Journal of Cybersecurity and Privacy

MDPI

Article

Enhancing Machine Learning Prediction in Cybersecurity Using Dynamic Feature Selector

Mostofa Ahsan [1,*], Rahul Gomes [2,*], Md. Minhaz Chowdhury [3] and Kendall E. Nygard [1]

[1] Department of Computer Science, North Dakota State University, Fargo, ND 58102, USA; kendall.nygard@ndsu.edu
[2] Department of Computer Science, University of Wisconsin-Eau Claire, Eau Claire, WI 54701, USA
[3] Department of Computer Science, East Stroudsburg University of Pennsylvania, East Stroudsburg, PA 18301, USA; mchowdhur1@esu.edu
* Correspondence: mostofa.ahsan@ndsu.edu (M.A.); gomesr@uwec.edu (R.G.)

Received: 29 January 2021; Accepted: 15 March 2021; Published: 21 March 2021

Abstract: Machine learning algorithms are becoming very efficient in intrusion detection systems with their real time response and adaptive learning process. A robust machine learning model can be deployed for anomaly detection by using a comprehensive dataset with multiple attack types. Nowadays datasets contain many attributes. Such high dimensionality of datasets poses a significant challenge to information extraction in terms of time and space complexity. Moreover, having so many attributes may be a hindrance towards creation of a decision boundary due to noise in the dataset. Large scale data with redundant or insignificant features increases the computational time and often decreases goodness of fit which is a critical issue in cybersecurity. In this research, we have proposed and implemented an efficient feature selection algorithm to filter insignificant variables. Our proposed Dynamic Feature Selector (DFS) uses statistical analysis and feature importance tests to reduce model complexity and improve prediction accuracy. To evaluate DFS, we conducted experiments on two datasets used for cybersecurity research namely Network Security Laboratory (NSL-KDD) and University of New South Wales (UNSW-NB15). In the meta-learning stage, four algorithms were compared namely Bidirectional Long Short-Term Memory (Bi-LSTM), Gated Recurrent Units, Random Forest and a proposed Convolutional Neural Network and Long Short-Term Memory (CNN-LSTM) for accuracy estimation. For NSL-KDD, experiments revealed an increment in accuracy from 99.54% to 99.64% while reducing feature size of one-hot encoded features from 123 to 50. In UNSW-NB15 we observed an increase in accuracy from 90.98% to 92.46% while reducing feature size from 196 to 47. The proposed approach is thus able to achieve higher accuracy while significantly lowering number of features required for processing.

Keywords: dynamic feature selection; meta-learner; cybersecurity; random forest; CNN; RNN; GRU; LSTM; Bi-LSTM

1. Introduction

The ability to learn and adapt has made machine learning techniques mainstream in cybersecurity. Training a model with a comprehensive dataset having multiple attack types is a key to improve anomaly detection performance [1]. However, issues like high dimensionality of datasets pose a significant threat for most of these techniques hindering real-time response and deployment on legacy systems. Manually reducing feature size is a key solution to lower the computational complexity. Removing insignificant features while retaining the important ones is a trade-off which can reduce complexity but compromise predictive performance of the model to some extent. Traditional feature selection models classified into two groups mainly wrappers and filters can be useful in feature

filtering [2]. Wrappers generate several feature subsets and select the one which generates the best result. Wrapper methods however require significant computation time to generate good results. The filter method employs various informative features such as entropy or information gain to decide which features yield the best output [3]. The drawback is the heuristic method of selecting these features. Datasets with certain degree of non-linearity may suffer from the possibility of reduced performance after feature filtering. In this paper, we explore a dynamic feature selector (DFS) for processing cybersecurity datasets. We combine several measurement indicators along with a meta-learner bagging ensemble-based approach to generate group of features that achieve the best performance in malware detection and deliver superior accuracy across datasets pertaining to the cybersecurity domain.

2. Machine Learning and Cybersecurity

Classification of cybersecurity datasets can be accomplished by several techniques. Two of the most common ones used are bagging and boosting. Bagging is an ensemble learning method and is the process of generating multiple versions of a predictor by resampling the training data and later aggregating those predictors to get a stable predictor [4]. A bootstrap sample also known as a bootstrap replica is generated by extracting a sample of observations from training datasets. Each bootstrap replica is generated by the process of sampling with replacement due to which some of the data from the training dataset may never appear in the training and some may appear multiple times [5]. The goal of this resampling is to reduce the variation in the prediction process. Hence, over-fitting of data is minimized by bagging. Bagging was used for the classification of NSL-KDD test dataset by voting in [6]. The classification process used random tree and the accuracy by bagging was highest among the other methods at the test phase. There are several research works in this domain that used, NSL-KDD and UNSW-NB15 datasets either separately [7–11] or together for comparative analysis [12–17]. A partial list of where these two datasets and their predecessors were used was presented in [17]. Both these datasets provided features which enable deployment of successful classification systems.

Boosting is also an ensemble learning method. It ensembles weak prediction models for its prediction or classification purpose. Gradient boosting algorithm is a boosting method and Extreme Gradient Boost or XGBoost is a type of Gradient boosting algorithm. It has fast learning speed and high accuracy [18]. Both bagging and boosting methods were used on UNSW-NB15 dataset to demonstrate the performance of classifier-based intrusion detection systems [19]. In this comparative analysis, boosting method performed better than the bagging method by significantly reducing the number of false positives. Similarly, boosting method outperformed the bagging method in improving the classifier's performance. In [20], both bagging and boosting methods were also used with specific features in the NSL-KDD datasets that resembled sensor node attack of Internet of Things (IoT). Eleven machine learning algorithms were used, and their malware detection performance were compared. The authors reported ensemble and tree-based methods to be most accurate. XGBoost, a tree-based ensemble method, outran all the other algorithms used in [20], with the accuracy of 97% in attack detection while the bagging technique had a high accuracy of 96.7%.

A wrapper method measures how useful a feature or a combination of features are in successfully classifying records in a dataset. The goodness of selected features is evaluated, via a learning algorithm [21]. These features are then passed (by the wrapper method) to a predictive model that evaluates the subset [22]. Since the wrapper method does not choose features based on correlation or other forms of univariate statistics, they are known to create good feature sets [21] that are validated based on performance. The wrapper method is superior than the other comparable methods in evaluating the goodness of the selected features [21–25]. This method can also be used to select the best subsets from the features. Wrapper method is also combined with other methods of feature selection and feature construction for an enhanced performance [24]. Some common wrapper methods include recursive feature selection algorithms or sequential feature selection algorithms. Wrapper

methods have been successfully applied in cybersecurity data mining for feature selection. A wrapper method based feature selection approach has previously been tested on National Security Laboratory NSL-KDD dataset [26]. The authors claimed that they did not use the Third Knowledge Discovery and Data Mining Tools Competition Dataset KDD-CUP as it is the earlier version of NSL-KDD data set and NSL-KDD is superior to KDD-CUP dataset. The authors reduced the features present in this dataset by a percentage close to 60%. In [13], the authors applied a Hybrid Filter-Wrapper feature selection method to detect distributed denial-of-service (DDoS) attack. During the application of this hybrid method, they claimed to reduce the number of features from 40 to 9 with a high accuracy of DDoS detection. Feature selection was applied on KDD dataset, but mostly on the KDD99 version rather NSL-KDD [27]. For example, a wrapper method's variation was used for feature selection using both KDD99 and UNSW-NB15 dataset [14]. To be noted, the authors used KDD99 instead of NSL-KDD. Their proposed method reduced the number of features to 18 and 20 respectively for KDD99 and UNSW-NB15.

Deep Learning [28,29] has also been successfully explored in this realm. A deep belief network was used for cyber attack detection using port scanning method where the model was tested using two security datasets UNSW-NB15 and NSL-KDD [16]. The proposed algorithm had a higher accuracy with low false positives. The authors mentioned about using the algorithm with real time malicious packets to verify the efficacy of these datasets and concluded that the NSL-KDD and UNSW-NB15 datasets have similar signatures to that of actual malware packets. In [17], an intrusion detection system, based on a random forest classifier, was developed, and tested using three datasets. Two of the datasets were UNSW-NB15 and NSL-KDD. These datasets have also been used to verify the performance of a network anomaly detector by analyzing network traffic in [15] using sparse-autoencoders. For UNSW-NB15 dataset, the anomaly detector's training phase used 206,138 records and the testing phase used 51,535 records. For the NSL-KDD dataset, there were 118,813 records for training and 29,704 records for testing. The detector performed better on the UNSW-NB15 dataset, compared to the NSL-KDD dataset. The authors mentioned that the reason of such performance is that UNSW-NB15 has more records.

To our knowledge there has been no significant work done on a dynamic feature selection approach for identifying malware from packet information to strengthen cybersecurity. In our approach, we combine different algorithms and use tuning parameters to generate feature subsets that could provide the best performance for malware detection. We then use employ a meta-learning approach to use the selected feature subsets and train multiple machine learning algorithms for accurate prediction. To ensure the viability of the proposed approach, we test the model on the NSL-KDD as well as UNSW-NB15 datasets. Experiments revealed that this new approach significantly outperforms standard classification algorithm such as Naïve Bayes. We also determine that removal of least important features significantly improves the prediction capability of the meta-learning algorithms while reducing complexity.

3. Algorithms Used for Dimensionality Reduction

Feature selection methods among other benefits contribute towards increasing classification accuracy [30]. It is helpful in reducing the number of irrelevant features that when included in the predictive model would increase computational complexity and training time but provide negligible or no increase in prediction accuracy [31]. A combination of methods for feature selection in this research has been discussed in this section.

3.1. Univariate Feature Selection

To select the most relevant features, univariate feature selection method utilizes univariate statistical tests to return a list of features which are ranked depending on the scoring function used. This can be an effective pre-processing step to retrieve the most significant features in a dataset that contribute to prediction accuracy significantly. To perform the feature selection process, a one-way ANOVA F-test was performed. Like Naïve Bayes Classifier, the one-way ANOVA (Analysis of variance)

F-test ensures that there is no relationship between the feature attributes used to accurately classify the dependent attribute. There is higher degree of variance if the means obtained from groups of data is different from the global mean derived from the dataset. Thus, it successfully returns the ratio of the inter-group to intra-group variability in a sample. ANOVA was selected over the T-test to give more stability and reduce the type 1 error while comparing population means of multiple groups. ANOVA is very effective in determining the difference of means of two or more groups at the same time. This also offers a significant advantage over the common T-test which conducts a repeating set of comparison between two attributes at a time [32]. Features with a score higher than a percentile value of 97 were considered useful in this analysis.

3.2. Correlated Feature Elimination

Another feature engineering method applied to reduce the dimensionality of the dataset was the elimination of highly correlated features. Consider two features a and b where a = $x_1 \ldots, x_n$ and b = $y_1 \ldots, y_n$. The degree of similarity between these two features can be represented using a correlation coefficient. The Pearson Correlation coefficient was used to express the correlation between features. It was the most logical choice due to two reasons. The complexity of Pearson Correlation is linear making it efficient [33,34]. Furthermore, the features on which Pearson Correlation was applied were mostly binary which reduces any drawbacks of having outliers. It is obtained as shown in Equation (1).

$$r = \frac{\sum_{i=1}^{n}(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=1}^{n}(a_i - \bar{a})^2 (b_i - \bar{b})^2}}. \quad (1)$$

Here, $\bar{a}$ and $\bar{b}$ denotes the mean of all records in features a and b respectively. The Pearson Correlation returns a value from -1 to 1. Values closer to 1 denote a high degree of correlation between the two features. In this research certain features which have a degree of collinearity greater than 0.8 with several other unique features were potential candidates for being dropped.

3.3. Gradient Boosting

Gradient boosting algorithms is an ensemble technique which works by fitting additive trees on top of existing decision trees and minimizing the line of steepest descent [35]. Gradient boosting algorithms especially XGBoost [18] are especially useful for classification due to their efficiency and scalability [36]. Boosting as the name suggests works by increasing the strength of weak learners. Once a decision tree is derived from preliminary classification of a dataset, the loss function for that tree is also calculated. This loss function is derived from the coefficients that are used to fit the model. In subsequent iterations, the model works to decrease this loss function by increasing the prediction accuracy for classification. For regression problems, the model tries to reduce the difference between the observed and predicted values. In this research, XGBoost was implemented on the two datasets. The algorithm was able to identify the features, which were highly important and contributed significantly towards classification. The trained model also returned a set of features that provided little or no contribution towards classification. This information was used in the later stages to exclude less important features for downstream analysis.

3.4. Information Gain

Information gain or mutual information refers to the probability theory of the mutual dependency between variables. In this scenario, it represents how much information can be obtained about the dependent feature from any of the independent features selected for evaluation. Information gain is a metric used in decision trees to evaluate how good a split has been made in a decision tree to classify the dataset. A higher information gain is derived when there is a pure split in a node [37]. A pure split denotes that the node is split in such a way that all the records belong to a single class. An impure split denotes a node that produces a split where the records are evenly distributed among all

the classes. An impure split is not useful since it is not able to classify records with higher accuracy [38]. Although information gain is a useful metric for evaluation, it suffers from cardinality where it favors attributes with larger number of values. To compensate for this problem, information gain ratio is used to decide a successful split [39]. It is a ratio of the Information Gain to Split Entropy. Split entropy is also referred to as the Intrinsic Value. Equation (2) denotes the Information Gain from X on Y. Here $H(Y)$ denotes the entropy of feature Y and $H(Y|x)$ denotes the entropy of Y given the attributes of feature x. Information gain ratio is obtained by dividing the information gain by intrinsic information.

$$IG(Y, x) = H(Y) - H(Y|x). \quad (2)$$

3.5. Wrapper Method Application

Wrapper methods were also included in the feature selection process. There are several benefits that wrapper methods provide in the feature selection process compared to filtering techniques offered by ANOVA and information gain used earlier. Since it evaluates all possible combinations of the features to determine their importance w.r.t other features, it reduces the chances of biases caused by filter methods such as the ANOVA which treats every feature as independent from another [40]. However, since it is a greedy approach, it suffers from being computationally intensive [40]. The machine learning algorithm used in the wrapper method was random forest classifier [41,42]. It works by creating multiple decision trees. These trees individually classify the records to belong to a certain class. However the final decision regarding which class the record belongs to is taken by majority of the votes from the decision trees. This offers a tremendous advantage over a single decision tree classifier, since there is a high possibility of a single decision tree to incorrectly classify a record compared to majority of decision trees. Random forest by default uses approximately 500 decision trees which increases its complexity. It also uses several features to see which combination of features yields better results. The two wrapper methods used in this research were forward selection and backward elimination. The training process can be stopped after a certain number of iterations have been completed or if the model stops finding any substantial increase in accuracy for a set number of iterations. Random forest also returns how important each feature is in building by computing the GINI importance value of each feature.

In the forward selection process, the algorithm begins with a single feature among all other features that produces the best classification result. Once that feature is selected, the algorithm goes another iteration to locate another feature, which, when paired with the first feature, would further increase the classification accuracy of the model [43]. This process keeps repeating for certain number of iterations to identify the combination of features which yields the highest accuracy. The algorithm also returns features with zero importance denoting that those features provide no contribution to enhancing the classification power of the model and hence should be discarded. Like forward selection, the backward elimination wrapper method also works to select a group of important features. However, instead of selecting one feature at a time and adding features in subsequent iterations, the backward elimination begins by selecting all the features together and then removing features one at a time with each iteration that have little or no effect on increasing the effectiveness of the model [43]. Forward selection does suffer from a drawback. Since the features are added incrementally to the model, there are scenarios where a combination of features may decrease accuracy and the best possible group may not be discovered since the combination did not include features selected by forward selection in the early stages of iteration. A similar problem may arise with backward elimination method. Using forward and backward selection methodology together allows us to verify if the features selected by both these methods are consistent and reduce this drawback to some extent. We implemented both these techniques on the two datasets to select another set of best possible features that can be used for classification.

4. Dataset Preprocessing

One of the two datasets used in this research is called the NSL-KDD dataset [44] and is the successor of the KDD'99 [45] dataset. The NSL-KDD dataset was developed to address shortcomings that were present in its predecessor [46]. The KDD'99 dataset developed on DARPA'98 Intrusion Detection System (IDS) evaluation program contained a significant amount of synthetic data. Study conducted on this dataset revealed that almost 78% of records were duplicated on train dataset and 75% of records were duplicated on test dataset. This redundancy introduced unnecessary bias to records that were more widely available in the dataset compared to records that were not present in a significant amount to provide sufficient information for the machine learning model to train on. NSL-KDD solved this issue by not including duplicate records in both test and train datasets. NSL-KDD consists of separate test and training datasets. The training dataset has 125,973 records and the test dataset has 22,544 records, each having 42 attributes that could be used for prediction. There are three categorical attributes *protocol_type*, *service*, and *flag*. Variables in these categorical attributes were one-hot encoded before using them as input for training.

One-hot encoding method maps categorical values into integer format [47]. An example of one-hot encoding method can be derived from [48] where the categorical variables are {apple, orange, berry}. The one-hot encoding is carried out by using unit vector for each category where apple is [1, 0, 0], orange is [0, 1, 0] and berry is [0, 0, 1]. This encoding can be written as a 3-dimensional feature vector {[1, 0, 0], [0, 1, 0], [0, 0, 1]}. The formatted data from one-hot encoding can now be fed into a machine learning algorithm for training. Examples of such applications are DNA sequencing [49] where unit value in the vector can represent the position of a DNA component called nucleotide [50], text representation as numerical matrix [50] where a text can be represented as a one-hot-encoded matrix. Such matrix can be fed into a convolutional neural network for the text classification. This encoding method is mostly used as a data pre-processing step. In this paper, the features are mutually exclusive making it suitable to apply one-hot encoding [51]. Also, the data used in this paper is free from dirty categories, which is a weakness of one-hot encoding [48].

Following the similar concept, one-hot encoding was applied to the categorical variables in the NSL-KDD dataset. For example, the feature *protocol_type* contains TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and ICMP (Internet Control Message Protocol). which, when one-hot-encoded, can be represented as [1, 0, 0], [0, 1, 0] and [0, 0, 1] respectively. Features *protocol_type*, *service*, and *flag* each comprised of 3, 70 and 11 variables respectively in the training dataset which were one-hot-encoded prior to training. Previous work also used one-hot-encoding on this dataset. where *protocol type*, *flag* and *service* were transformed into numerical values [52]. One-hot-encoding assumes that both training and test datasets contain equal number of variables in a feature column. However, this was not true for the feature *service* since the number of variables in test and training data were different. In the test dataset, there were 64 variables while the training dataset had 70. To address this issue, artificial records were created in the test dataset with the missing variables before applying one-hot-encoding. The application of one-hot-encoding on categorical variables mentioned above along with the existing continuous attributes now yielded 123 training features from the previously existing 42 features.

The UNSW-NB15 [53] dataset was also used to verify the usefulness of this feature engineering methodology. The dataset was generated in the Cyber Range Lab of the Australian Centre for Cyber Security using the IXIA PerfectStorm tool. The 100 Gb of raw traffic data captured by the tcpdump tool had 49 attributes used to identify 9 different types of attacks. The attributes were collected by the lab using Argus, Bro-IDS and a collection of 12 models developed specifically for extracting these features. In this research the training dataset that consisted of 82,332 records and testing dataset having 175,341 records were merged. This was followed by an 80–20 split where 80% of the records selected at random were used for training and 20% used for testing. This additional step was carried out to give the proposed feature engineering model more information for training. Implementing one-hot-encoding on the categorical variables generated 196 trainable features. Finally, the prediction class label was converted to a binary class label where 0 represented a normal packet and 1 represented that the packet was malicious.

5. Experimental Analysis

In this section, we test the efficacy of the proposed feature engineering model on the two datasets. The overall diagram of the entire process is summarized in Figure 1. In the first stage, features are selected independently by four algorithms used for dimensionality reduction. In stage 2, a meta-learning approach uses a group of selected features from the first stage to train five separate machine learning algorithms. These algorithms include a hybrid of Convolutional Neural Network (CNN) and Long Short term Memory (LSTM) proposed in [54], Bidirectional Long Short Term Memory (BiLSTM), Gated recurrent units (GRU), Decision Tree and Random Forest. Prediction accuracies were compared before and after application of the feature engineering steps on the datasets to validate the importance of this approach.

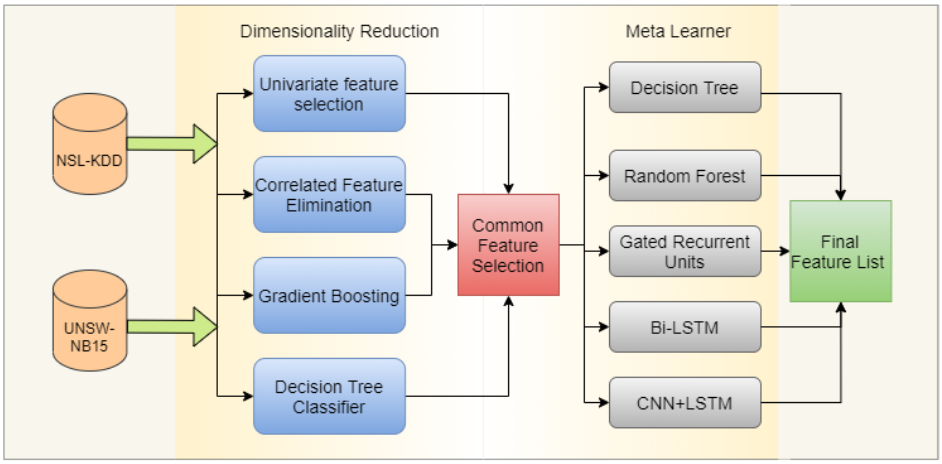


Figure 1. Illustration of the entire approach.

5.1. Univariate Feature Selection

Univariate feature selection was applied by using one way ANOVA F-test with the second percentile method on both NSL-KDD and UNSW NB-15 datasets. Out of the 123 input features generated by one-hot encoding of categorical attributes and adding continuous attributes of the NSL-KDD dataset, ANOVA F-test suggested only 13 features. From the 196 continuous and one-hot encoded categorical features in the UNSWNB-15 dataset, the ANOVA F-test suggested a list of only 20 features as being very important. These features are listed in Table 1.

Table 1. Important features from ANOVA test.

UNSW NB-15 Features		NSL KDD Features	
rate	sttl	count	logged_in
dload	swin	srv_serror_rate	serror_rate
stcpb	dtcpb	dst_host_srv_count	same_srv_rate
dwin	dmean	dst_host_serror_rate	dst_host_same_srv_rate
ct_srv_src	ct_state_ttl	service_http	dst_host_srv_serror_rate
ct_src_dport_ltm	ct_dst_sport_ltm	flag_S0	service_private
ct_dst_src_ltm	ct_src_ltm	flag_SF	
ct_srv_dst	proto_tcp		
service_dns	state_CON		
state_FIN	state_INT		

5.2. Correlated Feature Elimination

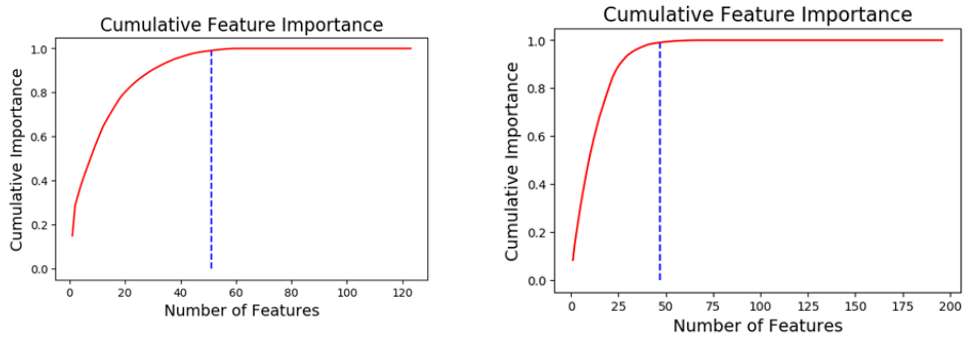
Based on the linear complexity, Pearson correlation was chosen for feature elimination process. We performed Karl Pearson Correlation test on both NSL-KDD and UNSW-NB15 datasets. The correlation threshold limit was set to 80 percentile. Features which have correlation value more than the set threshold were listed as highly correlated features and shown in Table 2. Several features appeared multiple times which enabled us to remove highly correlated features and reduce the model complexity.

Table 2. Pearson Correlation values.

UNSW NB-15 Dataset			NSL-KDD Dataset		
Feature	**Corr. Feature**	**Corr.**	**Feature**	**Corr. Feature**	**Corr.**
sbytes	spkts	0.964	num_root	num_compromised	0.998
dbytes	dpkts	0.973	is_guest_login	hot	0.860
sloss	spkts	0.972	srv_serror_rate	serror_rate	0.993
sloss	sbytes	0.996	srv_rerror_rate	rerror_rate	0.989
dloss	dpkts	0.979	dst_host_same_srv_rate	dst_host_srv_count	0.897
dloss	dbytes	0.997	dst_host_serror_rate	serror_rate	0.979
dwin	swin	0.981	dst_host_serror_rate	srv_serror_rate	0.978
synack	tcprtt	0.946	dst_host_srv_serror_rate	serror_rate	0.981
ackdat	tcprtt	0.919	dst_host_srv_serror_rate	srv_serror_rate	0.986
ct_dst_ltm	ct_srv_src	0.841	dst_host_srv_serror_rate	dst_host_serror_rate	0.985
ct_src_dport_ltm	ct_srv_src	0.862	dst_host_rerror_rate	rerror_rate	0.927
ct_src_dport_ltm	ct_dst_ltm	0.961	dst_host_rerror_rate	srv_rerror_rate	0.918
ct_dst_sport_ltm	ct_srv_src	0.815	dst_host_srv_rerror_rate	rerror_rate	0.964
ct_dst_sport_ltm	ct_dst_ltm	0.871	dst_host_srv_rerror_rate	srv_rerror_rate	0.970
ct_dst_sport_ltm	ct_src_dport_ltm	0.908	dst_host_srv_rerror_rate	dst_host_rerror_rate	0.925
ct_dst_src_ltm	ct_srv_src	0.954	service_ftp	is_guest_login	0.820
ct_dst_src_ltm	ct_dst_ltm	0.857	flag_REJ	rerror_rate	0.835
ct_dst_src_ltm	ct_src_dport_ltm	0.872	flag_REJ	srv_rerror_rate	0.841
ct_dst_src_ltm	ct_dst_sport_ltm	0.836	flag_REJ	dst_host_rerror_rate	0.813
ct_ftp_cmd	is_ftp_login	0.999	flag_REJ	dst_host_srv_rerror_rate	0.829
ct_src_ltm	ct_dst_ltm	0.901			

5.3. Gradient Boosting

A benefit of using gradient boosting tree is retrieving importance scores for each attributes in a relatively straight forward manner when the boosted tree is constructed. Importance for every single decision tree is calculated by the amount that each attribute split improves the performance of the model. For both datasets, the cut-off was set by the cumulative feature importance of the features as 99%. The cumulative feature importance is the weighted average of individual feature importance. It is the mean of all the importance values evaluated by the decision trees within the model. Out of the 123 features obtained from one-hot-encoding of the NSL-KDD dataset, the XGBoost classifier suggested 51 very important features. It also assigned 72 features as having zero importance. Out of the 196 features in the UNSW-NB15 dataset after one hot encoding, the XGBoost classifier returned 47 features as important and 149 features as having zero importance. The graphs shown in Figure 2a,b highlight these important features.



(**a**) (**b**)

Figure 2. Gradient Boosting importance values. (**a**) NSL-KDD dataset. (**b**) UNSW dataset.

5.4. Information Gain

Information gain mostly calculates the entropy reduction from transforming dataset. Besides being used in decision tree classifiers to define the best split ratio, it is also used to evaluate every feature in the training dataset in the context of a target variable. We calculated both information gain and gain ratio to check for any imbalance dependency between features. For both datasets, we limited our information gain output list size to same as the important feature list size as discussed above. Hence, the top 51 information gain feature list for NSL-KDD dataset and 47 feature list for the UNSW-NB15 dataset were recorded.

5.5. Wrapper Method Application

Wrapper method employs a greedy approach, which works by evaluating multiple subsets of features with a machine learning algorithm. This algorithm employs a search strategy to find the space of possible feature subsets and evaluates the performance by multiple iterations. In this experiment, random forest classifier was used to evaluate the performance for the feature subsets. We performed both forward sequential feature selection process and backward sequential feature elimination process to select the most important features. The output size was limited to 47 features for the UNSW-NB15 and 50 features for the NSL-KDD dataset based on Area Under the Curve (AUC) score. The features returned by the wrapper method are shown in Table 3.

5.6. Dynamic Feature Selection Using a Meta-Learning Approach

Algorithm 1 explains how features were selected for meta-learning. We begin by storing the output from each of the feature selection processes in six initial lists labelled $L1$ to $L6$. For lists $L3$ and $L5$ we begin by selecting all features that are common among Information Gain and Information Gain Ratio. In the next phase, we begin by removing all features from $L1$, $L2$, $L3$ and $L6$ that had no importance as reported by list $L4$ from XGBoost. This step allows us to eliminate any features that do not contribute towards enhancing the performance of the classification model prior to feeding them to the meta-learner.

In the final stage, the results from these four lists are evaluated to find the common features. These common features will serve as input for our meta-learner. While common features are being selected, we have also analyzed this common feature list L and removed features that appear to have a high correlation with other features and appear multiple times as reported by the output of correlated feature elimination processed and stored in $D[\,]$. Finally the list $L[\,]$ is sorted based on the XGBoost importance values to ensure that the most relevant features are selected earlier in the training process.

Algorithm 1 Prepare Datasets for Meta-Learners.

```
# Step 1:  Initialize the arrays by running the following methods.
L1[ ] ← ANOVA F-test
L2[ ] ← XG-Boost Most Important Features
L2[feature : score] ← Dictionary with XG-Boost Importance values and scores
L3[ ] ← Information Gain
L4[ ] ← XG-Boost Zero Importance
L5[ ] ← Information Gain Ratio
L6[ ] ← Wrapper Method Output
# Step 2:  Remove Zero Importance Features from L1, L2, L3, L6
if L3 == L5 then
    Return L3
else
    L3 ← L3 ∪ L5
end
ZeroF[] ← [L1,L2,L3,L6]
j ← 0
while j < len(ZeroF) do
    TempList ← ZeroF[j]
    for (i = 0, i < len(L4), i++) do
        k ← 0
        while k < len(TempList) do
            if (L4[i] == TempList[k] then
                Remove TempList[k];
            end
            k++;
        end
    end
    j++
end
# Step 3:  Combine output and remove correlated features
D[ ] ← Correlated Features
L ← L1 ⋂ L2 ⋂ L3 ⋂ L6
for (i = 0, i < len(D), i++) do
    k ← 0
    while k < len(L) do
        if (L[k] == D[i] then
            Remove L[k];
        end
        k++;
    end
end
Sort List L[ ] by XGBoost importance values from L2[feature : score]
```

Table 3. Important features from Wrapper Method.

NSL KDD Features		UNSW NB-15 Features	
duration	src_bytes	dpkts	dur
wrong_fragment	num_failed_logins	rate	sbytes
logged_in	root_shell	dload	dttl
count	dst_host_count	swin	sloss
dst_host_srv_count	dst_host_same_srv_rate	ackdat	tcprtt
dst_host_diff_srv_rate	dst_host_same_src_port_rate	response_body_len	trans_depth
dst_host_srv_diff_host_rate	dst_host_serror_rate	ct_state_ttl	ct_srv_src
dst_host_rerror_rate	diffic	ct_src_dport_ltm	ct_dst_ltm
Protocol_type_icmp	Protocol_type_tcp	ct_ftp_cmd	ct_dst_sport_ltm
service_IRC	service_echo	ct_src_ltm	ct_flw_http_mthd
service_eco_i	service_efs	proto_any	proto_3pc
service_exec	service_ftp	proto_ax.25	proto_aris
service_ftp_data	service_gopher	proto_ddp	proto_br-sat-mon
service_http	service_http_2784	proto_i-nlsp	proto_fc
service_http_8001	service_ldap	proto_idrp	proto_idpr
service_login	service_netbios_ns	proto_igp	proto_ifmp
service_nnsp	service_ntp_u	proto_ipv6-frag	proto_ipcv
service_pm_dump	service_red_i	proto_netblt	proto_iso-tp4
service_remote_job	service_supdup	proto_pvp	proto_pgm
service_telnet	service_tftp_u	proto_sdrp	proto_scps
service_uucp	service_uucp_path	proto_tlsp	proto_secure-vmtp
service_vmnet	service_whois	proto_trunk-2	proto_tp++
flag_REJ	flag_RSTO	proto_vines	proto_ttp
flag_RSTR	flag_S1	proto_wb-mon	
flag_S3	flag_SF		

5.7. Meta-Learner

Meta-learning, which is inspired by cognitive psychology is a proven effective method for machine learning techniques to excel at mastering predictive skills [55,56]. When a meta-learner is applied to the machine learning algorithm, it uses prior experience to change certain aspects of the algorithm. In simple terms, a meta-learner can be defined as a way that enables learning algorithms to learn themselves [57]. Machine learning algorithms are getting better over time, but still lack the versatility. This can be achieved by intelligent amalgamation of meta-learning along with similar techniques such as reinforcement learning, transfer learning and active learning [58,59]. Researchers also focus on using meta-learning for hyper-parameter tuning, neural network optimization and specifying best network architecture. In this research, we have used the meta-learner to optimize hyper-parameters through a bagging ensemble process [60]. The algorithm attempts to find the best features after carefully investigating the performance and robustness of the predictive model. This dynamic feature selector takes the output of ANOVA F-test, Pearson Correlation, Gradient Boosting, Information Gain and Wrapper Method to find out the best feature subset. What makes it dynamic is the nature of the meta-learner in this approach. The meta-learner processes feature that have been deemed as important by various feature selection processes. It then selects a subset of these best features and trains several machine learning models. The bagging ensemble process uses five different algorithms as mentioned

earlier, namely, a hybrid of CNN+LSTM proposed in [54], BiLSTM, GRU, Decision Tree and Random Forest. Based on how accuracy increases with every instance, the meta-learner dynamically adds and adapts its training to include new features to the already existing features used for prediction. Hence the meta-learner dynamically determines on per-instance training results as to what features are important for generating a high prediction accuracy.

5.7.1. CNN-LSTM

The hybrid of CNN and LSTM was developed in two steps namely, feature extraction based on CNN followed by a feature fusion part based on LSTM. This specific hybrid approach proved effective on NSL-KDD dataset without any feature reduction [54]. CNN is used since it has a proven track record for accurate classification in several domains such as image classification and prediction on multivariate datasets. Using LSTM modules enabled us to incorporate Recurrent Neural Network (RNN). An RNN is able to perform sequence modelling. Unlike a CNN, an RNN can help to create interaction between input sequences. To predict the next element of a series or a batch, RNN uses the previous features or elements thereby creating a loop which helps to keep track of this information [61–65]. This makes RNN extremely suitable for finding patterns in time series datasets [66], especially in the case of network intrusions since the datasets have multiple features like *sessions*.

The first part of this approach is feature extraction based on CNN. This is followed by a feature fusion part based on LSTM as shown in Figure 3. In the first stage, the forward propagation process is applied as it assumes the l layer is a convolution layer and the $l-1$ layer is a pooling layer or another input layer for the next extraction process. The entire process can be represented using Equation (3). The f here denotes the activation function, b represents the offset, k represents the convolution kernel, M_j represents the input feature map from the previous layer and variable x_{ji} represents the jth feature map of the lth layer. The summation represents the total result obtained from a convolution operation of all the feature maps in $l-1$ layer and the jth convolutional kernel of layer l.

$$x_{ji} = f(\sum i \epsilon M_j x_{il-1} \times k_{ijl} + b_{jl}). \tag{3}$$

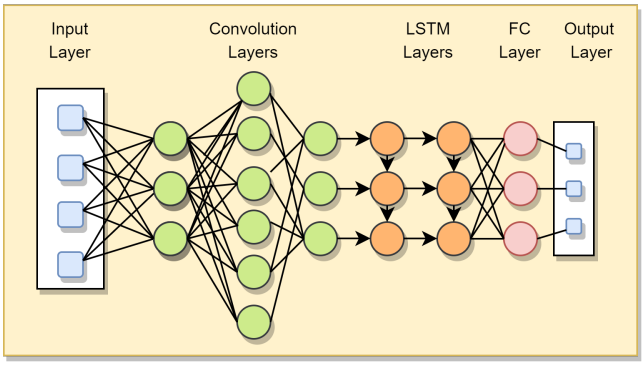


Figure 3. Illustration of the Hybrid Convolutional Neural Network (CNN) and Long Short term Memory (LSTM) architecture.

In a CNN, information may get lost as the network gets deeper due to the vanishing gradient problem. This is referred to as having a short term memory. LSTM modules is comprised of gates and a cell state. The cell state can be designated as a highway of information where features from a certain layer can be directly passed down several layers without any modification. The gates behave like neural networks and are responsible for knowledge discovery during training. These two features overcome the drawbacks of short term memory found in a CNN.

In the feature extraction stages, the sequential CNN model was comprised to 4 convolution layers with a max-pooling layer of size 2 placed after every 2 convolution layers. Relu was used as

an activation function for both convolutions. For the first convolution and pooling layer, there were 48 convolutional kernels with with a size of 3. The feature fusion consisting of LSTM was used where the output size of the LSTM part as 100. Finally, the classification results of the attack types were obtained through the Softmax function for better stochastic gradient descent. Number of epochs was set to 10. To prevent over-fitting issue, we used dropout as 0.1.

5.7.2. GRU

A gated recurrent unit (GRU) is a newer version of RNN and shares similar features with an LSTM. In a GRU, the cell state has been replaced by a hidden state to transmit information. Two new gates called reset and update are present in a GRU. The reset gate decides how much information needs to be forgotten from the previous layers, while the update gate decides what new information is required to be included in the LSTM. GRUs also offer faster training due to less operations. The GRU model was built using 2 GRU layers added sequentially. Each of these layers had 100 units. The final layer used a Softmax activation like before and trained for 10 epochs.

5.7.3. Bi-LSTM

Bi-LSTM or Bi-directional Long Short Term Memory is another variant of RNN which was tested in this meta-learning approach. In a traditional LSTM or RNN, the information propagates sequentially and in forward direction only, enabling future predictions based on past events. Bi-LSTM overcomes this unidirectional problem by training the model in both forward and reverse directions. BiLSTM usually brings together two independent RNNs which enable running input in two directions as future and past [67,68]. Our goal to using Bi-LSTM was to extract contextual information from every time interval in features like *sessions* to observe if the accuracy could be increased. The meta-learner included 2 Bi-LSTM layers, each having 50 LSTM units for training with Softmax activation. The Bi-LSTM was trained for 10 epochs.

5.7.4. Decision Trees and Random Forest

Tree based algorithms sometime outperform neural networks because they approach problems in a similar way by deconstructing them piece-by-piece, instead of finding one complex decision boundary that can separate the entire dataset like Support Vector Machine (SVM) or Logistic Regression. Tree-based methods progressively split the feature space along various features to optimize the total gain where as neurons of neural networks computes the probability of specific section of feature space with various overlapping. To compare the performance of neural networks we also introduced two tree-based algorithms; Decision Tree and Random Forest to get deterministic view.

Random Forest constructs multiple decision trees and then uses a voting mechanism to identify a record in a particular category based on the majority of votes that it received. Random Forest has been proven to generate a higher prediction accuracy since it uses results from multiple decision trees to make a prediction. The only drawback of Random Forest has to do with memory allocation for multiple decision trees. GINI index for used to identify feature importance for both decision trees and random forest.

5.7.5. Algorithm

Algorithm 2 highlights the entire process. We begin by using L which consists of the list of selected features from Algorithm 1. The meta-learner uses a subset of these features to begin training. We selected this subset to have the same feature size as the ANOVA F-Test list $L1$. Hence, 13 features from NSL-KDD and 20 features from UNSW datasets were selected to begin with. The meta-learner trains on this subset L' and records the accuracy in a list P. If the accuracy of previous iteration is less than the accuracy of current iteration, we add two subsequent features from L to the feature subset L'. However, if it is observed that the accuracy starts decreasing over two subsequent iterations, we stop the meta-learner and record the highest accuracy. The meta-learner also stops if it has added all the features from L to L' and records the final accuracy. Tables 4 and 5 shows the list of features that were

considered most important in the dynamic feature selection process by the meta-learner after feature selection algorithms were used. These features were selected by the meta-learner from the 123 one-hot encoded input features in the NSL-KDD dataset and the 196 trainable features on UNSW-NB15 dataset. In Figure 4a,c we demonstrate the accuracy of the NSL-KDD and UNSW datasets respectively which are generated from the training datasets using the selected features in Tables 4 and 5 by sequential CNN-LSTM, GRU and Bi-LSTM models. In Figure 4b,d we demonstrate the training loss of the NSL-KDD and UNSW datasets respectively using those selected features.

Algorithm 2 Meta-Learners.

Data: $L[\,]$; // List of selected features L from Algorithm 1
Data: $len(L1[\,])$; // Length of ANOVA feature list from Algorithm 1
Result: L'; // Final Feature List
Result: $P[\,]$; // Accuracy List derived from Meta Learner
# Begin Meta-Learner Approach
Initialize $L'[\,]$;
for $(i = 0, i < len(L1), i{+}{+})$ **do**
  L'[i] $\leftarrow$ L[i] ; // Copy initial features to L'
end
$A \leftarrow \text{len}(L1)$;
$j \leftarrow 0$;
while *True* **do**
  P[i] $\leftarrow$ Meta-Learner (CNN-LSTM, GRU, Bi-LSTM, DT, RF) on $L'[\,]$;
  if $A <$ *len(L)* **then**
    break; // Stop Meta Learner since no more features to add
  end
  if $i > 2$ **then**
    if *(P[i] < P[i-1] and P[i] < P[i-2]* **then**
      break; // Stop Meta Learner since accuracy is decreasing
    else
      L' $\leftarrow$ L' + L[A] + L[A+1] ; // Add two more features from L
      $A \leftarrow A + 2$;
    end
  end
  $j++$;
end
print(P[j]); // Print highest accuracy
print(*MetaLearnerStats*);

Table 4. Importance of final selected featues of NSL-KDD.

Features	XGBoost Imp.	Inf. Gain	Features	XGBoost Imp.	Inf. Gain
dst_host_srv_count	383.4	0.4179	count	275.3	0.4164
diffic	1223.5	0.2612	dst_host_count	309.4	0.2089
dst_host_diff_srv_rate	306.4	0.4518	dst_host_rerror_rate	269	0.0979
dst_host_same_src_port_rate	206.1	0.2358	dst_host_same_srv_rate	336.4	0.4004
dst_host_serror_rate	273.2	0.3986	dst_host_srv_diff_host_rate	309	0.2614
duration	196.3	0.0647	flag_SF	58.4	0.4892
logged_in	123.9	0.3063	root_shell	176.5	0.0018
service_http	98.4	0.0011	src_bytes	1321.7	0.7202

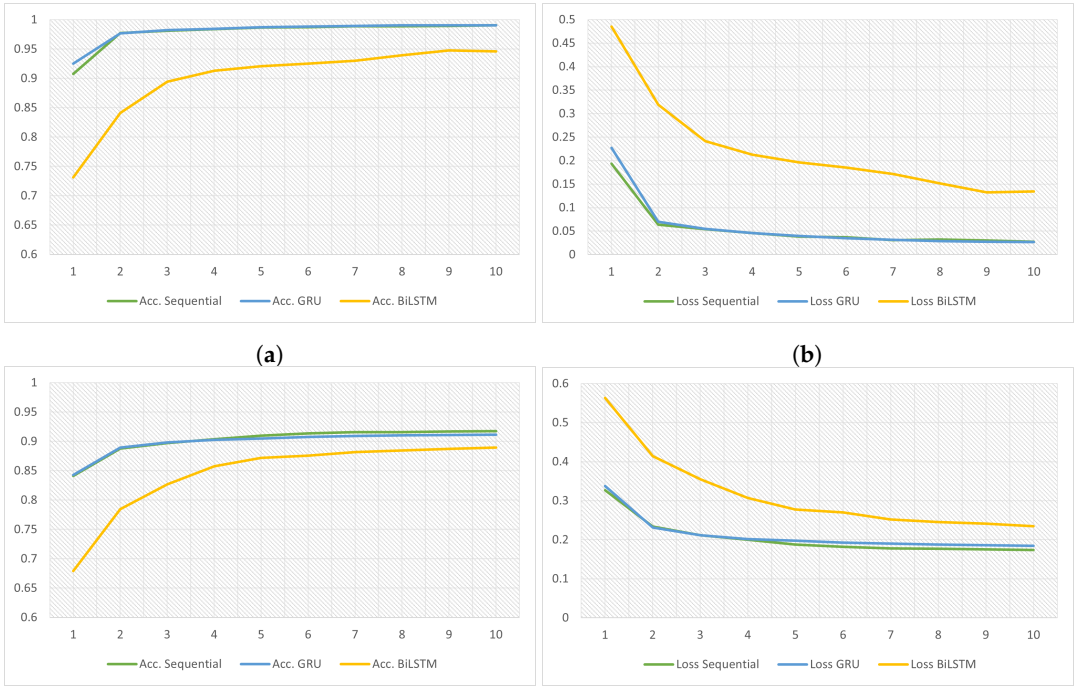


Figure 4. Accuracy and Loss values generated from final feature list selected by meta-learner. (**a**) NSL-KDD Accuracy. (**b**) NSL-KDD Loss. (**c**) UNSW Accuracy. (**d**) UNSW Loss.

Table 5. Importance of final selected featues of UNSW.

Features	XGBoost Imp.	Inf. Gain	Features	XGBoost Imp.	Inf. Gain
sbytes	2423.9	0.4722	dtcpb	1575.6	0.3811
ct_srv_src	1572.6	0.0827	stcpb	1519.7	0.3813
ackdat	1311.9	0.3401	ct_srv_dst	1215.6	0.0974
tcprtt	1098.5	0.3599	dur	1033.7	0.5392
ct_src_ltm	911	0.0745	dload	891.1	0.4938
ct_dst_ltm	712.2	0.0833	rate	664.1	0.5397
dmean	663.9	0.2837	response_body_len	506.2	0.0359
ct_src_dport_ltm	462.9	0.0941	ct_dst_sport_ltm	379.1	0.1502
sloss	321.8	0.1098	ct_state_ttl	281.1	0.3137
dwin	231.4	0.0582	dpkts	177.1	0.2427
ct_flw_http_mthd	152.8	0.0009	is_ftp_login	115.6	0.0001
service_dns	102.3	0.0368	trans_depth	92.3	0.0001
service_http	91.9	0.00006	proto_tcp	79.5	0.0677
state_CON	49	0.0616	state_FIN	25.4	0.0479
state_INT	22.2	0.1473			

6. Discussion

Table 6 shows the prediction results before and after the application of dynamic feature selection on NSL-KDD datasets. The accuracy, precision, F1-score and the Area Under Curve (AUC) in every experiment resulted in consistent performance improvement using the Dynamic Feature Selection algorithm. We did observe the random forest algorithm provide the best performance in the

meta-learner. Accuracy while using random forest jumped from 99.54% to 99.64% while reducing the feature size from 123 to 50.

Table 7 represents the output of Dynamic Feature Selector algorithm of UNSW-NB15 dataset. The UNSW-NB15 dataset also showed consistent performance increase using the best algorithm as random forest from 90.98% to 92.46% while reducing the feature size from 196 to 47. GRU produced a better precision, but the meta-learner selected random forest algorithm to produce output based on the overall performance metrics such as F1 score, accuracy and AUC. Accuracy is the most intuitive performance measure and a simple way to observe prediction, but it is often misleading due to specificity and sensitivity [69]. Also, since the class distribution is uneven, F1 score is a better indicator to select robustness of the model. For both datasets, dynamic feature selector shows an increment of F1 score and accuracy which clearly states that, this method reduces the feature size effectively with increment in performance as well.

We also applied Naive Bayes algorithm on the NSL-KDD and UNSW-NB15 datasets directly without using feature selection and a meta-learner to validate if our proposed approach enhances prediction accuracy. An accuracy of 53.32% was observed on NSL-KDD and 73.58% on the UNSW-NB15 test datasets after fitting the model on training datasets. These results further reveal the significance of our proposed approach.

Table 6. NSL-KDD Performance Metrics.

NSL-KDD	Performance without Dynamic Feature Selector					Performance with Dynamic Feature Selector				
	CNN+ LSTM	GRU	Bi-LSTM	Decision Trees	Random Forest	CNN+ LSTM	GRU	Bi-LSTM	Decision Trees	Random Forest
Accuracy	98.96	97.04	95.47	97.94	99.37	99.1	99.04	95.15	99.63	99.65
F1	98.21	98.43	93.50	98.01	99.33	99.15	99	95.05	99.62	99.63
Precision	96.92	97.74	95.70	98.79	99.63	99.35	98.74	96.6	99.78	99.84
AUC	98.93	98.72	96.45	97.13	99.54	99.17	99.05	95.15	99.63	99.64

Table 7. UNSW-NB15 Performance Metrics.

UNSW-NB15	Performance without Dynamic Feature Selector					Performance with Dynamic Feature Selector				
	CNN+ LSTM	GRU	Bi-LSTM	Decision Trees	Random Forest	CNN+ LSTM	GRU	Bi-LSTM	Decision Trees	Random Forest
Accuracy	91	90.12	88.93	85.31	90.98	91.91	90.75	89.28	86.8	92.76
F1	92.07	91.24	90.32	88.96	93.35	93.7	93.08	91.53	89.5	94.44
Precision	91.4	97.84	91.89	87.99	95.96	94.23	97.35	90.61	88.01	96.11
AUC	91	90.87	88.43	84.91	91.74	91.38	91.78	88.2	85.49	92.73

7. Conclusions

Network security has become an essential issue in any distributed system. Although a lot of machine learning algorithms have been experimented with to increase the efficacy of intrusion detection, it is still a major challenge for existing intrusion detection algorithms to achieve good performance. In this research we have dealt with two high dimensional network traffic datasets and proposed a novel Dynamic Feature Selector (DFS) algorithm which is based on feature selection and meta-learning technique. Our approach combined five prominent algorithms to process two well established datasets NSL-KDD and UNSW-NB15. Primarily, we conducted multiple feature engineering steps using statistical analysis like univariate test and Pearson coefficient test along with XGBoost importance, wrapper technique and information gain ratio to reduce feature dimensionality. The outputs generated from these algorithms were documented and used as input for a novel bagging ensemble technique known as the meta-learner. This meta-learner is used as an optimizer for feature

selection and ranking from multiple feature subsets. The output of different feature selection process and algorithms are filtered in a stochastic process with the help of the meta-learner which consisted of five state of art classification techniques to increase the performance of the model. With the usage of DFS, we were able to reduce the number of features by more than 50% and increase the predictive performance significantly. DFS also suggested the best algorithms out of the five machine learning algorithms which is reproducible and deployment ready.

Future research will include the extension of the DFS algorithm for reinforcement learning to prevent the intrusion in network. Self-learning techniques will enhance the performance of DFS and extend its application to datasets from other domains to produce a robust machine learning model. DFS will also be implemented on live network traffic to document its performance and adjust hyper-parameters of meta-learner to enhance cybersecurity.

Author Contributions: Conceptualization, M.A., K.E.N., R.G. and M.M.C.; methodology, M.A. and K.E.N.; software, M.A.; validation, M.A., K.E.N. and R.G. data curation, M.A. and K.E.N.; writing—original draft preparation, M.A., R.G. and M.M.C.; writing—review and editing, R.G. and M.M.C.; visualization, R.G., M.A. and M.M.C.; supervision, K.E.N.; funding acquisition, K.E.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Department of Computer Science at North Dakota State University.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RNN	Recurrent Neural Network
CNN	Convolutional Nueral Network
LSTM	Long Short Term Memory
Bi-LSTM	Bidirectional Long Short Term Memory
GRU	Gated Recurrent Units

References

1. Chowdhury, M.; Nygard, K. Machine Learning within a Con Resistant Trust Model. In Proceedings of the The 33rd International Conference on Computers and their Applications (CATA 2018), Las Vegas, NV, USA, 19–21 March 2018.
2. Blum, A.L.; Langley, P. Selection of relevant features and examples in machine learning. *Artif. Intell.* **1997**, *97*, 245–271. [CrossRef]
3. Yang, Y.; Pedersen, J.O. A comparative study on feature selection in text categorization. In Proceedings of the ICML, Nashville, TN, USA, 8–12 July 1997; Volume 97, p. 35.
4. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
5. Hu, H.; Li, J.; Plank, A.; Wang, H.; Daggard, G. A comparative study of classification methods for microarray data analysis. In Proceedings of the 5th Australasian Data Mining Conference (AusDM 2006): Data Mining and Analytics 2006, Sydney, NSW, Australia, 29–30 November 2006; pp. 33–37.
6. Niranjan, A.; Prakash, A.; Veena, N.; Geetha, M.; Shenoy, P.D.; Venugopal, K. EBJRV: An Ensemble of Bagging, J48 and Random Committee by Voting for Efficient Classification of Intrusions. In Proceedings of the 2017 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), Dehradun, India, 18–19 December 2017; pp. 51–54.
7. Camargo, C.O.; Faria, E.R.; Zarpelão, B.B.; Miani, R.S. Qualitative evaluation of denial of service datasets. In Proceedings of the XIV Brazilian Symposium on Information Systems, Caxias do Sul, Brazil, 4–8 June 2018; pp. 1–8.
8. Bachl, M.; Hartl, A.; Fabini, J.; Zseby, T. Walling up Backdoors in Intrusion Detection Systems. In Proceedings of the 3rd ACM CoNEXT Workshop on Big Data, Machine Learning and Artificial Intelligence for Data Communication Networks, Orlando, FL, USA, 9 December 2019; pp. 8–13.

9. Liu, H.; Liu, Z.; Liu, Y.; Gao, X. Abnormal Network Traffic Detection based on Leaf Node Density Ratio. In Proceedings of the 2019—9th International Conference on Communication and Network Security, Chongqing, China, 15–17 November 2019; pp. 69–74.
10. Faker, O.; Dogdu, E. Intrusion detection using big data and deep learning techniques. In Proceedings of the 2019 ACM Southeast Conference, Kennesaw, GA, USA, 18–20 April 2019; pp. 86–93.
11. Thejas, G.; Jimenez, D.; Iyengar, S.S.; Miller, J.; Sunitha, N.; Badrinath, P. COMB: A Hybrid Method for Cross-validated Feature Selection. In Proceedings of the ACM Southeast Regional Conference, Tampa, FL, USA, 2–4 April 2020; pp. 100–106.
12. Ding, Y.; Zhai, Y. Intrusion detection system for NSL-KDD dataset using convolutional neural networks. In Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, Shenzhen, China, 8–10 December 2018; pp. 81–85.
13. Belouch, M.; Elhadaj, S.; Idhammad, M. A hybrid filter-wrapper feature selection method for DDoS detection in cloud computing. *Intell. Data Anal.* **2018**, *22*, 1209–1226. [CrossRef]
14. Khammassi, C.; Krichen, S. A GA-LR wrapper approach for feature selection in network intrusion detection. *Comput. Secur.* **2017**, *70*, 255–277. [CrossRef]
15. Tun, M.T.; Nyaung, D.E.; Phyu, M.P. Network Anomaly Detection using Threshold-based Sparse. In Proceedings of the 11th International Conference on Advances in Information Technology, Bangkok Thailand, 1–3 July 2020; pp. 1–8.
16. Viet, H.N.; Van, Q.N.; Trang, L.L.T.; Nathan, S. Using deep learning model for network scanning detection. In Proceedings of the 4th International Conference on Frontiers of Educational Technologies, Moscow, Russia, 25–27 June 2018; pp. 117–121.
17. Primartha, R.; Tama, B.A. Anomaly detection using random forest: A performance revisited. In Proceedings of the 2017 International Conference on Data and Software Engineering (ICoDSE), Palembang, Indonesia, 1–2 November 2017; pp. 1–6.
18. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
19. Belouch, M.; hadaj, S.E. Comparison of ensemble learning methods applied to network intrusion detection. In Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing, Cambridge, UK, 22–23 March 2017; pp. 1–4.
20. Liu, J.; Kantarci, B.; Adams, C. Machine learning-driven intrusion detection for Contiki-NG-based IoT networks exposed to NSL-KDD dataset. In Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning, Linz, Austria, 13 July 2020; pp. 25–30.
21. Tran, B.; Xue, B.; Zhang, M. Class dependent multiple feature construction using genetic programming for high-dimensional data. In *Australasian Joint Conference on Artificial Intelligence, Proceedings of the AI 2017: AI 2017: Advances in Artificial Intelligence, Melbourne, VIC, Australia, 19–20 August 2017*; Springer: Cham, Switzerland, 2017; pp. 182–194.
22. Krishna, G.J.; Ravi, V. Feature subset selection using adaptive differential evolution: An application to banking. In Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, Kolkata, India, 3–5 January 2019; pp. 157–163.
23. Wang, L.; Zhou, N.; Chu, F. A general wrapper approach to selection of class-dependent features. *IEEE Trans. Neural Netw.* **2008**, *19*, 1267–1278. [CrossRef]
24. Tran, B.; Zhang, M.; Xue, B. Multiple feature construction in classification on high-dimensional data using GP. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece, 6–9 December 2016; pp. 1–8.
25. Hariharakrishnan, J.; Mohanavalli, S.; Kumar, K.S. Survey of pre-processing techniques for mining big data. In Proceedings of the 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP), Chennai, India, 10–11 January 2017; pp. 1–5.
26. Enache, A.C.; Sgarciu, V.; Petrescu-Niţă, A. Intelligent feature selection method rooted in Binary Bat Algorithm for intrusion detection. In Proceedings of the 2015 IEEE 10th Jubilee International Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania, 21–23 May 2015; pp. 517–521.
27. Mohammadi, S.; Mirvaziri, H.; Ghazizadeh-Ahsaee, M.; Karimipour, H. Cyber intrusion detection by combined feature selection algorithm. *J. Inf. Secur. Appl.* **2019**, *44*, 80–88. [CrossRef]

28. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1.
29. Ahsan, M.; Gomes, R.; Denton, A. Application of a Convolutional Neural Network using transfer learning for tuberculosis detection. In Proceedings of the 2019 IEEE International Conference on Electro Information Technology (EIT), Brookings, SD, USA, 20–22 May 2019; pp. 427–433.
30. Liu, H.; Yu, L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 491–502.
31. Guyon, I.; Elisseeff, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
32. Kim, T.K. Understanding one-way ANOVA using conceptual figures. *Korean J. Anesthesiol.* **2017**, *70*, 22. [CrossRef]
33. Benesty, J.; Chen, J.; Huang, Y.; Cohen, I. Pearson correlation coefficient. In *Noise Reduction in Speech Processing*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–4.
34. Benesty, J.; Chen, J.; Huang, Y. On the importance of the Pearson correlation coefficient in noise reduction. *IEEE Trans. Audio Speech Lang. Process.* **2008**, *16*, 757–765. [CrossRef]
35. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
36. Chen, T.; He, T.; Benesty, M.; Khotilovich, V.; Tang, Y. Xgboost: Extreme gradient boosting. *R Package Version 0.4-2*. 2015; pp. 1–4. Available online: https://mran.microsoft.com/web/packages/xgboost/vignettes/xgboost.pdf (accessed on 15 July 2020).
37. Gomes, R.; Denton, A.; Franzen, D. Quantifying Efficiency of Sliding-Window Based Aggregation Technique by Using Predictive Modeling on Landform Attributes Derived from DEM and NDVI. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 196. [CrossRef]
38. Bennett, K.P. *Decision Tree Construction via Linear Programming*; Technical Report; University of Wisconsin-Madison Department of Computer Sciences: Madison, WI, USA, 1992.
39. Harris, E. Information Gain Versus Gain Ratio: A Study of Split Method Biases. In Proceedings of the ISAIM, Fort Lauderdale, FL, USA, 2–4 January 2002.
40. Hall, M.A.; Smith, L.A. Feature Selection for Machine Learning: Comparing a Correlation-Based Filter Approach to the Wrapper. In Proceedings of the FLAIRS Conference, Orlando, FL, USA, 1–5 May 1999; Volume 1999, pp. 235–239.
41. Gomes, R.; Ahsan, M.; Denton, A. Random forest classifier in SDN framework for user-based indoor localization. In Proceedings of the 2018 IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, USA, 3–5 May 2018; pp. 0537–0542.
42. Pal, M. Random forest classifier for remote sensing classification. *Int. J. Remote Sens.* **2005**, *26*, 217–222. [CrossRef]
43. Mao, K.Z. Orthogonal forward selection and backward elimination algorithms for feature subset selection. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2004**, *34*, 629–634. [CrossRef]
44. NSL-KDD Dataset. 2020. Available online: https://www.unb.ca/cic/datasets/nsl.html (accessed on 15 July 2020).
45. KDD Cup 1999 Data. 2020. Available online: http://kdd.ics.uci.edu/databases/kddcup99/ (accessed on 15 July 2020).
46. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
47. Seger, C. An Investigation of Categorical Variable Encoding Techniques in Machine Learning: Binary Versus One-Hot and Feature Hashing. 2018. Available online: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1259073&dswid=-2157/ (accessed on 10 November 2020).
48. Cerda, P.; Varoquaux, G.; Kégl, B. Similarity encoding for learning with dirty categorical variables. *Mach. Learn.* **2018**, *107*, 1477–1494. [CrossRef]
49. Choong, A.C.H.; Lee, N.K. Evaluation of convolutionary neural networks modeling of DNA sequences using ordinal versus one-hot encoding method. In Proceedings of the 2017 International Conference on Computer and Drone Applications (IConDA), Kuching, Malaysia, 9–11 November 2017; pp. 60–65.

50. Nguyen, N.G.; Tran, V.A.; Ngo, D.L.; Phan, D.; Lumbanraja, F.R.; Faisal, M.R.; Abapihi, B.; Kubo, M.; Satou, K. DNA sequence classification by convolutional neural network. *J. Biomed. Sci. Eng.* **2016**, *9*, 280. [CrossRef]
51. Cohen, J.; Cohen, P.; West, S.G.; Aiken, L.S. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*; Lawrence Erlbaum Associates Publishers: Mahwah, NJ, USA, 2013.
52. Su, T.; Sun, H.; Zhu, J.; Wang, S.; Li, Y. BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset. *IEEE Access* **2020**, *8*, 29575–29585. [CrossRef]
53. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 military communications and information systems conference (MilCIS), Canberra, ACT, Australia, 10–12 November 2015; pp. 1–6.
54. Ahsan, M.; Nygard, K.E. Convolutional Neural Networks with LSTM for Intrusion Detection. In Proceedings of the CATA, San Francisco, CA, USA, 23–25 March 2020; pp. 69–79.
55. Nichol, A.; Achiam, J.; Schulman, J. On first-order meta-learning algorithms. *arXiv* **2018**, arXiv:1803.02999.
56. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv* **2017**, arXiv:1703.03400.
57. Lemke, C.; Budka, M.; Gabrys, B. Metalearning: A survey of trends and technologies. *Artif. Intell. Rev.* **2015**, *44*, 117–130. [CrossRef] [PubMed]
58. Cruz, R.M.; Sabourin, R.; Cavalcanti, G.D.; Ren, T.I. META-DES: A dynamic ensemble selection framework using meta-learning. *Pattern Recognit.* **2015**, *48*, 1925–1935. [CrossRef]
59. Lin, S.C.; Yuan-chin, I.C.; Yang, W.N. Meta-learning for imbalanced data and classification ensemble in binary classification. *Neurocomputing* **2009**, *73*, 484–494. [CrossRef]
60. Dvornik, N.; Schmid, C.; Mairal, J. Diversity with cooperation: Ensemble methods for few-shot classification. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 3723–3731.
61. Fu, R.; Zhang, Z.; Li, L. Using LSTM and GRU neural network methods for traffic flow prediction. In Proceedings of the 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Wuhan, China, 11–13 November 2016; pp. 324–328.
62. Dey, R.; Salemt, F.M. Gate-variants of gated recurrent unit (GRU) neural networks. In Proceedings of the 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS), Boston, MA, USA, 6–9 August 2017; pp. 1597–1600.
63. Chang, L.Y.; Chen, W.C. Data mining of tree-based models to analyze freeway accident frequency. *J. Saf. Res.* **2005**, *36*, 365–375. [CrossRef] [PubMed]
64. Aldous, D. Tree-based models for random distribution of mass. *J. Stat. Phys.* **1993**, *73*, 625–641. [CrossRef]
65. Yang, Y.; Morillo, I.G.; Hospedales, T.M. Deep neural decision trees. *arXiv* **2018**, arXiv:1806.06988.
66. Zhang, J.; Man, K. Time series prediction using RNN in multi-dimension embedding phase space. In Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218), San Diego, CA, USA, 14 October 1998; Volume 2, pp. 1868–1873.
67. Zhang, L.; Xiang, F. Relation classification via BiLSTM-CNN. In *International Conference on Data Mining and Big Data, Proceedings of the DMBD 2018: Data Mining and Big Data, Shanghai, China, 17–22 June 2018*; Springer: Cham, Switzerland, 2018; pp. 373–382.
68. Sharfuddin, A.A.; Tihami, M.N.; Islam, M.S. A deep recurrent neural network with bilstm model for sentiment classification. In Proceedings of the 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), Sylhet, Bangladesh, 21–22 September 2018; pp. 1–4.
69. Powers, D.M. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv* **2011**, arXiv:2010.16061.