization and smaller coefficient values occur as the penalty increases. When there is a lack of training data, this can be useful [43].

2) Ensemble approaches: Here we illustrate the ensemble algorithms bagging, boosting, and stacking. Let's start by bagging.

Bagging Bagging is a "bootstrap" ensemble technique that develops several classification models for the ensemble by training each classifier on a random redistribution of the training data set. Each classifier's training set is produced by selecting N examples at random and replacing them with other instances from the data. Some of the original instances might be repeated while others might be omitted, depending on the size of the original training set. Individual classifiers for the ensemble are built using a variety of random samples from the training data, according to [49].

We depend on the Decision Tree classifier model described in Figure 11, to be the base classifier of the bagging algorithm. we use the most popular bagging models which are Random Forest, and Extra Trees models.

Decision Tree A supervised machine learning approach called a decision tree can be applied to classification and regression problems. The method is based on a decision support tool that employs a model of options and possible outcomes that resembles a tree.

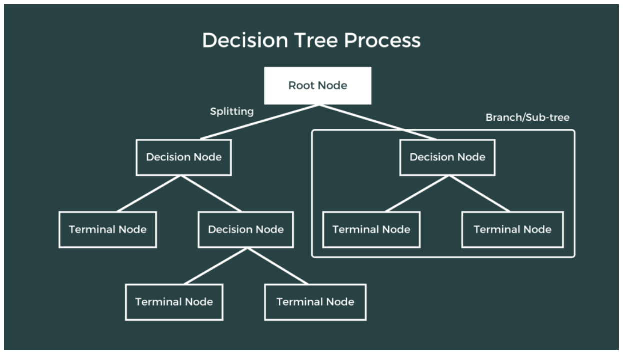


Fig. 11. Flow chart structure describing the decision tree model [53]

- The full data set, which can be further partitioned into several subsets, is represented by the root node.
- To split a node into two or more sub-nodes is referred to as splitting.
- Decision Node: Depending on specific circumstances, a sub-node may divide into more sub-nodes. A decision node is a node that decides to split.
- Nodes that do not split are referred to as Leaf or Terminal nodes. These nodes are frequently the tree's conclusion.
- Pruning: Pruning is the removal of a decision node's child nodes. To avoid overfitting issues, pruning is frequently used in decision trees.
- Branch/Sub-Tree: A branch or sub-tree is a portion of the overall tree.
- Sub-nodes are the offspring of the parent node, whereas a node that is divided into sub-nodes is referred to as the parent node of sub-nodes. The decision node is the parent of the terminal nodes in the illustration below (child).

A decision tree classifier is typically built top-down, with a variable being selected at each stage to determine the best way to divide the set of variables. There are various ways to calculate the splits, and the ”optimal split” is chosen by a pre-established splitting criterion. We depend on the Entropy criterion which is given by Equation (14).

$$Entropy = -\sum_{i=1}^{n} p_i log_2 p_i \tag{14}$$

The randomness of the samples in a specific split is measured by entropy. Entropy equals zero if the samples are entirely homogeneous. Entropy equals one if the samples are evenly divided among the specified classes [53].

Random Forest Algorithm Random forest is a popular supervised machine-learning technique for classification and regression problems. It constructs decision trees from a variety of samples using the average for regression and the majority vote for classification. One of the Random Forest Algorithm's key features is its ability to handle data sets with both continuous and categorical variables, as in regression and classification [40].

Extra Trees Algorithm Extra trees are another type of bagging ensemble approach, where random trees are constructed using samples from the training data set. An excessively Randomized Trees Classifier (Extra Trees Classifier) is a type of ensemble learning technique that uses the results of multiple de-correlated decision trees gathered in a ”forest” to obtain its classification output [10]. It simply differs conceptually from a Random Forest Classifier in terms of how the forest's decision trees are constructed. Each decision tree in the Extra Trees Forest is constructed using the original training sample. Then, at each test node, each tree is given a random sample of k features from the feature set and is required to select the best feature from this sample to split the data in accordance with specific mathematical criteria typically the Gini Index [20]. This random selection of features results in a large number of de-correlated decision trees [22].

Boosting Boosting is an approach that emphasizes producing numerous classifiers. The training set used by each next classifier of the series is chosen based on how well the prior classifier(s) performed [49]. Boosting seeks to construct new classifiers to enhance the performance of the current ensemble for situations for which it is unable to anticipate. Note that when resampling the training set in Bagging, the prior classifiers' effectiveness is not considered.

XGBoost Algorithm XGBoost technology is used to create gradient-boosted decision trees. XGBoost models frequently win Kaggle Competitions. In this method, decision trees are produced in order. In XGBoost, weights matter a lot. Prior to entering the decision tree that predicts outcomes, each independent variable is given a weight. Before being entered into the second decision tree, variables that the first one predicted wrongly are given additional weight. To create a strong and reliable model, these many classifiers and predictors are then combined. Regression, classification, ranking, and custom prediction are just a few of the issues it can be used to solve.

Stacking A machine learning ensemble algorithm is called stacking or stacked generalization. To determine how to combine the predictions from two or more underlying machine learning algorithms, it employs a meta-learning algorithm. The advantage of stacking is that it can use a variety of effective models to accomplish classification or regression tasks and produce predictions that perform better than any one model in the ensemble [11]. the advantages of the Stacking ensemble techniques over another bagging, and boosting techniques:

- In contrast to bagging, stacking often uses different models (e.g., not all decision trees) that are trained on the same data set instead of samples of the training data set.
- In contrast to boosting, a single model is employed in stacking to figure out how to most effectively combine the predictions from the contributing models instead of a sequence of models that correct the predictions of prior models.

The architecture of a stacking model consists of two or more base models shown in Figure 12, also known as level-0 models, and a meta-model, or level-1 model, which integrates the predictions of the base models.

- Level-0 Models (Base-Models): The models are trained on the original data source, and then predictions are compiled.
- Level-1 Models (Meta-Model): The models trained to combine the predictions of the level-0 models.

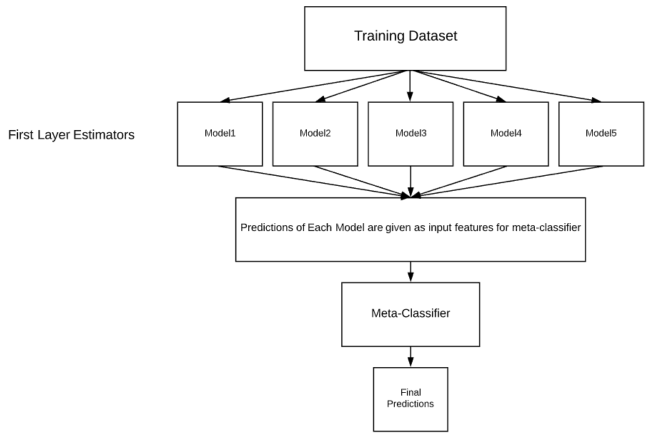


Fig. 12. Stacking classifier [57]

When several separate machine learning models are competent on a data set but are competent in various ways, stacking is acceptable. Another way to put it is that there is little or no association between the model predictions and the mistakes in those predictions.

Base models are usually complicated and diverse. Therefore, it is frequently a good idea to apply a number of models, such as linear models, decision trees, support vector machines, neural networks, and more, that make rather various assumptions about how to achieve the predictive modeling assignment. Other ensemble approaches, such as random forests, can be used as foundation models [11].

Although stacking is designed to boost modeling performance, this gain is not always ensured. If performance may be increased, it is determined by the complexity of the issue and if it is sufficiently well represented by the training data and complex enough that there is more information to learn by combining predictions. It also depends on the base models used and whether or not they are accurate and sufficiently uncorrelated or errors.

F. Evaluation Metrics

The main purpose of using these evaluation metrics is to predict how well a machine learning model will perform on new data. Metrics like accuracy, precision, and recall are helpful for evaluating classification models for balanced data sets. Alternative methods, including ROC/AUC, and f1 score are more useful for evaluating the model's performance when the data is imbalanced.

The ROC curve, which is a complete curve that offers detailed information about the classifier's behavior, is more than just a single number. Additionally, it takes time to swiftly compare various ROC curves to one another.

1) Accuracy: The frequency that the classifier predicts correctly is how accuracy is calculated. Accuracy can be defined as the proportion of accurate predictions to all predictions as formulated in Equation (15).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

You would assume that a model is working well when it reports an accuracy rate of 99%, but this isn't always the case and in some cases, it might be deceptive. I'll use an example to demonstrate how this works.

Accuracy is beneficial when the target class is well-balanced, but it is not a good alternative for classes that are out of balance. Imagine a case where our training data includes 99% images of the dog but just 1% images of the cat. The dog would then always be predicted by our model, giving us 99% accuracy. Data is really continually skewed, as proven by spam emails, credit card fraud, and inaccurate medical diagnoses. Additional metrics like as recall and precision should be incorporated to improve model evaluation and gain a complete picture of the model evaluation.

Our data sets don't suffer imbalanced data but there is a slight difference between positive and negative target labels.

So, we also consider precision, recall, f1-score, and AUC metrics to get better insight into how our models behave. Also, it may be better if we want to adjust the precision-recall trade-off. The recall will decrease as precision is increased, and vice versa. The precision/recall trade-off is what is referred to as. Because the classifier behaves differently for various threshold values, the threshold value can be adjusted to vary the positive and negative predication.

2) Precision: Precision reveals how many of the situations that were predicted with accuracy ended up being positive. When false positives are more problematic than false negatives, precision is helpful. Precision is essential for e-commerce websites, music or video recommendation systems, and other applications where inaccurate results could cause customers to leave, which would be bad for business.

Precision for a label is defined as the number of real positives divided by the number of expected positives, as seen in Equation (16).

$$Precision = \frac{TP}{TP + FP} \tag{16}$$

3) Recall: Recall measures the proportion of real positive cases that our model correctly predicted. It is a useful indicator when a false negative is more significant than a false positive. In medical conditions, it is essential since, even if a false alarm is raised, genuine positive examples shouldn't go ignored.

Recall for a label is defined as the number of true positives divided by the total number of actual positives, as seen in Equation (17).

$$Recall = \frac{TP}{TP + FN} \tag{17}$$

4) F1 score: The F1 score is a combination of Precision and Recalls measurements. It achieves its optimum when Precision and Recall are equal.

The F1 Score is the harmonic mean of precision and recall, and can be calculated as seen in Equation (18).

$$F1 = 2 * \frac{PrecisionxRecall}{Precision + Recall} \tag{18}$$

More high values are penalized by the F1 score. F1 Score may function as a useful evaluation statistic in the following circumstances:

- When FP and FN are equally costly.
- Adding more data doesn't effectively change the outcome.
- True Negative is high.

5) AUC-ROC: Plotting the TPR (True Positive Rate) vs the FPR (False Positive Rate) at various threshold values results in a probability curve known as the Receiver Operator Characteristic (ROC), which separates the "signal" from the "noise". A classifier's capacity to distinguish between classes is measured by the area under the curve (AUC).

A ROC curve's X-axis value corresponds to the False Positive Rate (FPR), while the Y-axis value corresponds to the True Positive Rate (TPR). A higher Y-axis value shows that there are more True Positives (TP) than False Negatives (FN), while a higher X-axis value indicates that there are more False Positives (FP) than True Negatives (TN). Therefore, choosing a threshold depends on how well you can balance FP and FN.

6) Confusion matrix: As you can see in Figure 13, there are only two classes in a binary classification problem, ideally a positive and a negative class. Let's now examine the Confusion Matrix's metrics.

- True Positive (TP): It refers to the number of predictions where the classifier correctly predicts the positive class as positive.
- True Negative (TN): It refers to the number of predictions where the classifier correctly predicts the negative class as negative.
- False Positive (FP): It refers to the number of predictions where the classifier incorrectly predicts the negative class as positive.
- False Negative (FN): It refers to the number of predictions where the classifier incorrectly predicts the positive class as negative.

It's usually preferable to utilize the confusion matrix as your machine learning model's evaluation criterion. It provides you with a very straightforward but effective performance measurement for our models.

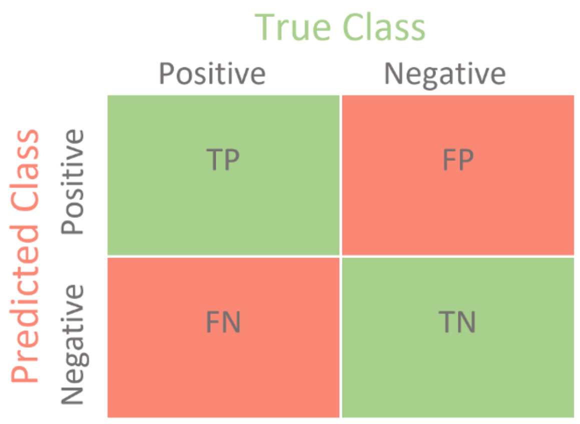


Fig. 13. Confusion matrix for binary classification

G. Conclusion

In this part, we introduced our pipeline and described each module in detail. We described each feature selection method Information Gain, Gain Ratio, Chi2, and Relief. We also introduced all the evaluation metrics we use to measure how our models perform.

In the next section, we show the results of applying all these modules to the three data sets. We select the best three models of this pipeline to be the base models (level-0) models of the stacking algorithm. these three selected models are Random Forest, Extra Trees, and XGBoost which are present as the base models of the stacking algorithm.

IV. Results and Discussion

A. Introduction

We first, train all 11 models on all features to better understand how our model performs on these data sets. This step gets a baseline accuracy for the next evaluation on a subset of features. Before applying feature selection we train all 11 models on all features, to select the most promising 3 models that could be used later as baseline models for the stacking algorithm.

Then we apply the feature selection on these 3 models to get the best subset of features for each data set independently and apply some regularization techniques for each model independently to prepare the models for the next step.

The next step is picking the most appropriate stacking model for each data set using these 3 models we prepared as base models (level-0 models) for the stacking algorithm. We try 3 different top models Logistic Regression, a multilayer perceptron (MLP) with two hidden layers of 20 neurons each, an output layer with just 1 neuron with a sigmoid activation function, and the third top model a Deeper Neural network (DNN) with 5 hidden layers with neurons 15, 15, 20, 20, and 20 neurons respectively and output layer is the same as the MLP.

In this section, we present all the results we obtained based on different metrics train accuracy, test accuracy, test f1 score, test precision, and test recall. We used cross-validation with 5 k-folds to get a robust result that describes each model in detail.

In the next subsection, we start with the results of the first step in our pipeline which is training all 11 models from single-based models, bagging, boosting, and stacking ensemble techniques without the features selection step.

B. Results of All Models Without Feature Selection

We start by training all the models on three collected data sets which are UNSW-NB15, NSL-KDD, and KDD Cup. For each data set, we apply all the models independently. We use cross-validation with 5 folds for obtaining robust results that we could rely on.

In the next subsections, we show the results of each model we use from Single-Based techniques, Bagging, Boosting and stacking ensemble algorithms, and we then depend on the test accuracy to select the best three models for further development steps of our pipeline.

1) Performance scores of single-based algorithm: We employ 6 different Single Based models, for each model we apply cross-validation with 5 folds. Tables I, II, and III, show the Train accuracy, test accuracy, test f1-score, test ROC AUC, test precision, and test recall for each model we employ.

The tables show that the Decision Tree is the best model for each data set we used, UNSW-NB15, KDD Cup, and NSL-KDD. It obtains 93.25% for the UNSW-NB15 test accuracy, 99.97% test accuracy for the KDD cup data set, and 99.81% test accuracy for the NSL-KDD data set.

In this evaluation, we use all features to select the best models suitable for more evaluation using features selection. Based on results obtained from Single-Based we should try ensemble algorithms using the Decision Tree model as a base model for bagging, boosting, and stacking to get better results.

2) Performance scores of bagging algorithm: We use two bagging models that depend on the Decision tree as the base model for the Bagging algorithms the more common and efficient models are Random Forests and Extra Trees. As seen in Table IV, Random Forest obtained a 96.15% test accuracy on all features for the UNSW-NB15 data set. This accuracy is high compared to the single base model alone it reduces the error by 2.9% compared to the Decision Tree.

TABLE I
PERFORMANCE SCORES OF SINGLE BASED ALGORITHM ON UNSW-NB15 DATA SET USING ALL FEATURES (48 FEATURES)

Model	Train accuracy	Test accuracy	Test F1	Test ROC AUC	Test Precision	Test Recall
DecisionTree	99.99%	93.25%	93.99%	93.1%	93.63%	94.65%
SVC	91.1%	87.73%	89.11%	96.55%	89.47%	89.63%
LogisticRegression	91.5%	88.33%	89.62%	96.79%	90.01%	90.07%
Lasso	91.76%	88.11%	89.55%	96.7%	89.59%	90.57%
Ridge	92.94%	90.56%	91.67%	97.48%	91.48%	92.48%
ElasticNet	91.76%	89.68%	90.70%	96.98%	91.73%	90.23%

TABLE II
PERFORMANCE SCORES OF SINGLE-BASED ALGORITHM ON KDD CUP 99 DATA SET USING ALL FEATURES (49 FEATURES)

Model	Train accuracy	Test accuracy	Test Precision	Test Recall	Test F1
DecisionTree	99.99%	99.97%	99.94%	99.95%	99.95%
SVC	99.54%	99.55%	99.36%	98.79%	99.07%
LogisticRegression	99.54%	99.56%	99.42%	98.76%	99.09%
Lasso	99.60%	99.62%	99.60%	98.84%	99.22%
Ridge	99.59%	99.60%	99.23%	99.13%	99.18%
ElasticNet	99.62%	99.65%	99.57%	98.98%	99.28%

TABLE III
PERFORMANCE SCORES OF SINGLE BASED ALGORITHM ON NSL-KDD DATA SET USING ALL FEATURES (51 FEATURES)

Model	Train accuracy	Test accuracy	Test F1	Test ROC AUC	Test Precision	Test Recall
DecisionTree	99.99%	99.81%	99.80%	99.81%	99.79%	99.80%
SVC	96.94%	96.91%	96.66%	99.49%	97.40%	95.93%
LogisticRegression	96.91%	96.89%	96.63%	99.53%	97.35%	95.93%
Lasso	97.37%	97.36%	97.14%	99.35%	97.78%	96.52%
Ridge	97.27%	97.25%	97.02%	99.00%	97.65%	96.40%
ElasticNet	97.31%	97.32%	97.10%	99.28%	97.88%	96.33%

As seen in Tables V, and VI, for other data sets the accuracy does not differ too much from the Decision Tree in the single-based learning but just reduces the error by 0.01% for the KDD cup data set, and 0.06% for NSL-KDD.

3) Performance scores of boosting algorithm: For Boosting algorithms, we employ three models that use the Decision Tree model in its base structure. We employ Gradient Boosting, XGBoost, and AdaBoost models. As seen in Table VII, XGBoost outperforms all other models for the three data sets. It reduces the error of Random Forest by 0.66%, it gets an accuracy of 96.33% on the test set of UNSW-NB15.

XGBoost could achieve great accuracy as well on the other data set as shown in Tables VIII, and IX, it gets 99.92% accuracy on the NSL-KDD data set, and for the KDD cup, it gets 99.98% the same as Random Forest and Extra Trees.

From our results on the three data sets using all ensemble methods and single-based methods, we select the best 3 models which are suitable for the stacking algorithm to present the base models (level-0 models), and for applying the feature selection methods.

The three models Random Forest, Extra Trees, and XGBoost are the best models that get comparable accuracy on the data sets we used. We can use these models for more evaluation of features selection methods, and for Regularization to get the best test accuracy we can achieve. and prepare these models to be base models (level-0 models) of the stacking algorithms.

In the next subsection, we apply the feature selection algorithms to select the best subset of features for each data set independently, and train stacking algorithms on the best subset of features with the best 3 models obtained from our first step as base models.

C. Feature Selection

We apply 10 thresholds between quantization of 10% to 90% of the scores. We loop over these 10 thresholds for each feature selection method Information Gain, Gain Ratio, Chi2, and Relief. In every loop, we train the three models that give us the best result from the first step we employ these models Random Forest, Extra Trees, and XGBoost. These give us $10(thresholds) * 4(feature - selection) * 3(models) = 120$ training times using cross-validation with 5 folds.

We apply our loop for each data set independently, Tables X, XI, and XII, show the best subset of features for each data set UNSW-NB15, NSL-KDD, and KDD Cup 99 respectively.

TABLE IV
PERFORMANCE SCORES OF BAGGING ALGORITHM ON UNSW-NB15 DATA SET USING ALL FEATURES (48 FEATURES)

Model	Train accuracy	Test accuracy	Test F1	Test ROC AUC	Test Precision	Test Recall
RandomForest	99.99%	95.77%	96.15%	99.32%	96.67%	95.77%
ExtraTrees	99.99%	95.59%	95.98%	99.37%	96.65%	95.46%

TABLE V
PERFORMANCE SCORES OF BAGGING ALGORITHM ON KDD CUP 99 DATA SET USING ALL FEATURES (49 FEATURES)

Model	Train accuracy	Test accuracy	Test Precision	Test Recall	Test F1
RandomForest	99.99%	99.98%	99.99%	99.94%	99.97%
ExtraTrees	99.99%	99.98%	99.97%	99.94%	99.96%

TABLE VI
PERFORMANCE SCORES OF BAGGING ALGORITHM ON NSL-KDD DATA SET USING ALL FEATURES (51 FEATURES)

Model	Train accuracy	Test accuracy	Test F1	Test ROC AUC	Test Precision	Test Recall
RandomForest	99.99%	99.90%	99.89%	99.99%	99.94%	99.83%
ExtraTrees	99.99%	99.87%	99.86%	99.99%	99.91%	99.81%

TABLE VII
PERFORMANCE SCORES OF BOOSTING ALGORITHM ON UNSW-NB15 DATA SET USING ALL FEATURES (48 FEATURES)

Model	Train accuracy	Test accuracy	Test F1	Test ROC AUC	Test Precision	Test Recall
GradientBoosting	96.25%	93.72%	94.96%	99.15%	95.34%	93.52%
XGBoost	99.13%	96.33%	96.65%	99.55%	97.04%	96.4%
AdaBoost	95.23%	93.95%	94.58%	98.92%	94.45%	94.96%

For the UNSW-NB15 data set Relief feature selection method gets the best result using only 15 features as in Table X, for the KDD Cup 99 data set Information gain feature selection method gets the highest result on this data set using only 12 features, and for the NSL-KDD data set Chi2 gets the highest result using 12 features.

We now finish the second step of our pipeline and select the best subset of features that represent each data set independently, also select the best three models that achieve the highest accuracy to represent the base models of the stacking algorithm.

In the next subsection, we apply our stacking proposed approach using these subsets of features for each data set and use the best 3 models selected from our first step for the level-0 models of the stacking algorithm.

TABLE X
RELIEF FEATURE SELECTION ALGORITHM BEST-SELECTED SUBSET OF FEATURES FOR UNSW-NB15 DATA SET

Num	Feature Name	Score
1	dttl	0.147692
2	service=dns	0.128000
3	service=-	0.128000
4	sttl	0.108235
5	ct_dst_sport_ltm	0.056973
6	ct_state_ttl	0.053667
7	ct_src_ltm	0.050508
8	smean	0.049118
9	ct_dst_ltm	0.044621
10	ct_src_dport_ltm	0.043414
11	dload	0.043035
12	ct_srv_src	0.042290
13	ct_srv_dst	0.041049
14	dmean	0.037801
15	ct_dst_src_ltm	0.035290

TABLE VIII
PERFORMANCE SCORES OF BOOSTING ALGORITHM ON KDD CUP 99 DATA SET USING ALL FEATURES (49 FEATURES)

Model	Train accuracy	Test accuracy	Test Precision	Test Recall	Test F1
GradientBoosting	99.94%	99.95%	99.96%	99.83%	99.89%
XGBoost	99.99%	99.98%	99.99%	99.96%	99.97%
AdaBoost	99.77%	99.77%	99.72%	99.36%	99.54%

TABLE IX
PERFORMANCE SCORES OF BOOSTING ALGORITHM ON NSL-KDD DATA SET USING ALL FEATURES (51 FEATURES)

Model	Train accuracy	Test accuracy	Test F1	Test ROC AUC	Test Precision	Test Recall
GradientBoosting	99.59%	99.56%	99.53%	99.97%	99.56%	99.50%
XGBoost	99.99%	99.92%	99.91%	99.99%	99.99%	99.90%
AdaBoost	99.00%	98.97%	98.89%	99.94%	99.15%	98.63%

TABLE XI
INFO. GAIN FEATURE SELECTION ALGORITHM BEST-SELECTED SUBSET OF FEATURES FOR KDD CUP 99 DATA SET

Num	Feature Name	Score
1	diff_srv_rate	0.626262
2	same_srv_rate	0.608975
3	src_bytes	0.568237
4	flag=SF	0.564254
5	count	0.558425
6	dst_bytes	0.513371
7	dst_host_srv_serror_rate	0.512426
8	srv_serror_rate	0.508108
9	flag=S0	0.507986
10	serror_rate	0.505596
11	dst_host_serror_rate	0.505240
12	dst_host_same_srv_rate	0.490741

TABLE XII
CHI2 FEATURE SELECTION ALGORITHM BEST-SELECTED SUBSET OF FEATURES FOR NSL-KDD DATA SET

Num	Feature Name	Score
1	diff_srv_rate	86910.52
2	dst_bytes	85616.03
3	serror_rate	73702.59
4	srv_serror_rate	71790.60
5	dst_host$_serror\_rate$	65551.50
6	dst_host_srv_serror_rate	64069.77
7	src_bytes	63004.26
8	dst_host_srv_count	49919.47
9	dst_host_same_srv_rate	47408.31
10	dst_host_diff_srv_rate	40685.25
11	flag=S0	38523.44
12	logged_in	36259.14

D. Performance Scores of Proposed Stacking Algorithm

In this part, we train the 3 models Random Forest, Extra Trees, and XGBoost on the selected 15 features for the UNSW-NB15 data set using cross-validation. Then we train the stacking models using these models as a base model, and Logistic Regression as a meta-classifier model. The stacking model achieves 97.79% test accuracy using only 15 features which outperforms the three models independently and all the baseline results for the UNSW-NB15 data set.

We then apply another two stacking models with the same base models but using different meta-classifier models multilayer perceptron (MLP), and Deep Neural Network (DNN). One uses Neural networks NN with two hidden layers of 20 neurons each and the other (DNN) uses 5 hidden layers of 15, 15, 20, 20, and 20 neurons respectively. From Table XIII, we can easily notice that the accuracy doesn't improve too much but the complexity of the model increases too much compared to the logistic regression model used before as a meta-classifier model.

In deployment, if we don't care about 0.01% or 0.02% improvement in accuracy but we care more about the inference time, we can use the stacking model with Logistic Regression as a meta-classifier to save the time of prediction and increase the inference time.

An efficient way to assess the effectiveness of predicted tests is to use the receiver operating characteristic (ROC) curve, which is defined as a plot of test sensitivity as the y coordinate vs its 1-specificity or false positive rate (FPR) as the x coordinate. Figure 16, shows that the three stacking models have a very close AUC metric. That means that the stacking+LR is the best one because the complexity of the model is not so high compared to the other stacking models which have a Neural network as a top model which increases the complexity of the model so much.

Table XIV, shows that using only 12 features, XGBoost gets the best results for the KDDCup 99 data set. Using the Information Gain feature selection method, the model achieves 99.98% test accuracy. Additionally, the Stacking + LR model for the NSL-KDD data set achieves 99.87% test accuracy with only 12 features through the Chi2 feature selection approach.

The stacking algorithm for the KDDCup 99 does not improve the accuracy of the XGBoost. It seems the XGBoost is the most appropriate model for this data set because of the simplicity of this data, which does not require a more complex model like a stacking algorithm to achieve higher accuracy.

We calculate the confusion matrices of all the stacking models we employ to better understand the error distribution of the models and get a better intuition of the models' performance. Figures 17, 18, and 19, show the confusion matrices of the stacking models on UNSW-NB15, KDD Cup, and NSL-KDD data sets respectively. For KDD Cup we employ only two stacking models using different meta classifiers one uses Logistic Regression, and the other uses multilayer perceptron (MLP) with only two hidden layers, but for other data sets we use three stacking models using three different meta-classifier same as MLP, and the third one is Deep Neural network (DNN) with 5 hidden layers.

Performance doesn't improve on KDD Cup using a more complex meta-classifier so, we use only Logistic Regression, and neural network (MLP) to represent the level-1 model.

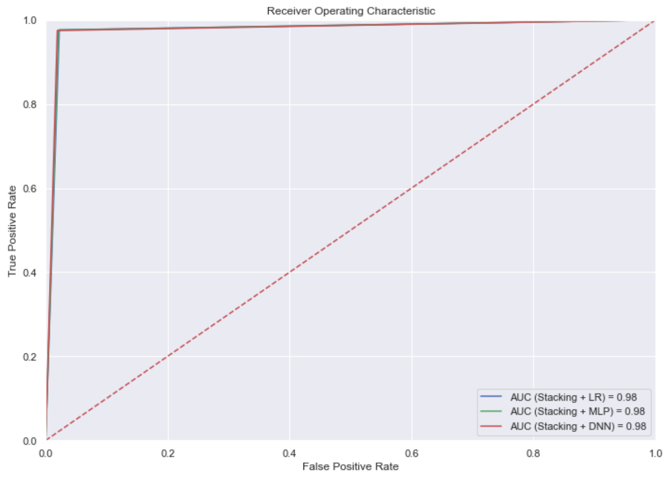


Fig. 14. ROC curve for the stacking models on the UNSW-NB15 data set

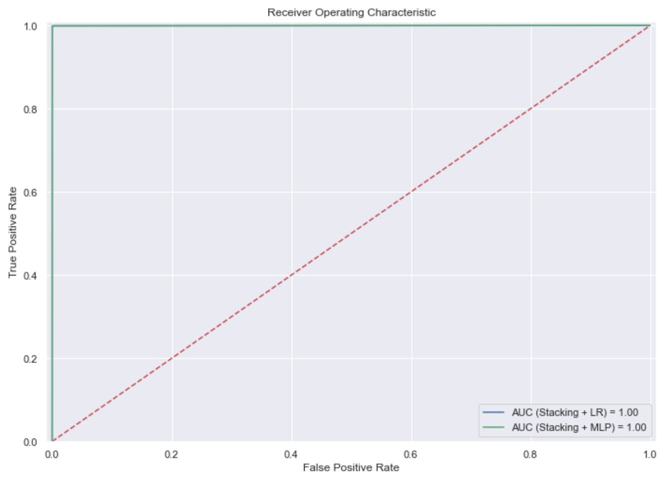


Fig. 15. ROC curve for the stacking models on the KDD Cup 99 data set

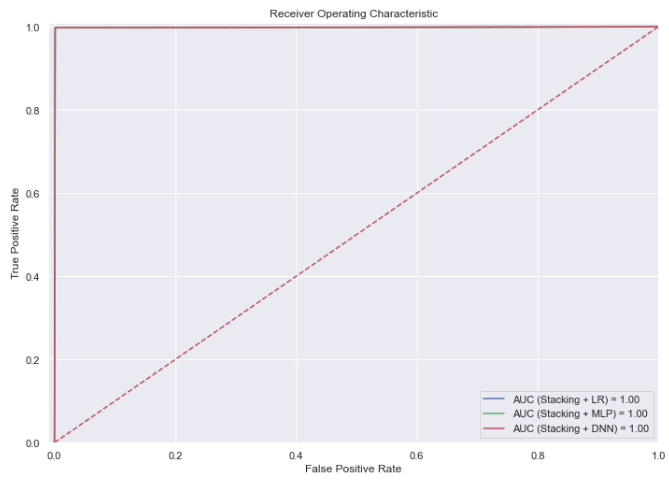


Fig. 16. ROC curve for the stacking models on the NSL-KDD data set

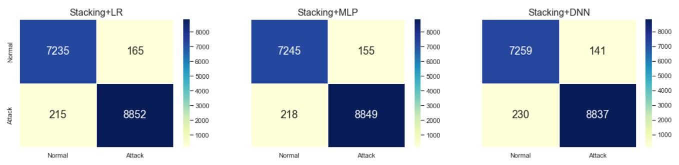


Fig. 17. Confusion matrices of the stacking models on UNSW-NB15 data set

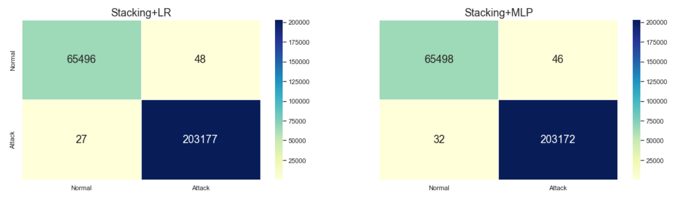


Fig. 18. Confusion matrices of the stacking models on KDD Cup 99 data set

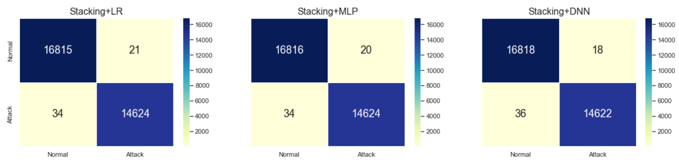


Fig. 19. Confusion matrices of the stacking models on NSL-KDD data set

E. Robustness of Our Approach

Our Stacking approach achieved great performance compared to the other approaches. We apply the single-based method, Bagging, Boosting, and Stacking algorithms, and proved how the stacking algorithm is the best algorithm that should be taken into consideration while solving the network intrusion detection problem. Table XV compares our results of the stacking approach with other state-of-the-art models that other researchers utilize. It's obvious that our stacking approach outperforms all other results, which is why the stacking approach should be considered for solving the

TABLE XIII
PERFORMANCE SCORES OF PROPOSED STACKING ALGORITHM ON UNSW-NB15 DATA SET USING THE BEST SUBSET OF FEATURES (15 SELECTED FEATURES)

Model	Train accuracy	Test accuracy	Test F1	Test ROC AUC	Test Precision	Test Recall
RandomForest	98.96%	94.73%	95.02%	99.47%	96.47%	94.11%
ExtraTrees	99.99%	95.59%	95.98%	99.37%	96.65%	95.46%
XGBoost	98.56%	95.45%	95.81%	99.51%	96.6%	95.29%
Stacking + LR	98.78%	97.79%	97.99%	99.77%	98.17%	97.82%
Stacking + MLP	98.82%	97.81%	98.0%	99.77%	98.29%	97.72%
Stacking + DNN	98.84%	97.82%	98.02%	99.77%	98.4%	97.64%
MLP refers to a Neural network with two layers of 20 neurons each. DNN refers to a Deeper neural network with 5 layers [15]*2, and [20]*3 neurons.						

TABLE XIV
PERFORMANCE SCORES OF PROPOSED STACKING ALGORITHM ON THE OTHER DATA SETS USING THE BEST SUBSET OF FEATURES

Model	Data	Num. of Features	Feature Selection Method	Train accuracy	Test accuracy	Test F1
XGBoost	KDDCup 99	12	Info.Gain	99.99%	99.98%	99.95%
Stacking + LR	KDDCup 99	12	Info.Gain	99.99%	99.98%	99.95%
Stacking + LR	NSL-KDD	12	Chi2	99.95%	99.87%	99.81%

Network Intrusion Detection problem.

TABLE XV
STUDIES ON COMPUTER NETWORK CLASSIFICATION PROPOSED BY DIFFERENT RESEARCHERS

Model	Data set	Accuracy
Al-Yaseen et al. 2017 [3]	KDDCup99	95.75%
Ahmed et al. 2021 [2]	KDDCup99	97.69%
Bhati et al. 2021 [9]	KDDCup99	99.95%
Belavagi and Muniyal 2016 [7]	NSL_KDD	99.00%
Gaikwad and Thool 2019b [19]	NSL_KDD	99.72%
Gaikwad and Thool 2019a [18]	NSL_KDD	99.66%
Jabbar and Samreen 2016[26]	NSL_KDD	99.15%
Nawir et al. 2018[46]	UNSW-NB15	94.37%
Rajagopal et al. 2020[52]	UNSW-NB15	94.00%
Ours	KDDCup99	99.98%
Ours	NSL_KDD	99.87%
Ours	UNSW-NB15	98.84%

V. CONCLUSION AND FURTHER DIRECTION

A. Outline of the Contributions

In this part, we outline our thesis and talk about the limitations of our work and how we can improve the work in the future. We also present an overall conclusion of our work and present the best models we achieved for each data set we used. We talked about what motivated us in the first place to start the problem of network intrusion detection and stop attack communication accurately with comparable results with other researchers and try to solve the problem using different perspectives to reach the best model and best subset of features that are suitable for deployment.

- We employ different machine learning algorithms among bagging, boosting, and stacking ensemble techniques on three different data sets UNSW-NB15, NSL-KDD, and KDD Cup 99.

- We employ four feature selection algorithms Information Gain, Gain Ratio, Chi2, and Relief. We select the most appropriate subset of features for each data set independently using the best feature selection method among these four methods.

- We develop a stacking approach that could achieve the best result on the three data sets. Depending on the best subset of features and the best three models selected from these ML algorithms we employ.

- We employ different meta-classifier models for the stacking algorithm to select the best appropriate meta-classifier model.

B. Overall Conclusion

We introduce a new approach using stacking models and solve the problem of network intrusion detection from another perspective using different feature selection methods we could reach the best subset of features using the best feature selection method Information Gain, Gain Ratio, Chi2, or Relief.

Also, we could choose the best 3 models among 11 different models. The 11 learning models we employed include Decision-Tree, RandomForest, ExtraTrees, GradientBoosting, XGBoost, AdaBoost, SVC, Logistic Regression, Lasso, Ridge, and ElasticNet. After selecting the best three models we apply regularization for each model independently and use them as base models for the stacking models.

We employ multiple meta-classifier models using Logistic Regression, Multilayer perceptron (MLP), and Deeper Neural network (DNN) . One uses Multilayer perceptron (MLP) with two hidden layers of 20 neurons each and the other

(DNN) uses 5 hidden layers of 15, 15, 20, 20, and 20 neurons respectively for selecting the meta-classier model of the stacking models. Depending on the results obtained we select the highest model accuracy with the best subset of features to be the output of our pipeline and ready for deployment or to improve in the future.

The stacking model achieves 97.79% test accuracy on the UNSW-NB15 data set using only 15 features with Random Forest, Extra Trees, and XGBoost as base models (level-0 models) and Logistic Regression as a meta-classifier model. By changing the top model to DNN we achieve 97.82% test accuracy, but the complexity of the stacking model is increased too much to improve the accuracy with only 0.03% this note should be considered when choosing the model for deployment which will affect the inference time of the stacking model.

For KDDCup 99 data set XGBoost has the best results using only 12 features, the model achieves 99.98% test accuracy using the Information gain feature selection method. And for the NSL-KDD data set Stacking + LR model achieves 99.87% test accuracy using just 12 features by Chi2 as the feature selection method.

References

[1] David W Aha, Dennis Kibler, and Marc K Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.

[2] Muhammad Ahmad, Qaiser Riaz, Muhammad Zeeshan, Hasan Tahir, Syed Ali Haider, and Muhammad Safeer Khan. Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using unsw-nb15 data-set. *EURASIP Journal on Wireless Communications and Networking*, 2021(1):1–23, 2021.

[3] Wathiq Laftah Al-Yaseen, Zulaiha Ali Othman, and Mohd Zakree Ahmad Nazri. Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system. *Expert Systems with Applications*, 67:296–303, 2017.

[4] T Aldhyani and Manish R Joshi. Analysis of dimensionality reduction in intrusion detection. *International Journal of Computational Intelligence and Informatics*, 4(3):199–206, 2014.

[5] Alexandre Alves. Stacking machine learning classifiers to identify higgs bosons at the lhc. *Journal of Instrumentation*, 12(05):T05005, 2017.

[6] Ahmad Azab, Mamoun Alazab, and Mahdi Aiash. Machine learning based botnet identification traffic. *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016.

[7] Manjula C Belavagi and Balachandra Muniyal. Performance evaluation of supervised machine learning algorithms for intrusion detection. *Procedia Computer Science*, 89:117–123, 2016.

[8] Sunita Beniwal and Jitender Arora. Classification and feature selection techniques in data mining. *International journal of engineering research & technology (ijert)*, 1(6):1–6, 2012.

[9] Bhoopesh Singh Bhati, Garvit Chugh, Fadi Al-Turjman, and Nitesh Singh Bhati. An improved ensemble-based intrusion detection technique using xgboost. *Transactions on emerging telecommunications technologies*, 32(6):e4076, 2021.

[10] Jason Brownlee. Ensemble machine learning algorithms in python with scikit-learn, Aug 2020.

[11] Jason Brownlee. Stacking ensemble machine learning with python, Apr 2021.

[12] James P Callan, Tom Fawcett, and Edwina L Rissland. Cabot: An adaptive approach to case-based search. In *Ijcai*, volume 12, pages 803–808, 1991.

[13] Yu-Shu Chen and Yi-Ming Chen. Combining incremental hidden markov model and adaboost algorithm for anomaly intrusion detection. In *Proceedings of the ACM SIGKDD Workshop on Cybersecurity and intelligence informatics*, pages 3–9, 2009.

[14] C Cortes and V Vapnik. Support-vector networks. mach. leaming 20, 273–297, 1995.

[15] KDD Cup. Kdd cup 99 data set available on: http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html, 2007.

[16] Dorothy E Denning. An intrusion-detection model. *IEEE Transactions on software engineering*, (2):222–232, 1987.

[17] Naoual El Aboudi and Laila Benhlima. Review on wrapper feature selection approaches. In *2016 International Conference on Engineering & MIS (ICEMIS)*, pages 1–5. IEEE, 2016.

[18] D.P. Gaikwad and Ravindra C. Thool. Intrusion detection system using bagging ensemble method of machine learning. In *2015 International Conference on Computing Communication Control and Automation*, pages 291–295, 2015.

[19] DP Gaikwad and Ravindra C Thool. Intrusion detection system using bagging with partial decision treebase classifier. *Procedia Computer Science*, 49:92–98, 2015.

[20] Joseph L Gastwirth. The estimation of the lorenz curve and gini index. *The review of economics and statistics*, pages 306–316, 1972.

[21] Nicholas Gray and Scott Ferson. Logistic regression through the veil of imprecise data. *arXiv preprint arXiv:2106.00492*, 2021.

[22] Alind Gupta. Ml— extra tree classifier for feature selection, 2020.

[23] Jiankun Hu, Xinghuo Yu, Dong Qiu, and Hsiao-Hwa Chen. A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection. *IEEE network*, 23(1):42–47, 2009.

[24] Weiming Hu, Wei Hu, and Steve Maybank. Adaboost-based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(2):577–583, 2008.

[25] KDD ieee. Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), 2015.

[26] M. A. Jabbar and Shirina Samreen. Intelligent network intrusion detection using alternating decision trees. In *2016 International Conference on Circuits, Controls, Communications and Computing (I4C)*, pages 1–6, 2016.

[27] Julian Jang-Jaccard and Surya Nepal. A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*, 80(5):973–993, 2014.

[28] Alan Jović, Karla Brkić, and Nikola Bogunović. A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1200–1205. Ieee, 2015.

[29] Asha Gowda Karegowda, AS Manjunath, and MA Jayaram. Comparative study of attribute selection using gain ratio and correlation-based feature selection. *International Journal of Information Technology and Knowledge Management*, 2(2):271–277, 2010.

[30] John T Kent. Information gain and a general measure of correlation. *Biometrika*, 70(1):163–173, 1983.

[31] Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Machine learning proceedings 1992*, pages 249–256. Elsevier, 1992.

[32] Kenji Kira, Larry A Rendell, et al. The feature selection problem: Traditional methods and a new algorithm. In *Aaai*, volume 2, pages 129–134, 1992.

[33] Igor Kononenko. Estimating attributes: Analysis and extensions of relief. In *European conference on machine learning*, pages 171–182. Springer, 1994.

[34] S Vanaja K Ramesh Kumar. Analysis of feature selection algorithms on classification: a survey. 2014.

[35] Gudrun Lange, John DeLuca, Joseph A Maldjian, Huey-Jen Lee, Lana A Tiersky, and Benjamin H Natelson. Brain mri abnormalities exist in a subset of patients with chronic fatigue syndrome. *Journal of the neurological sciences*, 171(1):3–7, 1999.

[36] S Latha and Sinthu Janita Prakash. A survey on network attacks and intrusion detection systems. In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–7. IEEE, 2017.

[37] Aleksandar Lazarevic, Vipin Kumar, and Jaideep Srivastava. Intrusion detection: A survey. In *Managing cyber threats*, pages 19–78. Springer, 2005.

[38] Riccardo Leardi, R Boggia, and M Terrile. Genetic algorithms as a strategy for feature selection. *Journal of chemometrics*, 6(5):267–281, 1992.

[39] Changki Lee and Gary Geunbae Lee. Information gain and divergence-based feature selection for machine learning-based text categorization. *Information processing & management*, 42(1):155–165, 2006.

[40] Weiwei Lin, Ziming Wu, Longxin Lin, Angzhan Wen, and Jin Li. An ensemble random forest algorithm for insurance big data analysis. *Ieee access*, 5:16568–16575, 2017.

[41] Huan Liu and Rudy Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of 7th IEEE international conference on tools with artificial intelligence*, pages 388–391. IEEE, 1995.

[42] Huan Liu and Rudy Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of 7th IEEE international conference on tools with artificial intelligence*, pages 388–391. IEEE, 1995.

[43] meAjitesh Kumar. Ridge classification concepts amp; python examples, Oct 2022.

[44] Biswanath Mukherjee, L Todd Heberlein, and Karl N Levitt. Network intrusion detection. *IEEE network*, 8(3):26–41, 1994.

[45] Srinivas Mukkamala, Andrew H Sung, and Ajith Abraham. Intrusion detection using an ensemble of intelligent paradigms. *Journal of network and computer applications*, 28(2):167–182, 2005.

[46] Mukrimah Nawir, Amiza Amir, Ong Bi Lynn, Naimah Yaakob, and R Badlishah Ahmad. Performances of machine learning algorithms for binary classification of network anomaly detection system. In *Journal of Physics: Conference Series*, volume 1018, page 012015. IOP Publishing, 2018.

[47] Bhuvaneswari Amma N.G. and Selvakumar S. Deep radial intelligence with cumulative incarnation approach for detecting denial of service attacks, Mar 2019.

[48] Stephen Northcutt and Judy Novak. *Network intrusion detection*. Sams Publishing, 2002.

[49] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[50] Mrutyunjaya Panda, Ajith Abraham, and Manas Ranjan Patra. A hybrid intelligent approach for network intrusion detection. *Procedia Engineering*, 30:1–9, 2012.

[51] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[52] Smitha Rajagopal, Poornima Panduranga Kundapur, and Katiganere Siddaramappa Hareesha. A stacking ensemble for network intrusion detection using heterogeneous datasets. *Security and Communication Networks*, 2020, 2020.

[53] The Click Reader. Decision tree classifier, Oct 2021.

[54] Noelia Sánchez-Maroño, Amparo Alonso-Betanzos, and María Tombilla-Sanromán. Filter methods for feature selection–a comparative study. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 178–187. Springer, 2007.

[55] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[56] Aakanksha Sharaff and Harshil Gupta. Extra-tree classifier with metaheuristics approach for email classification. In *Advances in computer communication and computational sciences*, pages 189–197. Springer, 2019.

[57] Raman Singh, Harish Kumar, Ravinder Kumar Singla,

and Ramachandran Ramkumar Ketti. Internet attacks and intrusion detection system: A review of the literature. *Online Information Review*, 2017.

[58] David B Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Machine Learning Proceedings 1994*, pages 293–301. Elsevier, 1994.

[59] Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282, 2011.

[60] KDD uni. Nslkdd. available on: http://nsl.cs.unb.ca/nslkdd/, 2009., 2009.

[61] Ryan J Urbanowicz, Melissa Meeker, William La Cava, Randal S Olson, and Jason H Moore. Relief-based feature selection: Introduction and review. *Journal of biomedical informatics*, 85:189–203, 2018.

[62] Alexander Vezhnevets and Olga Barinova. Avoiding boosting overfitting by removing confusing samples. In *European Conference on Machine Learning*, pages 430–441. Springer, 2007.

[63] Wikipedia. Wikipedia, support vector machine, Oct 2022.

[64] Yuhua Yin, Julian Jang-Jaccard, Wen Xu, Amardeep Singh, Jinting Zhu, Fariza Sabrina, and Jin Kwak. Igrf-rfe: A hybrid feature selection method for mlp-based network intrusion detection on unsw-nb15 dataset. *arXiv preprint arXiv:2203.16365*, 2022.

[65] Ying Zhou, Thomas A Mazzuchi, and Shahram Sarkani. M-adaboost-a based ensemble system for network intrusion detection. *Expert Systems with Applications*, 162:113864, 2020.

[66] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.