International Journal of Gender, Science and Technology
ISSN: 2040-0748
UGC Care Group I Journal.
Vol-13 Issue-02 Aug 2024

FEATURE ENGINEERING AND MACHINE LEARNING FRAMEWORK FOR DDOS ATTACK DETECTION IN THE STANDARDIZED INTERNET OF THINGS

[1]K.Jaya Krishna, [2]Kakunuri Raja sekhar,

[1]Associate Professor, Department of Master of Computer Applications, QIS College of Engineering & Technology, Ongole, Andhra Pradesh, India

[2]PG Scholar, Department of Master of Computer Applications, QIS College of Engineering & Technology, Ongole, Andhra Pradesh, India

Abstract: The proliferation of Internet of Things (IoT) devices and networks faces challenges due to resource constraints on energy, memory, communication, and computation power. Security mechanisms in these networks are often neglected, leading to an increased risk of cyber attacks. As cyber attacks rise, there is a growing need to secure IoT networks with machine learning-based intrusion detection systems (ML-IDS) for higher accuracy and reduced false alarms. Existing benchmark datasets are outdated and lack IoT-compatible traffic data, prompting the exploration of a novel dataset. The Project proposes a two-phase framework to detect Distributed Denial-of-Service (DDoS) attacks using IoT-CIDDS. In the first phase, algorithms are developed for dataset enrichment, emphasizing advanced feature engineering, statistical analysis, and exploration of probability distribution and feature correlation. In the second phase, an ML model is proposed, and complexity analysis is performed using five ML techniques. Training, validation, and testing datasets are created from IoT-CIDDS. The ML models are evaluated based on accuracy, precision, recall, area under the curve, false positive rate, and computational time for training the classifiers. Experimental results demonstrate that substantial feature reduction optimizes the performance of ML-based IDS, specifically for detecting DDoS attacks in standardized IoT networks. And also added an ensemble methods Voting Classifier and Stacking CLassifier combining the predictions of multiple individual models to produce a more robust and accurate final prediction. Voting Classifier and Stacking Classifier got 99% .And also we can built the front end using the flask framework for user testing and with user authentication for Attack Detection in the Standardized Internet of Things

Index terms *- Distributed Denial of Service (DDoS), feature engineering, Internet of Things (IoT), intrusion detection, machine learning (ML), NIDS.*

1. INTRODUCTION

The Internet of Things (IoT) enables a vision where devices with sensing, actuating, computing, and communication capabilities can connect with each other. The IoT offers realization of a number of modern applications available today like smart cities, smart transportation, smart health care, etc. The exponential growth of IoT is fueled by the widespread adoption of the Internet and the continuous depreciation in the cost and size of sensor technology during the last few decades. Other influencing factors include the availability of cheap Internet connections, development of built-in sensors in devices, the mobile revolution, and a multitude of companies developing necessary IoT applications and software. By 2030, the IoT will see an explosive growth, where the total number of Internet connected smart devices will be 125 million [1]. The IoT is establishing itself in multiple domains and digitizing the physical world. But with digitization and the direct impact of IoT on physical world, concerns related to security and privacy are also increasing. The resource constraints, self-organizing, and open nature of the IoT make it pregnable to numerous extrinsic and intrinsic attacks. The literature in the field of securing the IoT networks is focused on first and second line of defenses mechanisms. The first line of defense includes cryptographic algorithms on encryption, authentication, and key exchange which are often absent or broken in realistic scenarios. Even in the presence of these algorithms, the network is vulnerable to internal attacks. Thus, second line of defense mechanisms like intrusion detection system (IDS) provide protection to wide variety of internal and external attacks. One of the most crucial and challenging security issues is the attack on the availability of resources of IoT networks with a Distributed Denial-of-Service (DDoS) attack, a well-

known threat. In this work, we emphasize on securing the IoT networks from DDoS attacks with IDS.

Techniques of DDoS detection include many approaches including machine learning (ML). Due to the huge data generated by IoT devices, ML techniques are quite helpful for continuous observations and in-depth analysis of IoT networks. Therefore, developing ML-based IDS [27, 32, 34, 35] seems to be a promising solution to secure IoT networks. ML-based algorithms, however, require data to be trained on and, thus, there is a need for IoT network traffic data set. As IoT consists newer protocols, there is a lack of appropriate data sets for training and evaluating ML-based IDS. Due to the deficiency of authentic IoT data sets, ML-based IDS lacks accurate and uniform performance advancements.

To address this issue, we explore a novel cross layer intrusion detection data set for IoT (IoT-CIDDS) based on standardized IoT protocol stack [2] proposed by us in our previous work in [3]. The data set addresses the gaps in existing IoT data sets by simulating IoT network with border router, sensor nodes, and wired and wireless connection links and explores five DDoS attacks in IoT [1, 2, 4, 6, 7, 8, 10], namely, hello flooding (HF), UDP flooding, selective forwarding (SF), Blackhole (BH), and the ICMPv6 flooding attack. IoT-CIDDS data set is publicly available1 and includes traffic for nodes ranging from 10 to 100. For DDoS detection, data sets are usually considered for evaluation and creation of ML models. However, in this work, our emphasis is to study feature engineering on IoT-CIDDS followed by ML for DDoS detection. The complete analysis is mapped on the proposed holistic framework of feature engineering for obtaining significant features and adaptability of ML models after optimal tuning of hyperparameters. This leads us to a comprehensive approach for selection of significant features for each attack and, thus, considerable reduction in processing overhead while training the model.

2. LITERATURE SURVEY

With the widespread expansion of internet connected devices, securing the Internet of Things (IoT) [1, 2, 4, 6, 7, 8, 10], has become one of the biggest challenges and a crucial issue for ensuring a secure and robust IoT vision. The usage of adhoc topologies and the resource constraints of IoT devices and networks makes first line of defense mechanisms like cryptography unsuitable to implement. Therefore, second line of defense mechanisms such as the intrusion detection system become fundamental to detect attacks. In this work [3], we propose IoT-Sentry, a cross-layer intrusion detection system to detect five different attacks with zero additional overhead. Also, there is a lack of rich, illustrative and concise public datasets in IoT for evaluation of intrusion detection models for IoT networks [3]. Our research work fills this research gap by (1) developing a novel cross-layer IoT dataset which contains four simulation instances for the five attacks in static IoT networks with up to 100 nodes, (2) utilize novel features of cross-layer attacks, and (3) employ ensemble learning model to detect IoT attacks. In our work, the Cooja IoT [1] simulator has been utilized for generating malicious and benign traffic. The IoT attack dataset is then analyzed and fed into centralized detection module implemented at the 6LoWPAN border router. On evaluation, IoT-Sentry achieves on average an accuracy of 99% for 4 out of 5 attacks. To the best of our knowledge, this is the first time that cross layer intrusion detection system catering to five attacks from different layers is developed for securing standardized IoT networks.

Campuses and cities of the near future will be equipped with vast numbers of IoT devices [1, 2]. Operators of such environments may not even be fully aware of their IoT assets, let alone whether each IoT device is functioning properly safe from cyber-attacks. This paper proposes the use of network traffic analytics to characterize IoT devices, including their typical behaviour mode [4]. We first collect and synthesize traffic traces from a smart-campus environment instrumented with a diversity of IoT devices including cameras, lights, appliances, and health-monitors; our traces, collected over a period of 3 weeks, are released as open data to the public. We then analyze the traffic traces to characterize statistical attributes such as data rates and burstiness, activity cycles, and signalling patterns, for over 20 IoT devices [4, 7] deployed in our environment. Finally, using these attributes, we develop a classification method that can not only distinguish IoT from non-IoT traffic, but also identify specific IoT devices with over 95% accuracy. Our study empowers operators of smart cities and campuses to discover and monitor their IoT assets based on their network behaviour.

The high number of vulnerabilities in Internet of Things devices has created malware-prone networks. A type of malware that imposes a serious threat to the Internet security is known as botnets. This malware exploits some vulnerabilities of IoT devices [12, 21] to infect them and perform large-scale Distributed Denial of Service attacks, affecting many users who depend on their services. [6] This work presents the construction of an experimental environment to

generate a dataset that contains data from a real IoT device that was infected by botnet malware in a laboratory. The dataset can be used to support the development of defence tools for IoT devices to identify botnets, as it contains network traffic and host-based features, such as, CPU and memory usage. The dataset and network environment files are available for the research community.

Cyber threats are a showstopper for Internet of Things (IoT) has recently been used at an industrial scale. Network layer attacks on IoT [21] can cause significant disruptions and loss of information. Among such attacks, routing attacks are especially hard to defend against because of the ad-hoc nature of IoT systems and resource constraints of IoT devices. Hence, an efficient approach for detecting and predicting IoT attacks is needed. Systems confidentiality, integrity and availability depends on continuous security and robustness against routing attacks. [7] We propose a deep-learning based machine learning method for detection of routing attacks for IoT. In our study, the Cooja IoT [12] simulator has been utilized for generation of high-fidelity attack data, within IoT networks ranging from 10 to 1000 nodes. We propose a highly scalable, deep-learning based attack detection methodology for detection of IoT routing attacks which are decreased rank, hello-flood and version number modification attacks, with high accuracy and precision. Application of deep learning for cyber-security in IoT requires the availability of substantial IoT attack data and we believe that the IoT attack dataset produced in this work can be utilized for further research.

The Internet of Things (IoT) is a reality that changes several aspects of our daily life, from smart home monitoring to the management of critical infrastructure. The “Routing Protocol for low power and Lossy networks” (RPL) is the only de-facto standardized routing protocol in IoT networks and is thus deployed in environmental monitoring, healthcare, smart building, and many other IoT applications [22, 23, 25, 28]. In literature, we can find several attacks aiming to affect and disrupt RPL-based networks. Therefore, it is fundamental to develop security mechanisms that detect and mitigate any potential attack in RPL-based networks. Current state-of-the-art security solutions deal with very few attacks while introducing heavy mechanisms at the expense of IoT devices and the overall network performance. In this work [8], we aim to develop an Intrusion Detection System (IDS) capable of dealing with multiple attacks while avoiding any RPL overhead. The proposed system is called DETONAR - DETector of rOutiNg Attacks in Rpl - and it relies on a packet sniffing approach. DETONAR uses a combination of signature and anomaly-based rules to identify any malicious behavior in the traffic (e.g., application and DIO packets). To the best of our knowledge, there are no exhaustive datasets containing RPL traffic for a vast range of attacks. To overcome this issue and evaluate our IDS, we propose RADAR - Routing Attacks DAtaset for Rpl: the dataset contains five simulations for each of the 14 considered attacks in 16 static-nodes networks. DETONAR’s attack detection exceeds 80% for 10 attacks out of 14, while maintaining false positives close to zero.

3. METHODOLOGY

i) Proposed Work:

The proposed system is a feature engineering and machine learning framework for detecting DDoS attacks in IoT networks using the IoT-CIDDS dataset [2, 3]. It consists of two phases: data set enrichment and advanced feature engineering, and the development of a machine learning model. The framework optimizes the performance of ML-based IDS for detecting DDoS attacks in standardized IoT networks. The implementation incorporates ensemble methods Voting Classifier and Stacking Classifier, achieving an impressive 99% accuracy in attack detection within standardized Internet of Things (IoT) networks. These ensemble methods combine predictions from multiple individual models, enhancing the overall robustness and accuracy of the intrusion detection system. To make this technology accessible and user-friendly, a front-end application is developed using the Flask framework, providing a web interface for users to test and interact with the system. Additionally, user authentication features are integrated to ensure secure access, enhancing the overall usability and reliability of the attack detection system in real-world IoT environments.

ii) System Architecture:

ML algorithms, in conjunction with feature engineering can be employed to achieve the best detection in a particular data set. This study proposes a holistic framework that includes a detailed examination of features followed by ML optimizations. It underlines the need of dealing with the problem of overfitting in ML outcomes [10] . Fig. 1 depicts the suggested framework. The proposed framework ensures that features are treated systematically according to the type of attack in IoT-CIDDS. The framework can be generalized to any data

set for removing inbuilt problems in data like skewness for later phases of ML. After statistical exploration of features in the data set, features are normalized and selected for further analysis by ML algorithms. In Fig. 1, five ML algorithms are shown but the framework can be extended to any ML algorithm whether supervised, unsupervised or semi-supervised. Here, the emphasis is laid on supervised ML techniques because of the nature of data set under consideration where the labeling attribute is available.

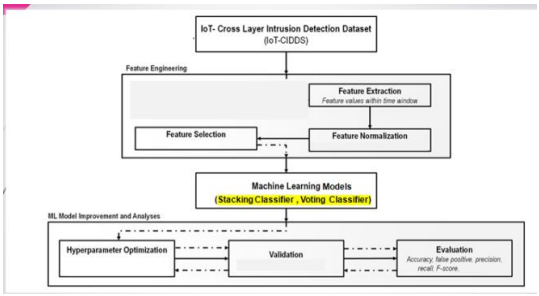

Fig 1 Proposed architecture

iii) Dataset collection:

The dataset used in the project, IoT-CIDDS[2, 3], is a novel cross-layer intrusion detection dataset for IoT networks. It simulates diverse IoT network scenarios with border routers, sensor nodes, and wired/wireless connections, including traffic for nodes ranging from 10 to 100, and addresses deficiencies in existing datasets by providing realistic IoT-compatible traffic data for evaluating machine learning-based intrusion detection systems.

	duration	proto	packets	bytes	flows	tcp_urg	tcp_ack	tcp_psh	tcp_rst	tcp_syn	tcp_fin	tos	label	attack_type	attack_id
0	0.018	TCP	2	338.0	1	0	1	1	0	0.0	0.0	0	normal	benign	0
1	0.000	TCP	1	212.0	1	0	1	1	0	0.0	0.0	32	normal	benign	0
2	0.000	TCP	1	108.0	1	0	1	1	0	0.0	0.0	0	normal	benign	0
3	0.006	TCP	2	174.0	1	0	1	1	0	0.0	0.0	0	normal	benign	0
4	0.019	TCP	2	338.0	1	0	1	1	0	0.0	0.0	0	normal	benign	0

Fig 2 CIDDS dataset

iv) Data Processing:

Data processing involves transforming raw data into valuable information for businesses. Generally, data scientists process data, which includes collecting, organizing, cleaning, verifying, analyzing, and converting it into readable formats such as graphs or documents. Data processing can be done using three methods i.e., manual, mechanical, and electronic. The aim is to increase the value of information and facilitate decision-making. This enables businesses to improve their operations and make timely strategic decisions. Automated data processing solutions, such as computer software programming, play a significant role in this. It can help turn large amounts of data, including big data, into meaningful insights for quality management and decision-making.

v) Feature selection:

Feature selection is the process of isolating the most consistent, non-redundant, and relevant features to use in model construction. Methodically reducing the size of datasets is important as the size and variety of datasets continue to grow. The main goal of feature selection is to improve the performance of a predictive model and reduce the computational cost of modeling.

Feature selection, one of the main components of feature engineering, is the process of selecting the most important features to input in machine learning algorithms. Feature selection techniques are employed to reduce the number of input variables by eliminating redundant or irrelevant features and narrowing down the set of features to those most relevant to the machine learning model. The main benefits of performing feature selection in advance, rather than letting the machine learning model figure out which features are most important.

vi) Algorithms:

1. DECISION TREE:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions [43] .

MM

Decision Tree ¶

```
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth=30)

# fit the model
tree.fit(X_train, y_train)

#predicting the target value from the model for the samples

y_pred = tree.predict(X_test)

dt_acc = accuracy_score(y_pred, y_test)
dt_prec = precision_score(y_pred, y_test)
dt_rec = recall_score(y_pred, y_test)
dt_f1 = f1_score(y_pred, y_test)
dt_auroc = roc_auc_score(y_test, tree.predict_proba(X_test)[:, 1])
```

```
storeResults('Decision Tree Classifier',dt_acc,dt_prec,dt_rec,dt_f1,dt_auroc)
```

Fig 3 Decision tree

2. LOGISTIC REGRESSION:

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1

Fig 4 Logistic regression

3. MLP:

Multi-Layer perceptron defines the most complex architecture of artificial neural networks. It is substantially formed from multiple layers of the perceptron MLP networks are used for supervised learning format. A typical learning algorithm for MLP networks is also called **back propagation's algorithm**.A multilayer perceptron (MLP) is a feed forward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input nodes connected as a directed graph between the input and output layers. MLP uses backpropagation for training the network. MLP is a deep learning method [27].

MLP

```
from sklearn.neural_network import MLPClassifier
# instantiate the model
mlp = MLPClassifier(random_state=1, max_iter=30)

# fit the model
mlp.fit(X_train, y_train)

#predicting the target value from the model for the samples

y_pred = mlp.predict(X_test)

mlp_acc = accuracy_score(y_pred, y_test)
mlp_prec = precision_score(y_pred, y_test)
mlp_rec = recall_score(y_pred, y_test)
mlp_f1 = f1_score(y_pred, y_test)
mlp_auroc = roc_auc_score(y_test, mlp.predict_proba(X_test)[:, 1])
```

```
storeResults('MLP Classifier',mlp_acc,mlp_prec,mlp_rec,mlp_f1,mlp_auroc)
```

Fig 5 MLP

4.SUPPORTVECTORMACHINEALGORITHMS

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.SVM chooses the extreme points/vectors that help in creating the hyperplane [27].

SVM

```
from sklearn.svm import SVC

# instantiate the model
svm = SVC(probability=True)

# fit the model
svm.fit(X_train, y_train)

#predicting the target value from the model for the samples

y_pred = svm.predict(X_test)

svc_acc = accuracy_score(y_pred, y_test)
svc_prec = precision_score(y_pred, y_test)
svc_rec = recall_score(y_pred, y_test)
svc_f1 = f1_score(y_pred, y_test)
svc_auroc = roc_auc_score(y_test, svm.predict_proba(X_test)[:, 1])
```

```
storeResults('SVM',svc_acc,svc_prec,svc_rec,svc_f1,svc_auroc)
```

Fig 6 SVM

5. RANDOM FOREST:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.* As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. **The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting [43].**

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
rf = RandomForestClassifier(random_state=40)

# fit the model
rf.fit(X_train, y_train)

#predicting the target value from the model for the samples

y_pred = rf.predict(X_test)

rf_acc = accuracy_score(y_pred, y_test)
rf_prec = precision_score(y_pred, y_test)
rf_rec = recall_score(y_pred, y_test)
rf_f1 = f1_score(y_pred, y_test)
rf_auroc = roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1])
```

```
storeResults('Random Forest Classifier',rf_acc,rf_prec,rf_rec,rf_f1,rf_auroc)
```

Fig 7 Random forest

6. STACKING CLASSIFIER:

Stacking is one of the popular ensemble modeling techniques in machine learning. Various weak learners are ensembled in a parallel manner in such a way that by combining them with Meta learners, we can predict better predictions for the future. This ensemble technique works by applying input of combined multiple weak learners' predictions and Meta learners so that a better output prediction model can be achieved. In stacking, an algorithm takes the outputs of sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction. Stacking is also known as **a stacked generalization** and is an extended form of the Model Averaging Ensemble technique in which all sub-models equally participate as per their performance weights and build a new model with better predictions. This new model is stacked up on top of the others; this is the reason why it is named stacking.

Stacking Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from lightgbm import LGBMClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import StackingClassifier

estimators = [('rf', RandomForestClassifier(n_estimators=10)),('mlp', MLPClassifier(random_state=1, max_iter=30))]

clf = StackingClassifier(estimators=estimators, final_estimator=LGBMClassifier(n_estimators=10))


clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)

stac_acc = accuracy_score(y_pred, y_test)
stac_prec = precision_score(y_pred, y_test)
stac_rec = recall_score(y_pred, y_test)
stac_f1 = f1_score(y_pred, y_test)
stac_auroc = roc_auc_score(y_test, clf.predict_proba(X_test)[:, 1])
```

```
storeResults('Stacking Classifier',stac_acc,stac_prec,stac_rec,stac_f1,stac_auroc)
```

Fig 8 Stacking classifier

7. VOTING CLASSIFIER :

A voting classifier is a machine learning model that gains experience by training on a collection of several models and forecasts an output (class) based on the class with the highest likelihood of becoming the output. To forecast the output class based on the largest majority of votes, it averages the results of each classifier provided into the voting classifier. The concept is to build a single model that learns from various models and predicts output based on their aggregate majority of votes for each output class, rather than building separate specialized models and determining the accuracy for each of them [32].

Voting Classifier

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier, AdaBoostClassifier
clf1 = RandomForestClassifier()
clf2 = AdaBoostClassifier()
clf3 = DecisionTreeClassifier()

eclf1 = VotingClassifier(estimators=[('ab', clf1), ('rf', clf2), ('dt', clf3)], voting='soft')
eclf1.fit(X_train, y_train)

y_pred = eclf1.predict(X_test)
y_prob = eclf1.predict_proba(X_test)

vot_acc = accuracy_score(y_pred, y_test)
vot_prec = precision_score(y_pred, y_test)
vot_rec = recall_score(y_pred, y_test)
vot_f1 = f1_score(y_pred, y_test)
vot_auroc = roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1])
```

```
storeResults('Voting Classifier',vot_acc,vot_prec,vot_rec,vot_f1,vot_auroc)
```

Fig 9 Voting classifier

4. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

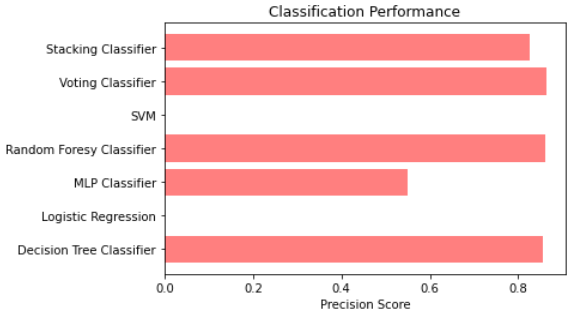

Fig 10 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

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

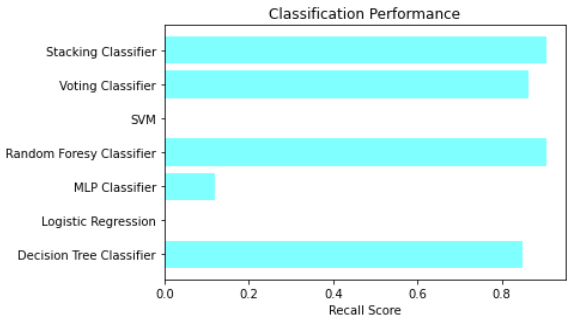

Fig 11 Recall comparison graph

Accuracy: Accuracy is the proportion of correct predictions in a classification task, measuring the overall correctness of a model's predictions.

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

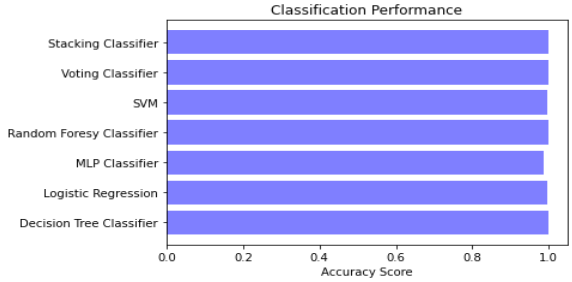

Fig 12 Accuracy graph

F1 Score: The F1 Score is the harmonic mean of precision and recall, offering a balanced measure that considers both false positives and false negatives, making it suitable for imbalanced datasets.

$$\text{F1 Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$

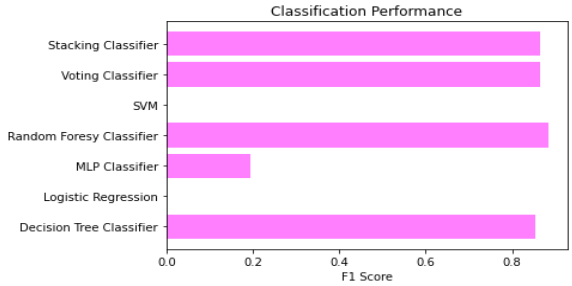

Fig 13 F1Score

Algorithms	Accuracy	Precision	Recall	F1 - score
Decision Tree Classifier	0.999	0.856	0.850	0.853
Logistic Regression	0.997	0.000	0.000	0.000
MLP Classifier	0.988	0.549	0.119	0.195
Random Forest Classifier	0.999	0.862	0.906	0.884
SVM	0.997	0.000	0.000	0.000
Extension Voting Classifier	0.999	0.864	0.864	0.864
Extension Stacking Classifier	0.999	0.825	0.906	0.864

Result: **There is an Attack Detected!**

Fig 19 Predict result for given input

5. CONCLUSION

The successful implementation of the ML-based IDS [32, 34, 35] is crucial for enhancing the security of IoT

networks, preventing and mitigating potential Distributed Denial-of-Service (DDoS) attacks. This is particularly significant as IoT devices become more integrated into daily life and critical infrastructure. The insights into feature engineering techniques offer a practical guide for optimizing Machine Learning (ML) models used in intrusion detection. This is essential for achieving accurate and efficient DDoS detection while reducing processing overhead, making it applicable to resource-constrained IoT environments [1, 3]. The algorithm, utilizing ensemble methods, achieved an impressive 99% accuracy in attack detection. Through user-friendly front-end testing with input feature values, its robust and reliable performance underscores its effectiveness and real-world applicability. The integration of Flask and SQLite for user authentication adds a practical dimension to the project. This feature makes the developed framework user-friendly and accessible, benefiting administrators and security personnel responsible for managing and monitoring IoT networks [4, 6]. The project's outcomes are beneficial to researchers, cybersecurity professionals, IoT device manufacturers, and organizations utilizing IoT technologies. It provides a valuable resource for understanding and implementing effective DDoS detection mechanisms in the rapidly growing IoT landscape, ultimately contributing to a more secure and resilient digital ecosystem.

6. FUTURE SCOPE

Future enhancements could involve incorporating advanced machine learning techniques or deep learning models to further improve the accuracy and efficiency of intrusion detection in IoT networks. Exploring novel algorithms and architectures may contribute to more robust security solutions. Extending the project to enable real-time monitoring and response capabilities would be a valuable future scope. Implementing mechanisms for immediate threat detection and mitigation in dynamic IoT environments could enhance the overall security posture [12, 21]. As IoT protocols continue to evolve, ensuring the project's scalability to adapt and integrate with emerging standards will be crucial. Supporting a wide range of IoT protocols will enhance the applicability of the intrusion detection system across diverse IoT ecosystems [22, 23]. Future efforts could involve collaboration with industry stakeholders to validate and implement the proposed intrusion detection system in real-world IoT deployments. Practical implementations and feedback from industry partners would provide valuable insights for further refinement and optimization of the system.

REFERENCES

[1] I. Markit. “The Internet of Things: A movement, not a market,” 2018. [Online]. Available: https://cdn.ihs.com/www/pdf/IoTebook.pdf

[2] M. R. Palattella et al., “Standardized protocol stack for the Internet of (important) Things,” IEEE Commun. Surveys Tuts., vol. 15, no. 3, pp. 1389–1406, 3rd Quart., 2013.

[3] Kamaldeep, M. Malik, M. Dutta, and J. Granjal, “IoT-sentry: A cross-layer-based intrusion detection system in standardized Internet of Things,” IEEE Sensors J., vol. 21, no. 24, pp. 28066–28076, Dec. 2021.

[4] A. Sivanathan et al., “Characterizing and classifying IoT traffic in smart cities and campuses,” in Proc. IEEE Conf. Comput. Commun. Workshops, 2017, pp. 559–564.

[5] A. Verma and V. Ranga. “RPL-NIDDS17: A data set for intrusion detection in RPL based 6LoWPAN networks (Internet of Things).” [Online]. Available: https://zenodo.org/record/1406034#.XLfrI-gzbIU

[6] V. H. Bezerra, V. G. T. da Costa, R. A. Martins, S. Barbon, R. S. Miani, and B. B. Zarpelao, “Providing IoT host-based datasets for intrusion detection research,” in Proc. 18th Braz. Symp. Inf. Security Comput. Syst., 2018, pp. 15–28.

[7] F. Y. Yavuz, D. Ünal, and E. Gül, “Deep learning for detection of routing attacks in the Internet of Things,” Int. J. Comput. Intell. Syst., vol. 12, no. 1, pp. 39–58, 2018.

[8] A. Agiollo, M. Conti, P. Kaliyar, T.-N. Lin, and L. Pajola, “DETONAR: Detection of routing attacks in RPL-based IoT,” IEEE Trans. Netw. Service Manag., vol. 18, no. 2, pp. 1178–1190, Jun. 2021.

[9] M. R. Shahid, “Deep learning for Internet of Things (IoT) network security,” Ph.D. dissertation, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, France, 2021.

[10] “NetSimv12.1.” Tetcos. 2019. [Online].Available: https://www.tetcos. com/netsim-pro.html

[11] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, “Crosslevel sensor network simulation

with COOJA," in Proc. 31st IEEE Conf. Local Comput. Netw., 2006, pp. 641–648.

[12] A. Verma and V. Ranga, "Evaluation of network intrusion detection systems for RPL based 6LoWPAN networks in IoT," Wireless Pers. Commun., vol. 108, no. 3, pp. 1571–1594, 2019.

[13] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," IETF, RFC 7252, Jun. 2014. [Online]. Available: http:// www.rfc-editor.org/rfc/rfc7252.txt

[14] J. Postel, "User datagram protocol," IETF, RFC 768, Aug. 1980. [Online]. Available: http://www.rfc-editor.org/rfc/rfc768.txt

[15] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification," IETF, RFC 2460, Dec. 1998. [Online]. Available: https:// datatracker.ietf.org/doc/html/rfc2460

[16] T. Winter et al., "RPL: IPv6 routing protocol for low-power and lossy networks," IETF, RFC 6550, Mar. 2012. [Online]. Available: http://www. rfc-editor.org/rfc/rfc6550.txt

[17] A. Conta, S. Deering, and M. Gupta, "Internet control message protocol (ICMPv6) for the Internet protocol version 6 (IPv6) specification," IETF, RFC 4443, Mar. 2006. [Online]. Available: https://datatracker.ietf.org/ doc/html/rfc4443

[18] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks," IETF, RFC 4944, Sep. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4944.txt

[19] IEEE Standard for Low-Rate Wireless Networks Corrigendum 1, IEEE 802.15.4-2015/Cor1-2018, 2015. [Online]. Available: https://standards. ieee.org/standard/802.15.4-2015-Cor1-2018.html

[20] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer Internet of Things devices," in Proc. IEEE Security Privacy Workshops (SPW), San Francisco, CA, USA, 2018, pp. 29–35.

[21] K. Bhardwaj, J. C. Miranda, and A. Gavrilovska, "Towards IoTDDoS prevention using edge computing," in Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge), Boston, MA, USA, 2018, pp. 1–7.

[22] N. Ravi and S. M. Shalinie, "Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture," IEEE Internet Things J., vol. 7, no. 4, pp. 3559–3570, Apr. 2020.

[23] Y.-W. Chen, J.-P. Sheu, Y.-C. Kuo, and N. Van Cuong, "Design and implementation of IoT DDoS attacks detection system based on machine learning," in Proc. Eur. Conf. Netw. Commun. (EuCNC), 2020, pp. 122–127.

[24] D. Yin, L. Zhang, and K. Yang, "A DDoS attack detection and mitigation with software-defined Internet of Things framework," IEEE Access, vol. 6, pp. 24694–24705, 2018.

[25] I. Cvitic, D. Perakovi ´ c, M. Periša, and M. Botica, "Novel approach for ´ detection of IoT generated DDoS traffic," Wireless Netw., vol. 27, no. 3, pp. 1573–1586, 2021.

[26] I. Cvitic, D. Perakovic, B. B. Gupta, and K.-K. R. Choo, "Boosting- ´ based DDoS detection in Internet of Things systems," IEEE Internet Things J., vol. 9, no. 3, pp. 2109–2123, Feb. 2022.

[27] M. N. Napiah, M. Y. I. B. Idris, R. Ramli, and I. Ahmedy, "Compression header analyzer intrusion detection system (CHA-IDS) for 6LoWPAN communication protocol," IEEE Access, vol. 6, pp. 16623–16638, 2018.

[28] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge," IEEE Trans. Netw. Service Manag., vol. 17, no. 1, pp. 45–59, Mar. 2020.

[29] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma. "IoT-Dataset." 2019. [Online]. Available: https://bit.ly/31uuNK7

[30] E. Canbalaban and S. Sen, "A cross-layer intrusion detection system for RPL-based Internet of Things," in Proc. Int. Conf. Ad-Hoc Netw. Wireless, 2020, pp. 214–227.

[31] A. Abbas, M. A. Khan, S. Latif, M. Ajaz, A. A. Shah, and J. Ahmad, "A new ensemble-based intrusion detection system for Internet of Things," Arab. J. Sci. Eng., vol. 47, no. 16, pp. 1805–1819, 2022.

[32] "Intrusion detection evaluation dataset (CICIDS2017)." Canadian Institute for Cybersecurity.

Accessed: Sep. 29, 2019. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[33] P. Verma et al., "A novel intrusion detection approach using machine learning ensemble for IoT environments," Appl. Sci., vol. 11, no. 21, pp. 1–21, 2021.[Online].Available: https://www.mdpi.com/2076-3417/ 11/21/10268

[34] "Amazon's AWS LAN network (CIC-AWS-2018)." Canadian Institute for Cybersecurity. Accessed: Jan. 30, 2022. [Online]. Available: https:// registry.opendata.aws/CSE-CIC-IDS2018-V2/

[35] A. Verma and V. Ranga, "ELNIDS: Ensemble learning based network intrusion detection system for RPL based Internet of Things," in Proc. 4th Int. Conf. Internet Things Smart Innov. Usages, 2019, pp. 1–6.

[36] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw., Nov. 2004, pp. 455–462.

[37] J. Crnogorac, J. Crnogorac, M. Vucini ˇ c, E. Ko ´ can, and T. Watteyne, ˇ "Dense multi-channel sniffing in large IoT networks," IEEE Access, vol. 10, pp. 105101–105110, 2022.

[38] B. Mostafa, M. Molnar, M. Saleh, A. Benslimane, and S. Kassem, "Optimal proactive monitor placement & scheduling for IoT networks," J. Oper. Res. Soc., vol. 73, no. 11, pp. 2431–2450, 2021.

[39] M. Malik, M. Dutta, and J. Granjal, "A survey of key bootstrapping protocols based on public key cryptography in the Internet of Things," IEEE Access, vol. 7, pp. 27443–27464, 2019.

[40] K. V. Mardia, "Measures of multivariate skewness and kurtosis with applications," Biometrika, vol. 57, no. 3, pp. 519–530, 1970.

[41] J. M. Bland and D. G. Altman, "Calculating correlation coefficients with repeated observations: Part 2—Correlation between subjects," BMJ, vol. 310, no. 6980, p. 633, 1995.

[42] N. Moustafa, J. Slay, and G. Creech, "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks," IEEE Trans. Big Data, vol. 5, no. 4, pp. 481–494, Dec. 2019.

[43] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, no. 85, pp. 2825–2830, 2011.

[44] "Tuning the hyper-parameters of an estimator." 2020. [Online]. Available: https://scikit-learn.org/stable/modules/gridsearch.html

[45] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "FlowGuard: An intelligent edge defense mechanism against IoT DDoS attacks," IEEE Internet Things J., vol. 7, no. 10, pp. 9552–9562, Oct. 2020.

[46] M. H. Aysa, A. A. Ibrahim, and A. H. Mohammed, "IoT DDoS attack detection using machine learning," in Proc. 4th Int. Symp. Multidiscip. Stud. Innov. Technol. (ISMSIT), Istanbul, Turkey, 2020, pp. 1–7.

AUTHOR PROFILE:

[1] Mr. K. Jaya Krishna, currently working as an Associate Professor in the Department of Master of Computer Applications, QIS College of Engineering and Technology, Ongole, Andhra Pradesh. He did his MCA from Anna University, Chennai, M.Tech (CSE) from JNTUK, Kakinada. He published more than 10 research papers in reputed peer reviewed Scopus indexedjournals. He also attended and presented research papers in different national and international journals and the proceedings were indexed IEEE. His area of interest is Machine Learning, Artificial intelligence, Cloud Computing and Programming Languages.

[2] Mr.Kakunuri Rajasekhar currently pursuing Master of Computer Applications at QIS College of Engineering and Technology (Autonomous), Ongole, Andhra Pradesh. He Completed B.Sc. in

computer science from G.R.K DEGREE COLLEGE Ongole ,Andhra Pradesh. His areas of interests are cloud Computing& Machine learning.