On the Performance of Machine Learning Models for Anomaly-Based Intelligent Intrusion Detection Systems for the Internet of Things

Ghada Abdelmoumin, Danda B. Rawat, *Senior Member, IEEE* and Abdul Rahman

***Abstract*—Anomaly-based machine learning-enabled intrusion detection systems (AML-IDS) show low performance and prediction accuracy while detecting intrusions in the Internet of Things (IoT) than that of deep learning-based intrusion detection systems (DL-IDS). In particular, AML-IDS that employ low complexity models for IoT, such as the Principal Component Machine (PCA) method and the One-class Support Vector Machine (1-SVM) method, are inefficient in detecting intrusions when compared to DL-IDS with the two-class Neural Network (2-NN) method. PCA and 1-SVM AML-IDS suffer from low detection rates compared to DL-IDS. The size of the dataset and the number of features or variants in the dataset may influence how well PCA and 1-SVM AML-IDS perform compared to DL-IDS. We attribute the low performance and prediction accuracy of the AML-IDS model to an imbalanced dataset, a low similarity index between the training data and testing data, and the use of a single-learner model. The intrinsic limitations of the single-learner model have a direct impact on the accuracy of an intelligent IDS. Also, the dissimilarity between testing data and training data leads to an increasingly high rate of false positives in AML-IDS than DL-IDS, which have low false alarms and high predictability. In this paper, we examine the use of optimization techniques to enhance the performance of single-learner AML-IDS, such as PCA and 1-SVM AML-IDS models for building efficient, scalable, and distributed intelligent IDS for detecting intrusions in IoT. We evaluate these AML-IDS models by tuning hyperparameters and ensemble learning optimization techniques using the Microsoft Azure ML Studio (AMLS) platform and two datasets containing malicious and benign IoT and industrial IoT (IIoT) network traffic. Furthermore, we present a comparative analysis of AML-IDS models for IoT regarding their performance and predictability.**

***Index Terms*—Intrusion detection, Internet of Things, machine learning, hyperparameter tuning, ensemble learning, Stacking.**

I. Introduction

Intelligent intrusion detection systems (IDSs) that employ machine-learning (ML) have shown promising results in detecting intrusions [1], [2], [3], [4], [5]. Anomaly-based machine learning-enabled intrusion detection systems (AML-IDS) that employ less complex models for IoT such as the Principal Component Machine (PCA) method and the One-class Support Vector Machine (1-SVM) method are inefficient in detecting intrusions when compared to deep learning IDS (DL-IDS) with the two-class Neural Network (2-NN) method. PCA and 1-SVM AML-IDS suffer from low detection rates compared to DL-IDS [6], [7]. PCA AML-IDS is better when dealing with relatively large data with a small number of features and 1-SVM AML IDS does not scale well when kernel-SVM is used [8]. The 1-SVM AML-IDS is more suitable when the data traffic is relatively small. Furthermore, PCA and 1-SVM AML-IDS are comparatively better than signature-based IDS for detecting unknown attacks. The 2-NN DL-IDS have shown good performance with low false alarms for large and small datasets with a large or small number of features.

ML based intelligent IDS models suffer from high false-positive rates and low detection rates. We attribute the poor performance of these models in detecting intrusions to using a single-learner model to train the AML-IDS model, among others. Hence, the need to explore ways to optimize the performance of these models is acutely critical for building efficient and high-performing AML-IDS models with low false-positives rates, good predictability, and high detection rate for IoT. According to [1], ensemble learning is used widely where the weakness of single-learner classifiers is compensatable by integrating the predictions made by each classifier in the ensemble and then combine these predictions to enhance the performance. In addition to using ensemble learning as a viable solution, we examine the use of hyperparameter tuning as a standalone solution and in conjunction with ensemble learning to optimize the low-performing AML-IDS models.

This paper focuses on optimizing the PCA and 1-SVM AML-IDS models using hyperparameter tuning and ensemble learning to enhance the security in IoT. First, we perform hyperparameter tuning to optimize these models by selecting the optimal model structure. Then, we apply ensemble learning using the Stacking technique to create multiple learners and train the single-learner AML-IDS models to boost the single-learner model detection performance. We train the optimized models using ground-truth network intrusion datasets containing both malicious and benign IoT and IIoT traffic and evaluate their performance using standard ML/DL metrics.

The remainder of this paper is organized as follows. Section II provides background and related work. Section III provides

Ghada Abdelmoumin and Danda B Rawat are with the Department of Electrical Engineering and Computer Science at Howard University, USA. Email: ghada.abdelmoumin@bison.howard.edu, danda.rawat@howard.edu

Abdul Rahman is with Microsoft Corp. USA. Email: abdulrahman@microsoft.com

This work was supported in part by the DoD Center of Excellence in AI and Machine Learning (CoE-AIML) at Howard University under Contract Number W911NF-20-2-0277 with the U.S. Army Research Laboratory, by the Microsoft Research and by the US National Science Foundation under the grant number 1828811 and CNS/SaTC 2039583. However, any opinion, finding, and conclusions or recommendations expressed in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the funding agencies.

an overview of optimization techniques. Section IV describes the experimental setup. Section V presents the performance evaluation. Section VII provides conclusion and future work.

II. Background and Related Work

Building reliable and accurate intrusion detection techniques requires efficient clustering/classification, less human interaction, low computational cost and overhead, identification of new attacks and their types, and robustness [9]. There are different approaches to overcome the limitations of traditional IDS ranging from mining and statistics to logic and intelligence to machine learning or deep learning based solutions [1], [4], [10], [9]. For boosting the performance of intelligent IDS, ensemble learning has been proposed as opposed to single learning, where multiple classifiers are combined by using the Stacking or Stacking Generalization technique to increase the prediction accuracy and reduce false alarms [11].

Focusing on network anomalies in critical cyber-physical systems network traffic, the authors in [12] proposed an ensemble method that leveraged deep models and the Stacking technique to obtain a reliable classification of outliers using a heterogeneous flow-based dataset that included IoT data. To find the best ensemble technique for detecting network intrusions, authors in [13] analyzed and evaluated the performance of the Bagging, AdaBoost, Stacking, Decorate, Voting, and Random Forest ensemble (with homogeneous and heterogeneous) techniques using several ML algorithms from the Weka Data Mining tool. In [13], the ensembles that use Stacking and Decorate techniques to aggregate the outcome of their base classifiers have higher training time and testing time than classifiers that used other ensemble techniques when compared to single classifiers. To apply ensemble learning while reducing the time complexity and utilizing resources efficiently, the authors in [14] proposed an intrusion detection method based on recursive feature elimination and the Stacking technique to reduce the training time and testing time in ensemble learning. The semi-stack approach in [15] used three layers; multi-view layer, semi-ensemble layer, and meta-learning layer to improve the network classification while reducing the bias and variance of the meta-classifier. The semi-stack approach showed significant improvements in its capability to classify network intrusions accurately compared to other methods; however, the other methods were faster when considering run-time. Focusing on the efficacy of intelligent IDS deployed in a fog-to-things environment, the study in [3] used a realistic intrusion detection dataset, ensemble learning, and two classification levels to detect and prevent anomalies while increasing detection accuracy, reducing latency, and minimizing resource utilization. The cluster-based ensemble classifier for IDS in [16] used clustering to reduce the number of the features in the dataset and the Boosting technique to obtain a precise prediction. The ANID-SEoKELM method in [17] proposed an adaptable real-time intelligent IDS based on the selective ensemble of kernel extreme learning machine with random features. The authors in [18] provided a systematic review of intrusion detection systems based on ensembles in machine learning and presented future research directions for developing effective IDSs.

However, none of the state-of-the-art approaches use the multi-expert method and Stacking techniques to build heterogeneous ensembles and aggregate the output of different intelligent IDS models (that are base learners) into an ensemble IDS model (ensemble learner) that we consider in this paper. The proposed approach focuses on optimizing anomaly detection IDS models where learning is statistical, 'normal' behavior is observable, and anomalous traffic is distinguishable using supervised, semi-supervised, and unsupervised approaches. Our goal is to optimize the performance of AML-IDS for IoT by employing two optimization methods; hyperparameter tuning and ensemble learning to improve the predictability and boost the performance of single-learner AML-IDS models for detecting intrusions in IoT environments.

III. Optimization Methods

A. Hyperparameter Tuning

Each ML/DL model has a set of hyperparameters that determine the complexity of its internal structure and learnability. For example, the learning rate, number of layers, number of hidden layers are examples of hyperparameters associated with a deep neural network (DNN) model [19], and SVM has hyperparameters related to balancing its prediction accuracy, defining the dimensionality of its feature space, and controlling the number of its data points [20], [21]. Finding the best configurations that yield the best performance is a critical optimization problem. Hence, hyperparameter tuning is achievable by solving an optimization problem [20]. Therefore, rather than choosing the best model via trial and error using different ML/DL methods, hyperparameter tuning can achieve the same results by tuning the model's hyperparameters until the best model is obtainable. According to [22], let Z be the n-dimensional hyperparameters phase space and $z_1, z_2, ..., z_n$ a set of n hyperparameters, such that z contained in Z, the goal of hyperparameter tuning is to find a set of hyperparameters ẑ (where ẑ= $argmin\ f(z)$) that gives an optimal model performance ŷ (ŷ = $f(\hat{z})$) measured using a validation set. Finding the optimal hyperparameters that yield a suitable model structure often depends on the dataset; hence, there are different optimal hyperparameter settings for different datasets, decision trees, or regression methods, each requiring a separate tuning [23], [24]. Furthermore, the best configuration setting depends not only on the dataset type but also on the dataset size [22]. In the AMLS platform, the integrated train and tune method helps to find the model's optimum settings. Finding the optimal model is achievable using the following steps:

1) Configure a set of parameters;
2) Iterate over multiple combinations of parameters;
3) For each set of combined parameters, measure the accuracy (or other metrics) of the combined parameters;
4) Continue to measure the combined parameters' accuracy (or other metrics) until an optimal model emerges.

Selecting hyperparameters is a tedious and computationally expensive process that may require an exhaustive search to obtain the best parameters settings. As a result, several selection methods, such as grid search, random search, and Bayesian-based optimization techniques, are available. Grid search uses a pre-defined grid to try different configurations or settings and identify the best model. The random search uses a defined range of values and randomly selects parameter values to obtain the optimum settings. While grid search is more suited to use when the best parameter settings are unknown, random search is more suitable to use when increasing performance based on user-defined metrics is the goal. Additionally, the random search conserves system resources compared to the grid search, where the system tries all combinations of values to obtain the optimum configurations. Bayesian-based optimization, a statistical model, treats the problem of tuning the model's hyperparameter as an optimization problem [22]. As a result, it has a high probability of finding the best parameter settings. Similar to random search, it is more accurate and computationally efficient compared to grid search. Both random search and Bayesian-based optimization are suitable for high-dimensionality datasets. This paper uses random search based on a given number of iterations to obtain the best set of hyperparameters that optimize the learning model F1-score, accuracy, and area under the curve (AUC).

B. Ensemble Learning

Ensemble learning is a machine learning approach that combines either similar or dissimilar models to improve model predictability. The aggregation of multiple models can boost the model performance by increasing its prediction accuracy and detection rate [18], [25]. Further, combining multiple models offset the weakness in individual models, thus producing a single model with stable output and improved prediction accuracy. In general, ensemble classifiers handle the model's bias, variance, or both more effectively than a single classifier [11], [13]. When different classifiers determine their individual biases, they collectively filter all biases. The multiple competing models work collaboratively to assess inaccurate regions in the feature space. The desired features possessed by each model invariably increase the model stability, thus addressing issues such as under-fitting, over-fitting, or over-trained models. Fig. 1 shows the general architecture of ensemble learning using the Stacking technique. The use

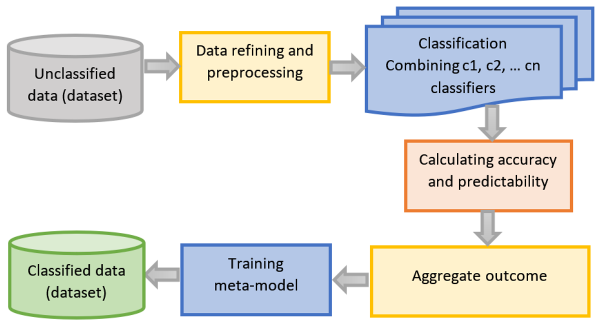

Unclassified data (dataset)
Data refining and preprocessing
Classification Combining c1, c2, … cn classifiers
Calculating accuracy and predictability
Aggregate outcome
Training meta-model
Classified data (dataset)

Fig. 1: General Architecture of Ensemble Techniques (adopted from [26])

of ensemble learning can address issues resulting from insufficient training data for modeling the hypothesis space or all potential solutions, models trapped in local optima, and the inability of the learning method to model the hypothesis space [18] accurately. Aggregating the outcome (output) of multiple models provides more accurate outcomes when data is insufficient, provides enough computational effort to find the global optima, and helps find the optimal model. However, the ensemble's efficiency depends on the base classifier, integration method, and errors reported by the multiple classifiers on the same training data subset. Building ensembles is a systematic process that involves three phases: ensemble generation, ensemble selection, and ensemble integration [18]. Consequently, creating the ensemble model, a.k.a. meta-model, involves the following four steps:

- Input dataset
- Generate base classifiers C_0, where $C_0 = \{C_1, C_2, C_3, ..., C_n\}$
- Select based classifiers C, where $C = \{C_1, C_3, ..., C_n\}$ and $C \subseteq C_0$
- Integrate base classifiers C_{Final}, where $C_{Final} = \{C_1, C_3, ..., C_n\}$

Several methods can be used to generate, select, and integrate an ensemble. The methods to generate base classifiers are either parameter-based, feature-based, or data-based. It is possible to select all the generated base classifiers, a subset, or a reduced set that exceeds a pre-set threshold, such as the performance threshold during the selection phase. Fig. 2 depicts the various phases of the ensemble building process and their associated methods. In general, building ensembles involves using two levels of classifiers; level-0 classifiers and level-1 classifier [11], [12], [13]. Level-0 represents the base models or classifiers, and level-1 represents the meta-model or meta classifier. In the steps listed above, C_{Final} is the meta-model whose output is the aggregate of level-0 classifiers or base models.

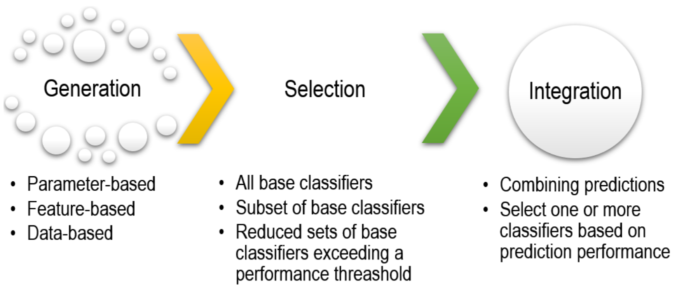

Generation
Selection
Integration
• Parameter-based
• Feature-based
• Data-based
• All base classifiers
• Subset of base classifiers
• Reduced sets of base classifiers exceeding a performance threshold
• Combining predictions
• Select one or more classifiers based on prediction performance

Fig. 2: The Ensemble Process Phases and their Methods

Ensemble learning is achievable using methods that exploit the different characteristics of the dataset for generating base classifiers to build an effective meta-model [18]. These methods include Bagging, Boosting, Stacking, Voting, AdaBoost, Output Code, Troika, Mixtures of Experts, Random Forest, and Decorate. The primary goal of these methods is to reduce or minimize the bias, variance, or their trade-off. Bias is an error due to the geometric model, whereas variance is an error due to variability of the model with respect to the dataset randomness [13]. Fig. 3 shows the trade-off or proportional relationship between the model's bias and variance. The Boosting and Stacking methods seek to minimize the bias of the model to improve prediction results, whereas the bagging method reduces the variance without increasing the bias [14].

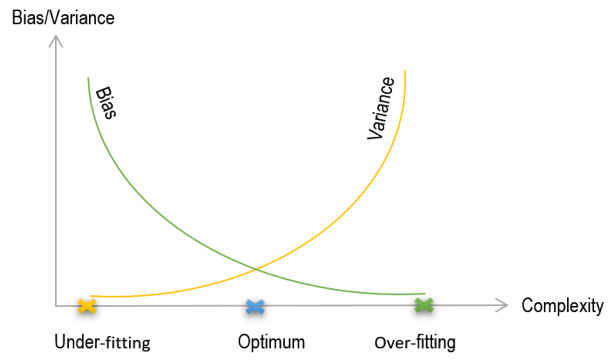


Fig. 3: Model Bias and Variance Trade-off(adopted from [13])

Ensemble techniques can be either homogeneous or heterogeneous. In the homogeneous ensemble techniques, such as Bagging, the ensemble models belong to the same ML/DL methods class. In contrast, heterogeneous ensemble techniques, such as Stacking, create a learning ensemble from different methods. Further, combining base learners is achievable using two methods: multi-expert and multi-stage [3]. In a multi-expert method, the base learning works in parallel, and their outputs are combined using a simple function such as sum, weighted sum, median, maximum, minimum, or product. However, the multi-stage method combines the base learner serially where each subsequent learner is only trainable or testable using instances where the performance of the previous learner is not satisfactory. Thus, while base learners using the multi-expert method are complementary, base learners using the multi-stage method are supplementary. This paper uses the Stacking method and heterogeneous ensemble techniques to design the ensemble model using the multi-expert method.

C. Optimization Methodology

We train and test each model (such as PCA and 1-SVM AML-IDS) using hyperparameter tuning and ensemble learning, respectively, to optimize the single-learner models. For ensemble learning, we consider four scenarios in which we combine the two AML-IDS models to create the first scenario, each of the AML-IDS models with DL-IDS to create the second and third scenarios, and finally, the AML-IDS models and DL-IDS to create the fourth scenario. We evaluate the models' detection rate, predictability, and performance using AUC, F1-Score, and accuracy as metrics and compare their performance to that of the non-optimized single-learner model. The hyperparameter tuning method uses an iterative random search to obtain the best set of hyperparameters that optimize the base-learners and meta-learner score functions, i.e., F1-Score, AUC, and accuracy. Although accuracy is not always a strong performance indicator, it may provide some insight. Fig. 4 shows the workflow of intelligent IDSs evaluation using the AMLS platform.

In general, the dataset influences the model efficacy and applicability in real applications, and the realistic of the dataset used to train intelligent IDS highly also influences their applicability in real-world applications. To that effect, we use the realistic ToN_IoT [27], [28] and UNSW_2018_IoT_Botnet [29], [30] to optimize the AML-IDS models. We optimize the AML-IDS models using one scenario of the ToN_IoT dataset and the UNSW_2018_IoT_Botnet dataset. We further processed the UNSW_2018_IoT_Botnet to include fewer features, records, and attack types than the original full dataset. The IoT_Fridge representing one scenario from the ToN_IoT dataset contains fewer features and a large number of records than the IoT_Fridge scenario dataset. Table I shows a summary of the two datasets and Table II shows some features of the two ground truth datasets. Furthermore, a full description of the data features and the frequency distribution of the attacks in the datasets are shown in [7], [29], [27].

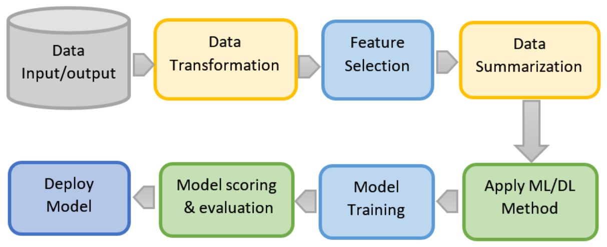


Fig. 4: Workflow for Intelligent IDS using AMLS

TABLE I: Summary of the IoT_BoTnet and ToN_IoT Datasets

Dataset	No. of Features	No. of Records	Traffic Type
IoT_Botnet	29	2,000	Normal, DoS, and DDoS
IoT_Fridge	6	587,076	DDoS, backdoor, injection, password, ransomeware, and XSS

TABLE II: Examples of Data Features

Dataset	Features
IoT_Botnet	flow state flag, protocol, source IP address and port number, destination IP address and port number, record time, number of packets, number of bytes, state, sequence number, packet count, byte count, class label, traffic category, and traffic subcategory.
IoT_Fridge	date, time, temperature measurement, temperature conditions, label, and type.

We use the following standard metrics: accuracy (A), precision (P), recall (R), and F1-Score (F) to evaluate the performance of the optimized single-learners (PCA and 1-SVM) and the ensemble-learner models using Stacking on the given IoT datasets. Also, the study considers area under the curve (AUC), receiver operating characteristic curve (ROC), True positive (TP), True negative (TN), false positive (FP), and false negatives (FN) metrics. The later four metrics are critical for calculating the A, P, R, and F metrics [10]. Table II and Table III provide a concise definition of the basic and derived metrics used in the evaluation of the optimized classifiers, as noted in the literature in [2], [31], [32], [33], [34].

IV. Experimental Setup

For learning models, We have six experimental runs for the single-learner models with hyperparameter tuning optimization and ten experimental runs for Stacking-based ensemble

TABLE III: DESCRIPTION OF DERIVED METRICS

Metrics	Description
Accuracy (A)	Number of correct predictions to the total number of predictions.
Precision (P)	Frequency of true positives among all positive output.
Recall (R)	The proportion of positive events that were classified correctly.
F1-Score (F)	The harmonic mean of the previous two values, i.e., model precision and recall. A classifier with excellent behavior has an F1-score close to 1.0.

TABLE IV: DESCRIPTION OF BASE METRICS

Metrics	Description
AUC	Indicates the degree of separability. A classifier with AUC =1 was able to classify all the observations correctly. A random classifier AUC would be 0.5.
ROC	Quantitatively measured by AUC, is a probability curve that compares the TP rate evolution versus the FP rate for different threshold values.
TP	Malicious events were detected as intrusive.
TN	Normal events were detected as normal.
FP	Normal events were detected as intrusive.
FN	Malicious events were detected as normal.

TABLE V: DESCRIPTION OF AMLS MODULES

Module	Description
Edit Metadata	A data transformation module to change the metadata associated with columns to indicate which column contains the label or values to predict, make columns as features, change the datetime to numeric values, and treat Boolean or numerical columns as categorical columns.
Select Column Data	A data transformation module to logically limit the number of columns (features) or select the features to include in the dataset.
Split Data	Data transformation module to divide a dataset into distinct sets, i.e., training and testing sets, or training, testing, and validation sets.
PCAbased Anomaly Detection	Creates an anomaly detection model based on the PCA algorithm. It analyzes available feature to determine the normal event.
One-Class SVM	Creates a model that is based on the SVM algorithm.
Two-Class NN	Creates a model that is based on a neural network to predict a target that has a binary value only.
Score Model	A scoring module to generate predictions using a trained classification or regression module.
Evaluate Model	An evaluation module to measure the accuracy of a trained model.
Train Anomaly Detection Model	An anomaly detection module that takes a set of parameters and an unlabeled dataset as input and trains an anomaly detection model and a set of labels for the training data.
Train Model	A training module to train the classification or regression model.
Tune Model Hyperparameters	Optimizes a given ML model by determining the optimum hyperparameters or model structure.

techniques. For the hyperparameter tuning experiments, the dataset split is 80% for training the model and 20% for testing it. For ensemble-learner models, the data split is the same for training models using IoT_Botnet dataset and 50% for training models using IoT_Fridge dataset. The 50% split in the dataset for IoT_Fridge was due to the significant amount of time spent training and testing the models. The datasets are made available in .csv format via an AMLS upload file operation. The model's construction is achieved via the AMLS drag and drop functionality and a set of predefined modules, which allows the user to create, tune, train, test, score, and evaluate the models for experimentation; see Table IV.

The three initial IDS models, PCA AML-IDS, 1-SVM AML-IDS, and 2-NN DL-IDS, did not consider optimization to boost performance and increase detection rates while reducing false alarms. The DL-IDS model does not require optimization due to its superior performance compared to the other two IDS. We describe the optimized single-learner experimental setup and evaluation results using hyperparameter tuning and Stacking-based ensemble learning in the following subsections.

A. Optimized Single-learner Models Experimental Runs

Generally, PCA and 1-SVM AML-IDS models analyze the feature space using imbalanced data to determine the non-anomalous events (i.e., normal class). Determining what constitutes a normal class requires training the models using datasets containing more or all normal examples. Further, 1-SVM AML-IDS models are better trainable on data with one class, i.e., the normal class. Our study uses both a balanced and an imbalanced dataset to evaluate the models' performance and optimize the features space to find the optimal single-learner model structure.

The optimized single-learner experimental runs focus on hyperparameter tuning to define the optimal model structure. The goal is to improve the PCA and 1-SVM AML-IDS models' performance using the IoT_Botnet dataset and 1-SVM AML-IDS using the IoT_Fridge dataset. We conducted two experiments for each of these three low-performing models, where we optimized the model based on AUC and F1-Score metrics. Each experiment builds an optimized single-learner model based on one of the specified metrics using the following steps: *1) Supply the dataset; Extract the features; 2) Select the ML or DL algorithms; 3) Split the data into training and testing data (for 1-SVM, 4) split the data further to select only the normal examples); 5) tune model hyperparameters based on specified metrics; 6) Train the model; 7) Score the model; and 8) Evaluate the model.*

B. Ensemble-learner Models Experimental Runs

In contrast to single-learner models, ensemble-learner models build on each model's strength in the ensemble to offset the individual model weaknesses, combine their collective knowledge, and aggregate their output to train a meta-model. This meta-model, which we also call the ensemble-learner, learns from other models to improve its performance and prediction accuracy. Our study uses the Stacking technique to train low-performing models using various Stacking by enumerating all possible combinations of the three pre-defined

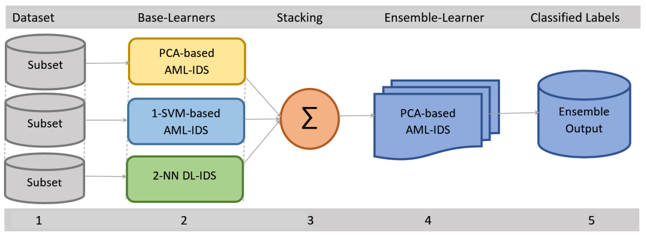


Fig. 5: A schematic diagram depicting a triple Stacking-based ensemble using a combination of PCA AML-IDS, 1-SVM AML-IDS, and 2-NN DL-IDS models

models (PCA AML-IDS, 1-SVM AML-IDS, and 2-NN DL-IDS). First, we use dual Stacking, in which we combine two different models using all possible combinations, and triple Stacking, in which we combine all three models to form the ensemble. Then, we use similar steps to that in [35] to build the ensemble and train and evaluate the meta-model using the metric that yields the best intrusion detection results as follows:

1) Combine n methods or classifiers ;
2) Find a method or classifier that optimizes AUC;
3) Select the ML or DL algorithms;
4) build n base methods or classifiers using the full training set and obtain baseline AUC performance;
5) build a stacked ensemble of methods or classifiers;
6) Split the training set into set 1 and set 2;
   a) Use Set 1 to train n base methods or classifiers
   b) Use Set 2 to find their best combination
7) Tune the base method or classifiers and their ensemble to optimize AUC; and
8) Output the final results by executing the R model.

To obtain the best model, first, we find a method or classifier that optimizes AUC, F1-Score, or accuracy and then get the ensemble's corresponding baseline performance. Fig. 5. shows a schematic diagram depicting a triple Stacking-based ensemble.

Given specific ensembles and their tuning methods, i.e., tuned to optimize AUC, F1-Score, or accuracy, the initial analysis of the three ensemble experiments reveals an improvement in the performance and predictability of the models in question. Next, we present the detailed analysis of the performance of the optimized single-learner and ensemble-learner IDS models in the following section.

V. Performance Evaluation and Analysis

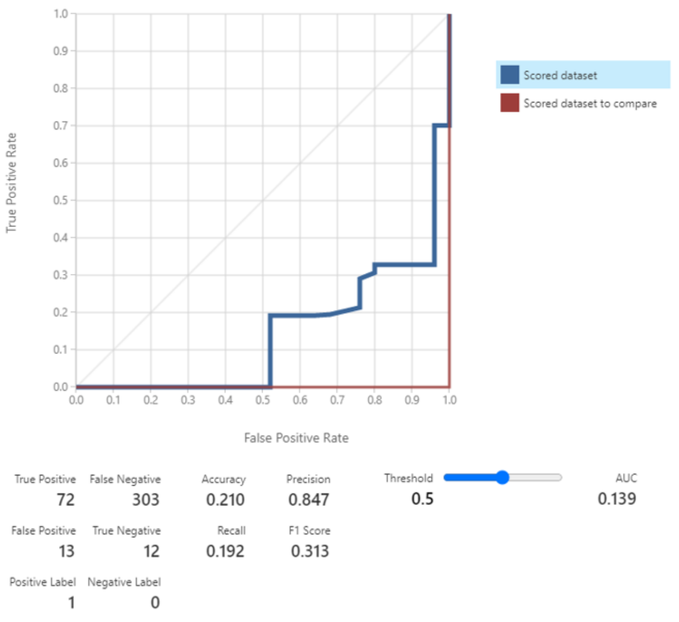
This section provides a performance evaluation to support our analysis presented above.

A. *Optimized Single-learner Models Analysis*

To find the best model structure that contains the optimal set of parameters for controlling the learning process, we optimized the three intelligent IDS models that exhibited low performance on the corresponding dataset using the AUC and F1-score metrics instead of accuracy. AUC provides a better indication of the trained model's performance based on its degree of separability, i.e., its ability to separate normal from anomalous examples. The F1-Score gives a good indication of the quality of the classification. A model with a high F1-Score value has high predictability, i.e., low FP and FN. While accuracy gives the intuition on how often the model predictions are correct, it may lead to erroneous results when the data used to train the models has an imbalanced distribution, e.g., the number of normal and anomalous examples varies significantly.

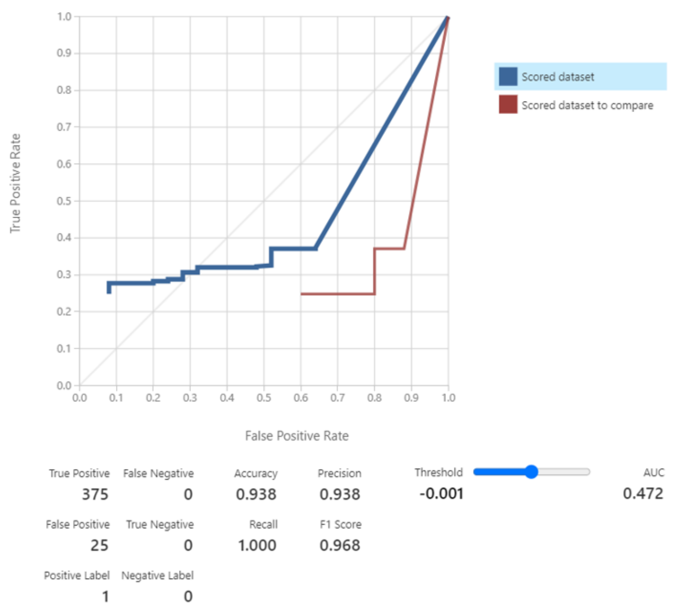
To obtain better results, We designed the three IDS models (PCA AML-IDS using IoT_Botnet dataset, 1-SVM AML-IDS using IoT_Botnet dataset, and 1-SVM AML-IDS using IoT_Fridge dataset) by adding the tune hyperparameter model module. We tuned the models' hyperparameters to optimize AUC and F1-Score and conducted three experimental runs. Fig. 6, Fig. 7, Fig. 8 show the PCA and 1-SVM AML-IDS models' evaluation results using IoT_Botnet dataset and 1-SVM AML-IDS model's evaluation using IoT_Fridge dataset, respectively. The ROC and metric values are the same for AUC and F1-Score optimization.

Tuning the model hyperparameters to optimize AUC has led to improved models' performance and predictability to varying degrees depending on the model and the selected tuning metrics. Given the IoT_Botnet dataset tuned to optimize AUC, the 1-SVM AML-IDS model has shown better performance and predictability than the PCA AML-IDS model. The AUC for the 1-SVM AML-IDS model is 0.472 (47%) with tuning compared to 0.155 (15%) without tuning, and the F1-Score is 0.968 (97%). The FN is 0 for the tuned model compared to 236 for the non-tuned model. On the other hand, PCA AML-IDS AUC is 0.139 (13.9%) with tuning compared to 0.002 (0.2%) without tuning, and the F1-Score is 0.313 with tuning compared to 0.272 (27.2%) without tuning, suggesting a high FP, FN, or both. The FN is 303 for the tuned model and 312



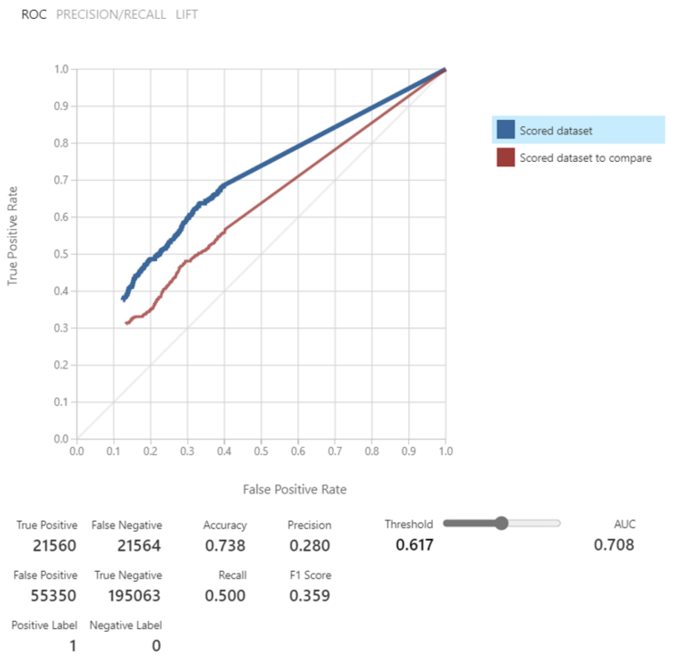
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
72	303	0.210	0.847	0.5	0.139
False Positive	**True Negative**	**Recall**	**F1 Score**		
13	12	0.192	0.313		
Positive Label	**Negative Label**				
1	0				

Fig. 6: Tuned PCA AML-IDS model for AUC and F1-score optimization using IoT_Botnet experimental run results.



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
375	0	0.938	0.938	-0.001	0.472
False Positive	**True Negative**	**Recall**	**F1 Score**		
25	0	1.000	0.968		
Positive Label	**Negative Label**				
1	0				

Fig. 7: Tuned 1-SVM AML-IDS model for AUC and F1-Score optimization using IoT_Botnet dataset experimental run results



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
21560	21564	0.738	0.280	0.617	0.708
False Positive	**True Negative**	**Recall**	**F1 Score**		
55350	195063	0.500	0.359		
Positive Label	**Negative Label**				
1	0				

Fig. 8: Tuned 1-SVM AML-IDS model for AUC and F1-score optimization using IoT_Fridge dataset experimental run results

for the non-tuned model.

B. Ensemble-learner Models Analysis

In general, ensemble-learning using Stacking techniques improves the meta-model's performance and predictability. However, such improvement depends on the selected base learners, tuning of the base learners and their ensemble tuning, and the optimization metric. Given the IoT_Botnet dataset, the three types of ensembles, and the metrics to optimize, the resulting PCA AML-IDS metal-model showed significant

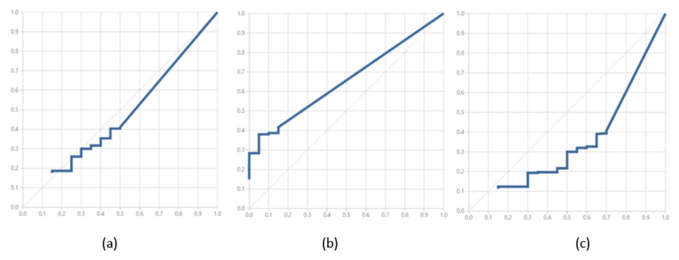


Fig. 9: Dual Stacking ROC: PCA and SVM using IoT_Botnet dataset and optimized using (a) AUC, (b) F1-Score, and (c) Accuracy

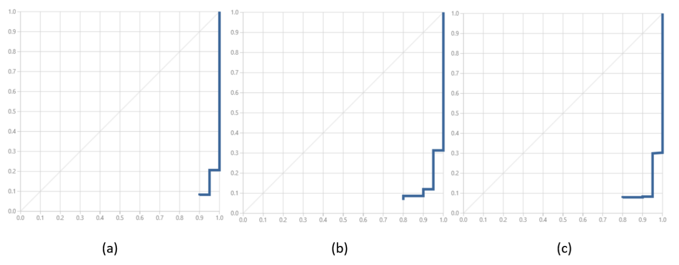


Fig. 10: Dual Stacking ROC: PCA and NN using IoT_Botnet dataset and optimized using (a) AUC, (b) F1-Score, and (c) Accuracy

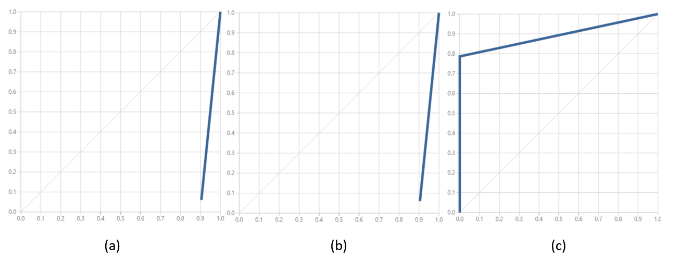
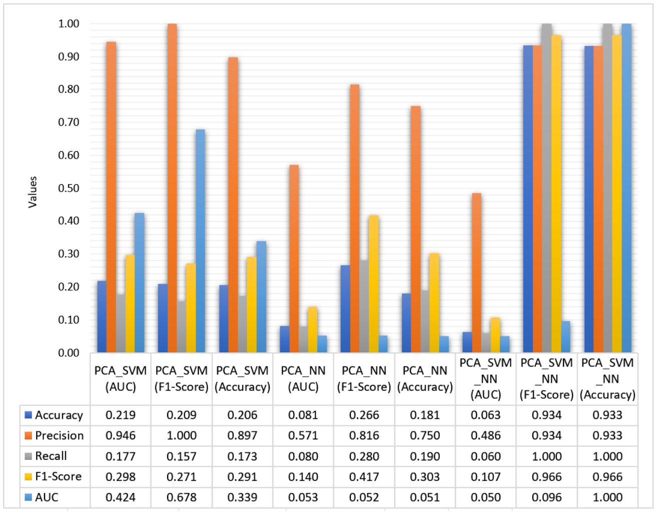


Fig. 11: Triple Stacking ROC: PCA, SVM, and NN using IoT_Botnet dataset and optimized using (a) AUC, (b) F1-Score, and (c) Accuracy

improvement when using a PCA-SVM-NN ensemble and then tuning the base-model hyperparameters to optimize accuracy. The resulting meta-model has a 0.966 (97%), suggesting all relevant examples are labeled and an AUC value of 1, suggesting a high performance. Nonetheless, accuracy is not always an accurate indication of the model's performance and predictability. Fig. 9, Fig. 10, and Fig. 11 show the ROC of the PCA AML-IDS meta-model trained using IoT_Botnet dataset and a PCA-SVM ensemble, PCA-NN ensemble, and PCA-SVM-NN ensemble,respectively.

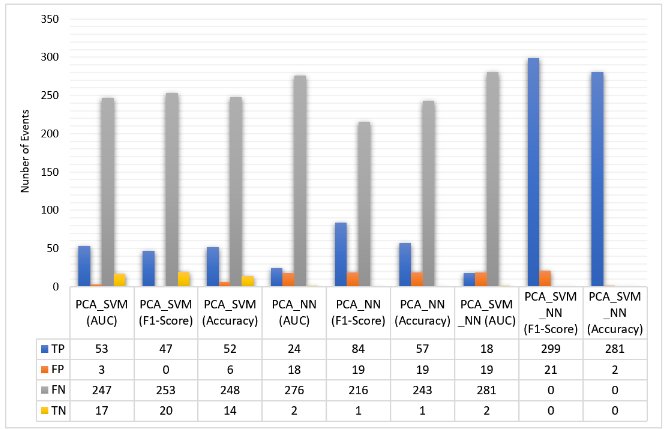
Each ensemble optimized AUC, F1-Score, and accuracy using hyperparameter tuning as shown in (a), (b), and (c), respectively. Fig. 12 and Fig. 13 show the plotted values of accuracy, precision, recall, and AUC and the plotted values of TP, FP, TN, and FN, respectively.

The 1-SVM AML-IDS meta-model trained using IoT_Botnet has shown a significant improvement in performance and predictability when trained using SVM-PCA ensemble to optimize F1-score, and accuracy, SVM-NN



	PCA_SVM (AUC)	PCA_SVM (F1-Score)	PCA_SVM (Accuracy)	PCA_NN (AUC)	PCA_NN (F1-Score)	PCA_NN (Accuracy)	PCA_SVM _NN (AUC)	PCA_SVM _NN (F1-Score)	PCA_SVM _NN (Accuracy)
Accuracy	0.219	0.209	0.206	0.081	0.266	0.181	0.063	0.934	0.933
Precision	0.946	1.000	0.897	0.571	0.816	0.750	0.486	0.934	0.933
Recall	0.177	0.157	0.173	0.080	0.280	0.190	0.060	1.000	1.000
F1-Score	0.298	0.271	0.291	0.140	0.417	0.303	0.107	0.966	0.966
AUC	0.424	0.678	0.339	0.053	0.052	0.051	0.050	0.096	1.000

Fig. 12: Dual and Triple Stacking: graphed derived metrics for PCA and SVM, PCA and NN, and PCA, SVM, and NN using the IoT_Botnet dataset and AUC, F1-Score, and Accuracy optimization



	PCA_SVM (AUC)	PCA_SVM (F1-Score)	PCA_SVM (Accuracy)	PCA_NN (AUC)	PCA_NN (F1-Score)	PCA_NN (Accuracy)	PCA_SVM _NN (AUC)	PCA_SVM _NN (F1-Score)	PCA_SVM _NN (Accuracy)
TP	53	47	52	24	84	57	18	299	281
FP	3	0	6	18	19	19	19	21	2
FN	247	253	248	276	216	243	281	0	0
TN	17	20	14	2	1	1	2	0	0

Fig. 13: Dual and Triple Stacking: graphed base metrics for PCA and SVM, PCA and NN, and PCA, SVM, and NN using the IoT_Botnet dataset and AUC, F1-Score, and Accuracy optimization

ensemble optimizing AUC, F1-Score, and accuracy. For the mentioned ensembles, the meta-model's AUC is 1, and the F1-score 0.968 (97%) or 0.969 (97%), showing high performance and predictability. However, for the SVM-PCA-NN ensemble, the meta-model did not show comparable results. While the meta-model recall is 0.966 (97%) for SVM-PCA-NN optimizing AUC and F1-score, respectively, the performance is significantly low (9.5%). Fig. 14, Fig. 15, and Fig. 16 show the ROC of the SVM-based IDS meta-model trained using IoT_Botnet dataset and an SVM-PCA ensemble, SVM-NN ensemble, and SVM-PCA-NN ensemble, respectively.

Similar to the PCA AML-IDS model, each ensemble optimized AUC, F1-Score, and accuracy using hyperparameter tuning as shown in (a), (b), and (c), respectively. Fig. 17 and Fig. 18 show the charted values of accuracy, precision, recall, and AUC and the charted values of TP, FP, TN, and

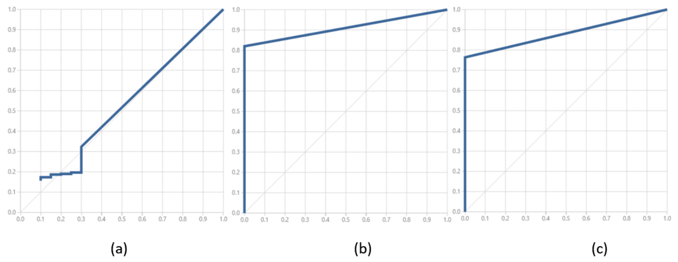


Fig. 14: Dual Stacking: SVM and PCA using IoT_Botnet dataset and optimized using (a) AUC, (b) F1-Score, and (c) Accuracy

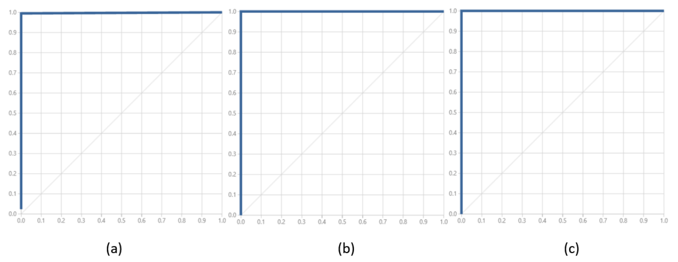


Fig. 15: Dual Stacking: SVM and NN using IoT_Botnet dataset and optimized using (a) AUC, (b) F1-Score, and (c) Accuracy

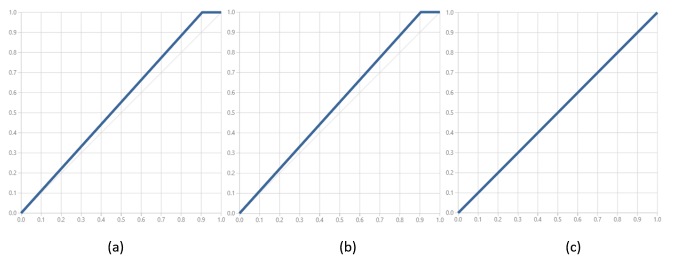
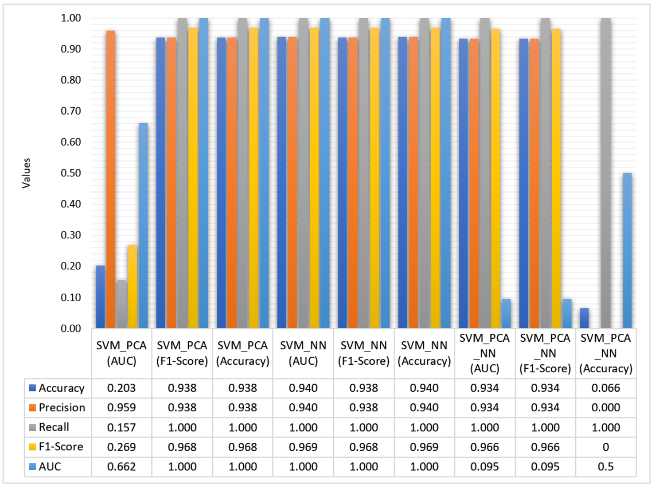


Fig. 16: Triple Stacking: SVM, PCA, and NN using IoT_Botnet dataset and optimized using (a) AUC, (b) F1-Score, and (c) Accuracy

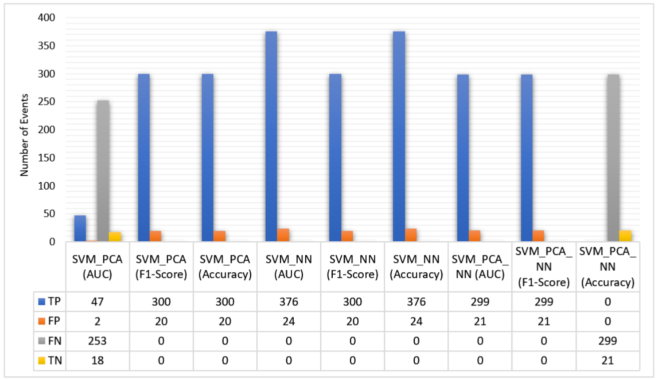
FN, respectively.

The ensemble-learning using the IoT_Fridge dataset has shown promising results for the 1-SVM AML-IDS model. The 1-SVM AML-IDS meta-model exhibited high performance when trained using the SVM-PCA and SVM-NN and AUC, F1-Score, and accuracy optimization. The meta-model trained using the SVM-NN ensemble showed superior performance and predictability with an AUC and F1-score value of 1 for AUC, F1-Score, and accuracy optimization. Also, the meta-model has high accuracy, precision, and recall of 1. While the meta-model performance using the SVM-PCA is 100%, the predictability is 87.5%, 89.4%, and 87.2% for the AUC, F1-Score, and accuracy optimization, respectively. Fig. 19, Fig. 20, and Fig. 21 show the ROC of the PCA-based IDS meta-model trained using IoT_Botnet dataset and a PCA-SVM ensemble, PCA-NN ensemble, and PCA-SVM-NN ensemble, respectively. Each ensemble optimized AUC, F1-Score, and accuracy using hyperparameter tuning as shown in (a), (b), and (c), respectively. Fig. 22 and Fig. 23 show the charted



	SVM_PCA (AUC)	SVM_PCA (F1-Score)	SVM_PCA (Accuracy)	SVM_NN (AUC)	SVM_NN (F1-Score)	SVM_NN (Accuracy)	SVM_PCA_NN (AUC)	SVM_PCA_NN (F1-Score)	SVM_PCA_NN (Accuracy)
Accuracy	0.203	0.938	0.938	0.940	0.938	0.940	0.934	0.934	0.066
Precision	0.959	0.938	0.938	0.940	0.938	0.940	0.934	0.934	0.000
Recall	0.157	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F1-Score	0.269	0.968	0.968	0.969	0.968	0.969	0.966	0.966	0
AUC	0.662	1.000	1.000	1.000	1.000	1.000	0.095	0.095	0.5

Fig. 17: Dual and Triple Stacking: graphed derived metrics for SVM and PCA, SVM and NN, and SVM, PCA, and NN using the IoT_Botnet dataset and AUC, F1-Score, and Accuracy optimization



	SVM_PCA (AUC)	SVM_PCA (F1-Score)	SVM_PCA (Accuracy)	SVM_NN (AUC)	SVM_NN (F1-Score)	SVM_NN (Accuracy)	SVM_PCA_NN (AUC)	SVM_PCA_NN (F1-Score)	SVM_PCA_NN (Accuracy)
TP	47	300	300	376	300	376	299	299	0
FP	2	20	20	24	20	24	21	21	0
FN	253	0	0	0	0	0	0	0	299
TN	18	0	0	0	0	0	0	0	21

Fig. 18: Dual and Triple Stacking: graphed base metrics for SVM and PCA, SVM and NN, and SVM, PCA, and NN using the IoT_Botnet dataset and AUC, F1-Score, and Accuracy optimization

values of accuracy, precision, recall, and AUC and the charted values of TP, FP, TN, and FN, respectively.

The above results show that while some ensembles improved performance and predictability, others provided either slight or zero improvement. While the 2-NN DL-IDS model in the non-optimized single-learner scenario scored well, showing high performance and good predictability, when used in an ensemble of a heterogeneous type, for some instances, it did not affect the outcome to boost the performance and predictability of the model, e.g., when combined with PCA and 1-SVM. The recall and F1-Score values are 0, suggesting no anomalous events and the inability of the model to predict anomalous events.

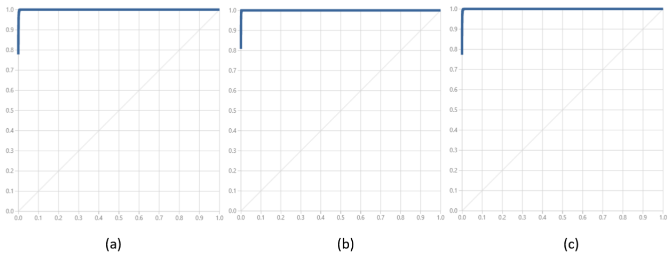


Fig. 19: Dual Stacking: SVM and PCA using IoT_Fridge dataset and optimized using (a) AUC, (b) F1-Score, and (c) Accuracy

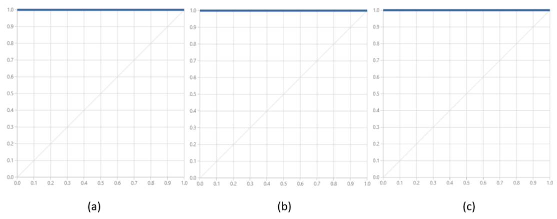


Fig. 20: Dual Stacking: SVM and NN using IoT_Fridge dataset and optimized using (a) AUC, (b) F1-Score, and (c) Accuracy

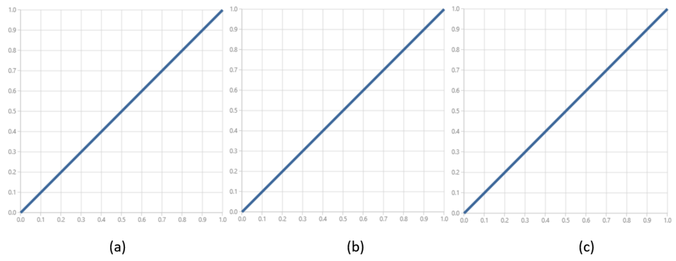
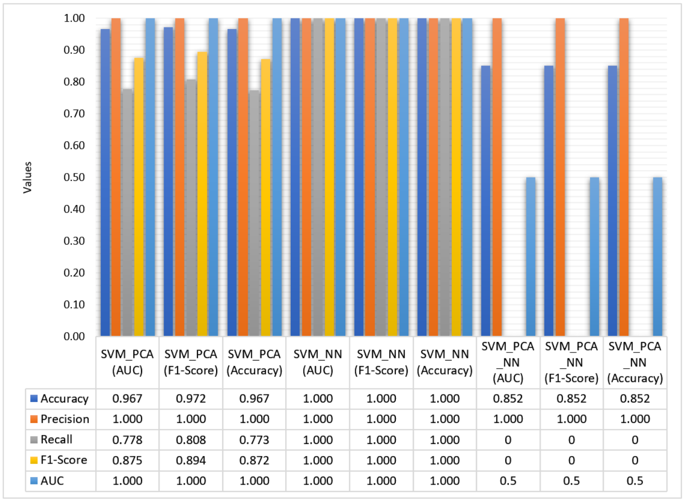


Fig. 21: Triple Stacking: SVM, PCA, and NN using IoT_Fridge dataset and optimized using (a) AUC, (b) F1-Score, and (c) Accuracy

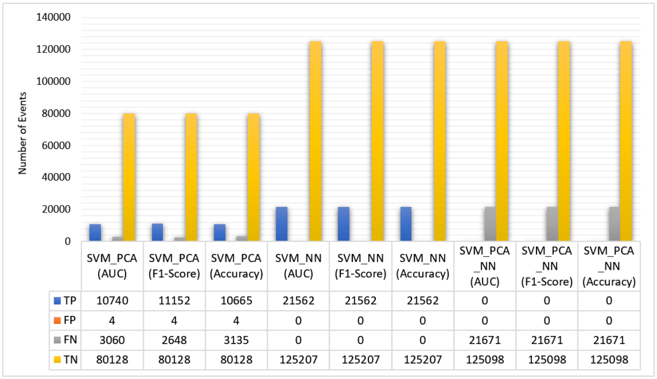


	SVM_PCA (AUC)	SVM_PCA (F1-Score)	SVM_PCA (Accuracy)	SVM_NN (AUC)	SVM_NN (F1-Score)	SVM_NN (Accuracy)	SVM_PCA_NN (AUC)	SVM_PCA_NN (F1-Score)	SVM_PCA_NN (Accuracy)
Accuracy	0.967	0.972	0.967	1.000	1.000	1.000	0.852	0.852	0.852
Precision	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Recall	0.778	0.808	0.773	1.000	1.000	1.000	0	0	0
F1-Score	0.875	0.894	0.872	1.000	1.000	1.000	0	0	0
AUC	1.000	1.000	1.000	1.000	1.000	1.000	0.5	0.5	0.5

Fig. 22: Dual and Triple Stacking: graphed derived metrics for SVM and PCA, SVM and NN, and SVM, PCA, and NN using the IoT_Fridge dataset and AUC, F1-Score, and Accuracy optimization

VI. Open Research Challenges

Ensemble learning is costly in terms of training time, testing time, and computational overhead. Consequently, this leads



	SVM_PCA (AUC)	SVM_PCA (F1-Score)	SVM_PCA (Accuracy)	SVM_NN (AUC)	SVM_NN (F1-Score)	SVM_NN (Accuracy)	SVM_PCA _NN (AUC)	SVM_PCA _NN (F1-Score)	SVM_PCA _NN (Accuracy)
TP	10740	11152	10665	21562	21562	21562	0	0	0
FP	4	4	4	0	0	0	0	0	0
FN	3060	2648	3135	0	0	0	21671	21671	21671
TN	80128	80128	80128	125207	125207	125207	125098	125098	125098

Fig. 23: Dual and Triple Stacking: graphed base metrics for SVM and PCA, SVM and NN, and SVM, PCA, and NN using the IoT_Fridge dataset and AUC, F1-Score, and Accuracy optimization

to high latency and resource utilization which significantly impacts intelligent IDS real-time capability. A distributed intelligent IDS architecture can help reduce the computational overhead, hence reduced latency. To decrease the false alarm and increase detection accuracy in ensemble learning, emerging approaches that use neutrosophic logic classifiers and Genetic algorithms to generate rules are showing promising potential [11].

In general, anomaly-based IDS models suffer from high false positives due to using one rule that focuses on detecting intrusions based on the normal traffic pattern. As a result, it considers all other traffic that does not conform to the normal pattern as anomalous. A better approach is to use a set of rules to filter non-anomalous traffic that does not conform to the normal pattern of behavior, which is achievable using meta-heuristic algorithms such as genetic algorithms.

VII. Conclusion and Future Work

In this paper, we studied optimization of PCA and 1-SVM AML-IDS models using hyperparameter tuning and ensemble learning to detect the intrusions in IoT. Specifically, we performed hyperparameter tuning to optimize these models by selecting the optimal model structure. then we used ensemble learning with the Stacking technique to create multiple learners and train the single-learner AML-IDS models to boost the single-learner model detection performance. We trained these optimized models using ground-truth network intrusion datasets containing both malicious and benign IoT and IIoT traffic and evaluated their performance using typical ML/DL metrics. In general, the SVM-based IDS model offered better results when a heterogeneous ensemble was used. For large datasets where normal examples occur more frequently, the SVM-based IDS model exhibited high performance and good predictability with ensembles involving SVM-PCA and SVM-NN and hyperparameter tuning to optimize AUC, F-Score, and accuracy. However, the model did not show similar results when using the SVM-PCA-NN ensemble. Single-learner 2-NN DL-IDS models have shown superior performance detecting intrusions in IoT environments irrespective of the size of the dataset, imbalances within the dataset, dataset frequency distribution, and the number of features. However, stacking high-performing classifiers such as neural networks and low performing classifiers such as PCA and 1-SVM in an ensemble does not necessarily generate a high-performing ensemble classifier.

In the future, we plan to examine the use of the Boosting ensemble learning method and Genetic algorithms in ensemble learning to improve AML-IDS detection rate and performance by reducing the false-positive rate and minimizing the training and testing time. Also, we plan to address issues related to latency and resource utilization, adaptability to the dynamic IoT environment, and deployment architecture.

References

[1] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, and A. Wahab, "A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions," *Electronics (Basel)*, vol. 9, no. 7, pp. 1177–, 2020.

[2] B. Kelem, "Comparison of machine learning techniques for intrusion detection system," phdthesis, National Academic Digital Repository of Ethiopia, 2018. [Online]. Available: https://search.datacite.org/works/10.20372/nadre/6054

[3] P. Illy, G. Kaddoum, C. Miranda Moreira, K. Kaur, and S. Garg, "Securing Fog-to-Things Environment Using Intrusion Detection System Based On Ensemble Learning," in *IEEE Wireless Comm and Networking Conf (WCNC)*, 4 2019, pp. 1–7.

[4] A. Uprety and D. B. Rawat, "Reinforcement Learning for IoT Security: A Comprehensive Survey," *IEEE Internet of Things Journal*, vol. 8, no. 11, 2021.

[5] X. Liu, G. Yan, D. B. Rawat, and S. Deng, "Data mining intrusion detection in vehicular ad hoc network," *IEICE TRANSACTIONS on Information and Systems*, vol. 97, no. 7, pp. 1719–1726, 2014.

[6] F. O. Olowononi, D. B. Rawat, and C. Liu, "Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 524 1– 552, 2021.

[7] G. Abdelmoumin and D. B. Rawat, "A Comparative Study of Intelligent Intrusion Detection Systems for Internet of Things," in *Springer Lecture Notes in Networks and Systems*. Springer, 28-29 October 2021, to appear.

[8] X. Zhang. ML studio (classic): One-class support vector machine - azure (online accessed on 10 may 2021). [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/one-class-support-vector-machine

[9] M. Chattopadhyay, R. Sen, and S. Gupta, "A comprehensive review and meta-analysis on applications of machine learning techniques in intrusion detection," *Australasian J. of Info. Sys.*, vol. 22, 2018.

[10] S. Laqtib, K. E. Yassini, and M. L. Hasnaoui, "A deep learning methods for intrusion detection systems based machine learning in MANET," in *Proceedings of the 4th International Conference on Smart City Applications*, ser. SCA '19. Association for Computing Machinery, 2019-10-02, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3368756.3369021

[11] S. Rajagopal, P. P. Kundapur, and K. S. Hareesha, "A stacking ensemble for network intrusion detection using heterogeneous datasets," *Security and Communication Networks*, vol. 2020, p. e4586875, 1 2020.

[12] V. Dutta, M. Choraś, M. Pawlicki, and R. Kozik, "A deep learning ensemble for network anomaly and cyber-attack detection," *Sensors*, vol. 20, no. 16, p. 4583, 1 2020.

[13] H. Thanh and T. Lang, "Use the ensemble methods when detecting DoS attacks in network intrusion detection systems," *EAI Endorsed Transactions on Context-aware Systems and Applications*, vol. "6", no. 19, 11 2019.

[14] W. Lian, G. Nie, B. Jia, D. Shi, Q. Fan, and Y. Liang, "An intrusion detection method based on decision tree-recursive feature elimination in ensemble learning," *Mathematical Problems in Engineering*, vol. 2020, p. e2835023, 11 2020. [Online]. Available: https://www.hindawi.com/journals/mpe/2020/2835023/

[15] A. Fahada, "A semi-stack approach for accurate network traffic classification using multi-view stacking," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 811, p. 012026, 5 2020.

[16] M. A. Jabbar, R. Aluvalu, and S. S. S. Reddy, "Cluster based ensemble classification for intrusion detection system," in *Proceedings of the 9th International Conference on Machine Learning and Computing*, ser. ICMLC 2017. Association for Computing Machinery, 2 2017, pp. 253–257. [Online]. Available: https://doi.org/10.1145/3055635.3056595

[17] J. Liu, J. He, W. Zhang, T. Ma, Z. Tang, J. P. Niyoyita, and W. Gui, "ANID-SEoKELM: Adaptive network intrusion detection based on selective ensemble of kernel ELMs with random features," *Knowledge-Based Systems*, vol. 177, pp. 104–116, 08 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095070511930173X

[18] G. Kumar, K. Thakur, and M. R. Ayyagari, "MLEsIDSs: machine learning-based ensembles for intrusion detection systems—a review," *The Journal of supercomputing*, vol. 76, no. 11, pp. 8938–8971, 2020.

[19] M. Kim, "Supervised learning-based DDoS attacks detection: Tuning hyperparameters," *ETRI Journal*, vol. 41, no. 5, pp. 560–573, 2019, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.4218/etrij.2019-0156.

[20] J. Liu and E. Zio, "SVM hyperparameters tuning for recursive multi-step-ahead prediction," *Neural Comput & Applic*, vol. 28, no. 12, pp. 3749–3763, 12 2017. [Online]. Available: https://doi.org/10.1007/s00521-016-2272-1

[21] R. G. Mantovani, A. L. D. Rossi, E. Alcobaça, J. Vanschoren, and A. C. P. d. L. F. de Carvalho, "A meta-learning recommender system for hyperparameter tuning: predicting when tuning improves SVM classifiers," *Information Sciences*, vol. 501, pp. 193–221, 10 2019. [Online]. Available: http://arxiv.org/abs/1906.01684

[22] A. Stuke, P. Rinke, and M. Todorović, "Efficient hyperparameter tuning for kernel ridge regression with bayesian optimization," *Mach. Learn.: Sci. Technol.*, 03 2021.

[23] A. Zheng, *Evaluating Machine Learning Models: A Beginner's Guide to Key Concepts and Pitfalls*. O'Reilly Media, 9 2015. [Online]. Available: https://books.google.com/books?id=OFhauwEACAAJ

[24] B. Li and P. Lu. (2020, 10) Tune model hyperparameters. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-module-reference/tune-model-hyperparameters

[25] P. N. Tattar, *Hands-On Ensemble Learning with R: A Beginner's Guide to Combining the Power of Machine Learning Algorithms Using Ensemble Techniques*. Packt Publishing, Limited, 2018.

[26] Nouf Rahimi, Fathy Eassa, and Lamiaa Elrefaei, "An ensemble machine learning technique for functional requirement classification," *Symmetry (Basel)*, vol. 12, no. 1601, pp. 1601–, 2020.

[27] N. Moustafa, "ToN_iot datasets," 10 2019. [Online]. Available: https://ieee-dataport.org/documents/toniot-datasets

[28] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_iot telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020.

[29] N. Moustafa, "The bot-IoT dataset," 10 2019. [Online]. Available: https://ieee-dataport.org/documents/bot-iot-dataset

[30] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 11 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18327687

[31] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 686–728, 2019.

[32] Y. Hamid, M. Sugumaran, and L. Journaux, "Machine learning techniques for intrusion detection: A comparative analysis," in *Proceedings of the International Conference on Informatics and Analytics*, ser. ICIA-16. Association for Computing Machinery, 8 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1145/2980258.2980378

[33] R. Magán-Carrión, D. Urda, I. Díaz-Cano, and B. Dorronsoro, "Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches," *Applied sciences*, vol. 10, no. 5, pp. 1775–, 2020.

[34] 20 popular machine learning metrics. part 1: Classification & regression evaluation metrics. [Online]. Available: https://bit.ly/3vZyUgm

[35] D. Pechyony. (2015, 02) Building ensemble of classifiers using stacking. [Online]. Available: https://gallery.azure.ai/Experiment/Building-Ensemble-of-Classifiers-using-Stacking-2

Ghada Abdelmoumin (IEEE Student Member) is currently pursuing her PhD in computer science under the supervision of Dr. Danda B Rawat at Howard University and a professor of information technology at Northern Virginia Community College. Ghada received a Bachelor of Science degree in mechanical power engineering from Alexandria University, a public research institution that is the second largest university in Egypt. She also earned Master of Science in computer science from Western Illinois University and Master of Science in computer science and applications from Virginia Tech. She is a member of Association for Computing Machinery (ACM), IEEE Young Professionals and a GEM Fellow. Her teaching and research interests have focused on machine learning, IoT, cybersecurity, cloud computing, mobile computing, and software engineering.

Danda B. Rawat (IEEE Senior Member, 2013) is a Full Professor in the Department of Electrical Engineering & Computer Science (EECS), Director of the Howard University Data Science and Cybersecurity Center, Director of DoD Center of Excellence in AI/ML (CoE-AIML), Director of Cyber-security and Wireless Networking Innovations (CWiNs) Research Lab, Graduate Program Director of Graduate CS Programs and Director of Graduate Cybersecurity Certificate Program at Howard University, Washington, DC, USA. Dr. Rawat is engaged in research and teaching in the areas of cybersecurity, machine learning, big data analytics and wireless networking for emerging networked systems including cyber-physical systems, Internet-of-Things, multi domain operations, smart cities, software defined systems and vehicular networks. He has secured over $16 million in research funding from the US National Science Foundation (NSF), US Department of Homeland Security (DHS), US National Security Agency (NSA), US Department of Energy, National Nuclear Security Administration (NNSA), DoD and DoD Research Labs, Industry (Microsoft, Intel, etc.) and private Foundations. Dr. Rawat is the recipient of NSF CAREER Award in 2016, Department of Homeland Security (DHS) Scientific Leadership Award in 2017, Researcher Exemplar Award 2019 and Graduate Faculty Exemplar Award 2019 from Howard University, the US Air Force Research Laboratory (AFRL) Summer Faculty Visiting Fellowship in 2017, Outstanding Research Faculty Award (Award for Excellence in Scholarly Activity) at GSU in 2015, the Best Paper Awards (IEEE CCNC, IEEE ICII, BWCA) and Outstanding PhD Researcher Award in 2009. He has delivered over 20 Keynotes and invited speeches at international conferences and workshops. Dr. Rawat has published over 200 scientific/technical articles and 10 books. He has been serving as an Editor/Guest Editor for over 50 international journals including the Associate Editor of IEEE Transactions of Service Computing, Editor of IEEE Internet of Things Journal, Associate Editor of IEEE Transactions of Network Science and Engineering and Technical Editors of IEEE Network. He has been in Organizing Committees for several IEEE flagship conferences such as IEEE INFOCOM, IEEE CNS, IEEE ICC, IEEE GLOBECOM and so on. He served as a technical program committee (TPC) member for several international conferences. He served as a Vice Chair of the Executive Committee of the IEEE Savannah Section from 2013 to 2017. Dr. Rawat is a Senior Member of IEEE and ACM, a member of ASEE and AAAS, and a Fellow of the Institution of Engineering and Technology (IET). He is an ACM Distinguished Speaker.

Abdul Rahman is a subject matter expert in the design and implementation of cloud analytics and architectures that support Cyber Situational Awareness (SA) tools and cyber defense capabilities. He has led large scale programs as the chief architect and chief engineer responsible for cyber network defense architectures and analytic capabilities (ML, AI, hybrid) focused on integrated full spectrum cyber operations supporting defensive and offensive cyber mission synergy. Dr. Rahman has over 25 years of information technology (IT) experience and has published in physics, mathematics, and information technology. He holds Ph.D.s in mathematics and physics.